Дж.Д. Мейер, Джейсон Тейлор,
Алекс Макман, Прашант Бансод, Кевин Джонс

Командная разработка с использованием Visual Studio Team Foundation Server

Microsoft® Visual Studio® Team System 2008

Оглавление

Предисловие

Прежде чем выпустить Microsoft® Visual Studio® 2005 Team Foundation Server (TFS), мы в последние полтора года активно применяли его для управления циклом его же разработки — что называется, «помоги себе сам». Благодаря этому «самоедству» мы многое узнали о создаваемой нами системе. Разумеется, попутно мы выловили и исправили множество ошибок, благодаря чему продукт получился более устойчивым и производительным. Но важнее то, что в процессе работы нам удалось глубже понять, как нужно (или не нужно) использовать создаваемый инструментарий. Это понимание, а также обмен опытом с нашими клиентами, и составили основу представляемого руководства.

На первый взгляд может показаться, что эту информацию логичнее публиковать вместе с документацией продукта или даже вместо нее. Сказать по правде, на протяжении какого-то времени я сам придерживался этой точки зрения. Однако тесное сотрудничество с Дж. Д. Мейером (J.D. Meier) и другими авторами руководства доказало мне, что его отделение от документации не только естественно, но и необходимо. Я думаю, разницу между ними уместнее всего сравнить с разницей между инструкцией по эксплуатации автомобиля и шоферским справочником — и то, и другое полезно, но по разным причинам. Как правило, команда продукта больше занимается его документацией в расчете на то, что инструкцию напишут другие. Но в данном случае мы решили, не отказываясь от посторонней помощи, уделить больше своего времени и сил именно практической части, поскольку осознаем, как важно это с точки зрения успешного продвижения продукта и удовлетворения требований пользователей.

Как и автомобиль, TFS готов доставить и вас, и вашу команду куда угодно. Это руководство поможет вам в пути. У каждой команды — собственный подход к TFS, определяемый конкретными потребностями и историей развития. Поэтому в процессе написания руководства мы старались сделать так, чтобы его можно было читать как от корки до корки, охватив всю картину целиком, так и фрагментарно, чтобы глубже ознакомиться именно с той темой, которая нужна в данный момент.

Стимулом к созданию руководства послужили в первую очередь пожелания клиентов. Да и теперь они продолжают играть важную роль в выборе направления и постановке задач. Мы убеждены, что тесное сотрудничество

с сообществом пользователей при написании подобных книг позволяет сделать их наиболее полезными и, в конечном итоге, наиболее успешными. По этой причине в составлении списка тематик, подборе описываемых методик и организации содержимого нам помогали реальные разработчики. И наш коллективный труд еще не окончен. Пожалуйста, и дальше помогайте нам совершенствовать это руководство, сообщайте, о чем еще нужно написать. Площадь соприкосновения TFS с внешним миром столь велика, что даже мы бываем иногда поражены этим. Благодаря вашим советам мы поможем клиентам с максимальной эффективностью пользоваться средствами, которые мы разрабатываем.

Назначение TFS — организовать работу команды и привести ее к созданию великолепных программ. Поскольку TFS сам разрабатывался при помощи TFS, нам также удалось организовать работу команды и получить — надеюсь, вы согласитесь со мной — великолепный продукт. Это руководство поможет вам и вашей команде осознать его мощь при выполнении вашего следующего проекта.

Всего наилучшего!

Джефф Билер
Руководитель персонала,
Visual Studio Team System
Июль 2007 года

Джефф Билер (Jeff Beehler) руководит персоналом Team System.Он начал свою карьеру в Майкрософт в 1990 году после окончания Университета Колорадо и работал над первыми версиями Visual C++. В 1996 году он ушел из Майкрософт и занимался собственными делами, включая консалтинг, преподавание в начальной школе и организацию семейной жизни. В 2003 году он вернулся в Microsoft и начал работать в команде Visual Studio Team System. Он участвует в самых разных этапах разработки продукта, от планирования до выпуска в свет. Он с энтузиазмом использует все компоненты Team System, чтобы улучшить их же разработку. Свободное от работы время Джефф проводит с семьей, фотографирует и играет на открытом воздухе великого Северо-Запада.

С первых дней существования Visual Studio Team System мы понимали, что группам разработки программного обеспечения (ПО) понадобится больше информации, чем мы сможем предоставить им до начала поставок. Скажем, совсем не помешают проверенное руководство и полезные советы. Но им неоткуда взяться, пока продукт не прошел испытание в разнообразных командах, обстоятельствах, проектах и сценариях, которые продемонстрировали бы: вот это работает, а это нет.

К несчастью, на написание руководств и выявление эффективных методик уходит время. На протяжении нескольких последних лет мы многое узнали об использовании Team System вообще и Team Foundation Server в частности. Добыть эти познания было нелегко. Даже у таких ветеранов группы

Patterns & Practices, как Дж. Д. Мейер и его команда, эта тонкая и методическая работа заняла не один месяц.

Но ожидание, наконец, подошло к концу! Эта книга представляет собой плод коллективных усилий людей, которые приняли явное или неявное участие в ее создании. Составившая ее команда не игнорировала предшествующий опыт. Ими просмотрено огромное количество блогов, обсуждений в форумах, статей и прочего материала, необходимого, чтобы понять, как именно Team System используется «в условиях дикой природы».

Попутно авторы выделили ключевые факторы, влияющие на работу команд по разработке ПО, и выявили методики, приводящие к предсказуемому и повторяемому успеху. В самых информативных разделах описаны некоторые ключевые задачи Team Foundation Server, например, отслеживание рабочих элементов, отчеты и шаблоны процессов.

Оглядываясь назад, я с удовольствием отмечаю, что нам, как команде по подготовке документации, хватило рассудка отложить эту работу, вместо того чтобы заполнить документацию догадками. Приношу свои извинения тем, кто страдал от отсутствия информации, и благодарю тех, кто выстоял и стал пионером применения Team System.

Роб Кэрон
Ведущий менеджер продукта
Microsoft Corporation
Июль 2007 года

Роб Кэрон (Rob Caron) — ведущий менеджер продукта по разработке стратегии содержимого. Работу в Макрософт Роб начал в 1999 году в качестве автора документации Visual Studio. В последующие годы он внес вклад в документацию Visual Studio .NET 2002, Visual Studio .NET 2003 и Visual Studio Team System. В середине 2004 года он завел блог, ставший средоточием информации о Team System. После семи лет писательского труда осенью 2006 года Роб перешел в группу Developer Marketing. Сейчас он возглавляет команду, которая пытается хоть немного упростить все усложняющуюся деятельность разработчиков Макрософт.

Процесс разработки программного обеспечения может быть очень сложным. Созданием ПО занимаются как очень маленькие группы, так и команды, объединяющие тысячи человек с массой специальных ролей. Приступая к работе над Microsoft® Visual Studio® Team System, мы хотели создать инструменты, которые помогут разработчикам представлять, организовывать, проектировать, планировать, собирать, тестировать, разворачивать приложения и управлять ими. Team Foundation Server (TFS) — это центральный элемент Team System, связывающий воедино всех людей и работы, вовлеченные в жизненный цикл приложения.

В результате Team System обладает широкими возможностями, поскольку спроектирована согласно потребностям разных людей, выполняющих

разные роли. Мы постарались сделать продукт максимально простым, но при этом достаточно гибким с возможностью адаптации к нуждам разнообразнейших групп. За последний год стало очевидным, что пользователям Team System не хватает ключевого компонента. Имеется масса документации и различных руководств, рассказывающих, «как это сделать», но не хватает хорошей книги о том, как это «должно быть». Руководство по процессу в Team System частично заполняет этот пробел, но оно преимущественно сосредоточено на людях, ролях и последовательности операций. Требовалась по-настоящему цельная документация по стратегии разработки ПО.

Новое руководство по TFS — именно то, что нужно. Оно охватывает широкий диапазон тем, от организации командных проектов до выбора структуры ветвления и слияния, от стратегии выбора числа и периодичности сборок до управления проектом с помощью Team System. Все вопросы в данном руководстве рассматриваются на примере лучших методик, в контексте применения в конкретных условиях.

Это руководство преимущественно основано на практике разработки ПО, созданной и принятой в Макрософт на протяжении последних 30 лет. Больше того, здесь учтен реальный опыт использования TFS, поскольку уже более двух лет Отдел разработки (Developer Division) корпорации применяет TFS в реальной работе. Я искренне надеюсь, что наш опыт и рекомендации будут вам полезны.

Брайан Гарри
Член технического совета (Technical Fellow),
Visual Studio Team System Microsoft Corporation
Июль 2007 года

Брайан Гарри (Brian Harry) — руководитель отдела по разработке продуктов для Team Foundation Server. Брайан всегда питал особую страсть к инструментальным средствам разработки ПО, еще со студенческих времен в середине 1980-х в Университете Серверной Каролины, когда он экспериментировал с компиляторами, компоновщиками, ассемблерами, системами моделирования процессоров и т. д. Придя в Макрософт, Брайан работал в отделе, который в то время назывался Tools and Databases (инструментальные средства и базы данных). Пару лет он проработал в SourceSafe, а затем в Microsoft Repository. Он участвовал в разработке .NET Framework в качестве руководителя группы исследований и разработки для среды CLR и руководителя отдела по разработке продукта. В конце 2002 года он вернулся в Северную Каролину, чтобы помочь в открытии центра разработки инструментария для разработчиков. Брайан собрал там команду примерно в 50–60 человек, которая работает над продуктами Team Foundation Server и Visual Studio Team System for Testers.

Введение

Эта книга поможет вам повысить эффективность командной разработки ПО при помощи Visual Studio 2005 Team Foundation Server. Она будет интересна тем, кто уже работает с Team Foundation Server, и тем, кто только приступает к его использованию.

Основу справочника составили отзывы клиентов, практический опыт поддержки продукта, а также сбор сведений о его использовании. Данный справочник составлен в соответствии с решаемыми задачами и разделен на следующие части:

- **Часть I** содержит краткий обзор командной разработки ПО при помощи Team Foundation Server. Здесь рассматривается процесс в целом — с точки зрения среды разработки ПО, включая как собственно разработку, так и тестирование. Здесь же описана общая архитектура Team Foundation Server.
- В **Части II** рассказывается, как структурировать исходный код и управлять зависимостями, а также как определить стратегию ветвления и слияния, если разработка части кода должна производиться изолированно.
- В **Части III** описано, как организовать сборку, как создавать непрерывные сборки, как передавать готовые сборки команде тестировщиков. Обсуждаются типичные проблемы и способы их решения.
- Из **Части IV** вы узнаете, с какой спецификой сталкиваются участники крупных проектов.
- В **Части V** показано, как использовать рабочие элементы, области и итерации Team Foundation Server, чтобы организовать процесс разработки независимо от принятого подхода к управлению проектами.
- **Часть VI** посвящена использованию шаблонов процесса из комплекта Team Foundation Server. Рассказывается о настройке шаблонов, а также о редактировании рабочих элементов в соответствии с процессом разработки, принятым в вашей команде.
- В **Части VII** описано, как хранилища данных компонентов Team Foundation Server объединяются в общий механизм подготовки отчетов. Вы узнаете, как пользоваться отчетами по умолчанию и создавать собственные отчеты.

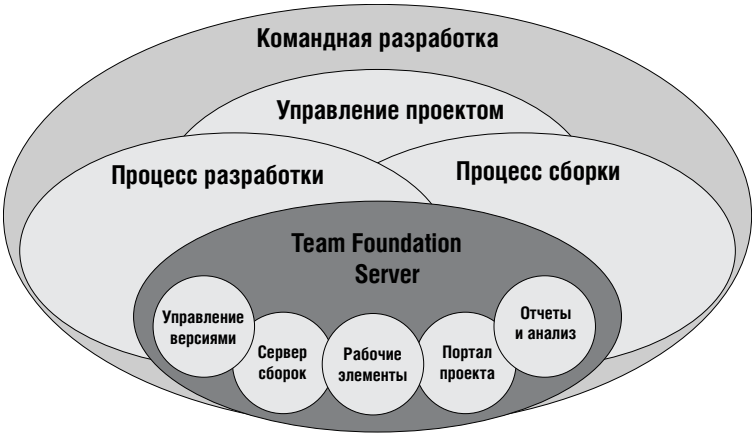
- Благодаря **Части VIII** для вас перестанет быть тайной развертывание Team Foundation Server. Вы узнаете, как выбрать между односерверной и раздельной установкой, а также как поддерживать удаленные группы разработчиков и повысить производительность Team Foundation Server.
- В **Части IX** рассказано об изменениях, внесенных в следующую версию Team Foundation Server. Вы узнаете, какие запланированы новые возможности, а также об усовершенствованиях в имеющихся функциях. Некоторые из этих изменений затронут и приведенные в книге рекомендации, поэтому обязательно познакомьтесь с этой частью, планируя модернизацию Team Foundation Server.
- Раздел «**Рекомендации**» поможет вам в работе со сборками, управлении проектами, составлении отчетов и управлении исходным кодом. В каждой рекомендации помимо описания собственно действий говорится также, когда и как нужно следовать данной рекомендации.
- В разделе **«Практические советы»** обобщен опыт, накопленный командами разработчиков в процессе использования Team Foundation Server как в корпорации Майкрософт, так и за ее пределами. Каждый совет относится к конкретной задаче, важной с точки зрения повышения эффективности работы команды с Team Foundation Server.
- Название раздела **«Вопросы и ответы»** говорит само за себя — здесь вы найдете ответы на распространенные вопросы о Team Foundation.
- В разделе **«Инструкции»** приводятся подробные инструкции по выполнению конкретных действий в Team Foundation Server.
- Раздел **«Ресурсы»** содержит адреса веб-сайтов, поставщиков услуг, форумов и блогов, из которых можно узнать много нового о Team Foundation Server, всегда оставаясь на переднем крае этой технологии.

Разработка в команде

Залогом успешной командной разработки ПО является сочетание многих элементов, процессов и ролей. Это справочник посвящен процессам:

- разработки;
- сборки;
- управления проектом.

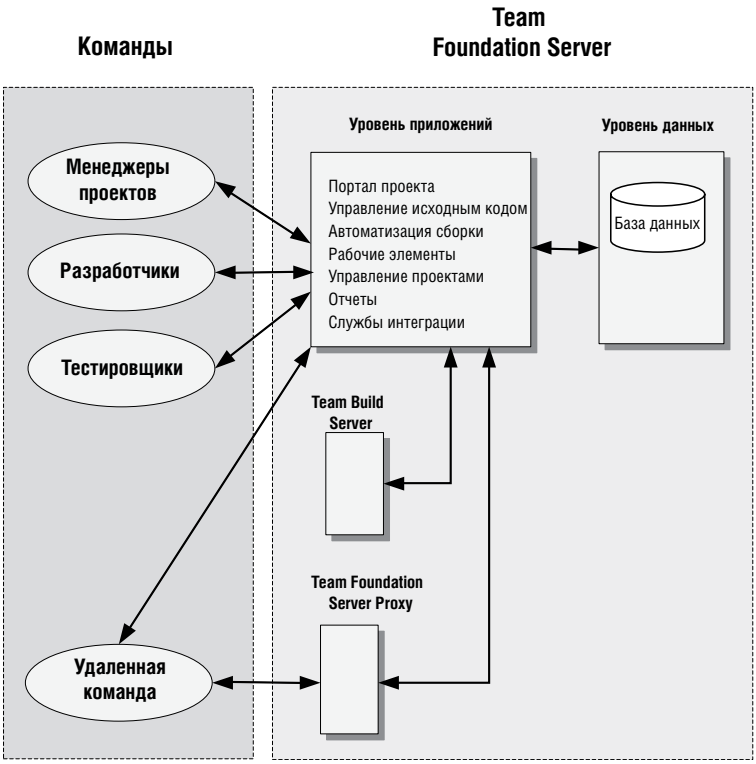
Ниже приведена схема, иллюстрирующая взаимосвязи между типичными процессами разработки ПО в команде и использование Team Foundation Server для их поддержки.



Содержание справочника

Справочник посвящен развертыванию Team Foundation Server и его эффективному использованию для управления исходным кодом, автоматизации сборок, управления рабочими элементами и процессами.

На приведенной ниже схеме показан пример логической реализации Team Foundation Server и его связь с наиболее типичными ролями в разработке ПО.



Почему мы решили написать этот справочник

Из собственного опыта работы с Team Foundation Server и из общения с другими сотрудниками Майкрософт, работающими в этой же области, мы выяснили, что имеется насущная потребность в справочнике, в котором рассказывалось бы о практическом применении Team Foundation. Конечно, некоторая информация содержится в документации продукта, что-то можно найти в блогах и на форумах, но до сих пор не было единого источника, знакомящего с проверенными методами эффективного использования Team Foundation Server для разработки ПО в реальных условиях.

Кому адресован справочник

Цель создания этого справочника — предоставить ресурсы, рекомендации и советы по созданию эффективной среды для командной разработки ПО всем, кто так или иначе вовлечен в эту деятельность. Вот лишь несколько примеров, кому будет интересна приведенная в книге информация:

- команда разработчиков, собирающаяся перейти на Team Foundation;
- менеджеры проектов, которые намерены извлечь из Team Foundation максимум пользы для управления проектом разработки ПО, а также для определения состояния проекта и подготовки отчета для заинтересованных лиц;
- разработчики, заинтересованные в знакомстве с Team Foundation, но пока не испытывающие уверенности в том, что эта система сможет эффективно функционировать в их конкретном случае и с учетом их конкретных ограничений;
- сотрудники, перед которыми поставлена задача спланировать развертывание Team Foundation.

Как пользоваться справочником

Справочник разделен на части, примерно соответствующие последовательности, в которой, по нашему мнению большинство команд осваивает Team Foundation. Если вы в настоящий момент переходите на использование Team Foundation, вам, вероятно, стоит прочитать книгу от начала до конца. Если вас интересует какой-то конкретный аспект использования Team Foundation, например, управление исходным кодом или система Team Build, ограничьтесь соответствующим разделом. В основных главах вы познакомитесь с концепциями и руководящими принципами, а статьи из приложений помогут вам глубже разобраться в деталях реализации. Такое разделение позволит вам сначала сосредоточиться на главном, а затем не забыть и о частностях.

Отзывы и поддержка

Мы приложили все усилия, чтобы этот справочник и приложения к нему были максимально точными.

Отзывы о справочнике

Если у вас возникнут замечания по поводу этого справочника, присылайте их по адресу *TFSguide@microsoft.com*.

Нас особенно интересуют ваши отклики по следующим вопросам:

- технические проблемы, непосредственно связанные с нашими рекомендациями;
- полезность и практическая ценность представленной информации.

Техническая поддержка

Техническую поддержку продуктов и технологий Майкрософт, упоминаемых в этом справочнике, предоставляют службы поддержки продуктов Майкрософт (Product Support Services, PSS). Подробную информацию о поддержке продуктов вы найдете на веб-сайте Microsoft Product Support по адресу *http://support.microsoft.com*.

Поддержка сообщества

Группы новостей MSDN: *http://forums.microsoft.com/MSDN/default.aspx?ForumGroupID=5&SiteID=1*.

Форум	Адрес
Team Foundation Server — Общие вопросы	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=22&SiteID=1*
Team Foundation Server — Установка	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=68&SiteID=1*
Team Foundation Server — Администрирование	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=477&SiteID=1*
Team Foundation Server — Автоматизация сборки	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=481&SiteID=1*
Team Foundation Server — Инструменты и надстройки	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1*
Team Foundation Server — Шаблоны процессов	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=482&SiteID=1*
Team Foundation Server — Отчеты и хранилище	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=480&SiteID=1*
Team Foundation Server — Веб-доступ к Team System	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=1466&SiteID=1*
Team Foundation Server — Управление версиями	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=478&SiteID=1*
Team Foundation Server — Отслеживание рабочих элементов	*http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=479&SiteID=1*

Кто написал этот справочник

Справочник был создан совместными усилиями команды:

- Дж. Д. Мейер (J.D. Meier)
- Джейсон Тейлор (Jason Taylor)
- Алекс Мэкмен (Alex Mackman)
- Прашант Бансоде (Prashant Bansode)
- Кевин Джонс (Kevin Jones)

Авторы и обозреватели

Сторонние авторы и обозреватели.
Вон Хьюз (Vaughn Hughes), Денис Ри (Dennis Rea), Дэвид Ромиг-старший (David Romig, Sr), Евгений Захареев (Eugene Zakhareyev), Кванг Тран (Quang Tran), Леон Ланглейбен (Leon Langleyben), Майк Фури (Mike Fourie), Майкл Раммир (Michael Rummier), Мартин Вудворд (Martin Woodward), Мигель Мендоса (Miguel Mendoza), Сарит Тамир (Sarit Tamir), Тушар Мо (Tushar More).
Авторы и обозреватели Майкрософт.
Аджай Судан (Ajay Sudan), Аджой Кришнамурти (Ajoy Krishnamoorthy), Алан Ридлхувер (Alan Ridlehoover), Алик Левин (Alik Levin), Амейа Бхатавдекар (Ameya Bhatawdekar), Арон Холберг (Aaron Hallberg), Ахмед Салиджи (Ahmed Salijee), Бак Ходжес (Buck Hodges), Барт Харрис (Burt Harris), Биджан Джавиди (Bijan Javidi), Билл Эссари (Bill Essary), Брайан Келлер (Brian Keller), Брайан Мур (Brian Moor), Брайан Харри (Brian Harry), Бретт Кеоун (Brett Keown), Венки Веерарагаван (Venky Veeraraghavan), Грегг Боер (Gregg Boer), Грэнвил Миллер (Granville Miller), Грэхем Берри (Graham Barry), Даг Нойманн (Doug Neumann), Джанет Вильямс Хэплер (Janet Williams Hepler), Джефф Билер (Jeff Beehler), Джули Макаллер (Julie MacAller), Дэвид Кофилд (David Caufield), Дэвид Лемферс (David Lemphers), Кен Перилмен (Ken Perilman), Конор Моррисон (Conor Morrison), Лени Фенстер (Lenny Fenster), Майкл Пулейо (Michael Puleio), Марио Родригес (Mario Rodriguez), Марк Куперштайн (Marc Kuperstein), Мэттью Митрик (Matthew Mitrik), Нобуюки Акама (Nobuyuki Akama), Пит Купленд (Pete Coupland), Питер Провост (Peter Provost), Пол Горинг (Paul Goring), Рили Такета (Ryley Taketa), Роб Кэрон (Rob Caron), Роберт Хорвик (Robert Horvick), Рохит Шарма (Rohit Sharma), Саджи Мэтью (Sajee Mathew), Сиддхарт Бхатья (Siddharth Bhatia), Том Марш (Tom Marsh), Том Холландер (Tom Hollander), Хозе Парра (Jose Parra), Эдвард Джезирский (Edward Jezierski), Эрик Бланше (Eric Blanchet), Эрик Шарран (Eric Charran).

Расскажите нам о себе

Нам интересно узнать, был ли этот справочник полезен. Напишите нам короткое сообщение о проблеме, с которой вы столкнулись, и о том, как наш справочник помог ее решить. Отправьте сообщение по адресу *MyStory@Microsoft.com*.

ЧАСТЬ I

Основные положения

В этой части

- Знакомство с Team Environment.
- АрхитектураTeam Foundation Server.

ГЛАВА 1

Знакомство с Team Environment

В этой главе

- Поддержка цикла разработки программного обеспечения в Microsoft® Visual Studio® Team Foundation Server.
- Использование Team Foundation Server в типичной группе разработчиков.
- Использование Team Foundation Server в типичной группе тестировщиков.
- Описание физической среды группы разработчиков и тестировщиков.

Обзор

В этой главе рассказывается, как использовать Team Foundation Server (TFS) и Microsoft Visual Studio Team System (VSTS) в среде командной разработки ПО. Здесь рассматриваются основные возможности TFS и VSTS, а также автоматизация документооборота между группами разработчиков и специалистов по тестированию. В TFS объединены контроль качества исходного кода, наблюдение за ходом выполнения работы, составление отчетов, управление проектом и автоматизация процесса сборки. Это позволяет команде разработчиков работать более эффективно.

Успешный проект командной разработки ПО состоит из множества процессов, слаженная работа которых необходима для создания эффективной рабочей среды. К основным процессам относятся:

- разработка;
- тестирование;
- сборка;
- развертывание;
- выпуск.

В этой главе вы познакомитесь с типичными приемами, которыми пользуются команды разработчиков и специалистов по тестированию при работе с TFS. Здесь описано, как использовать TFS для управления рабочим процессом и обеспечения продуктивного сотрудничества между группами.

Более подробную информацию об архитектуре и основных компонентах TFS вы найдете в главе 2.

Логическая организация работы в Team Foundation Server

Продукт TFS позволяет команде разработчиков хранить исходный код в централизованном хранилище. Извлекая код из хранилища, при помощи сервера сборки вы создаете сборки (build), а затем передаете их группе испытателей.

На рис. 1-1 показано, как в TFS организован рабочий процесс и как связаны между собой среда разработчиков и среда испытателей.

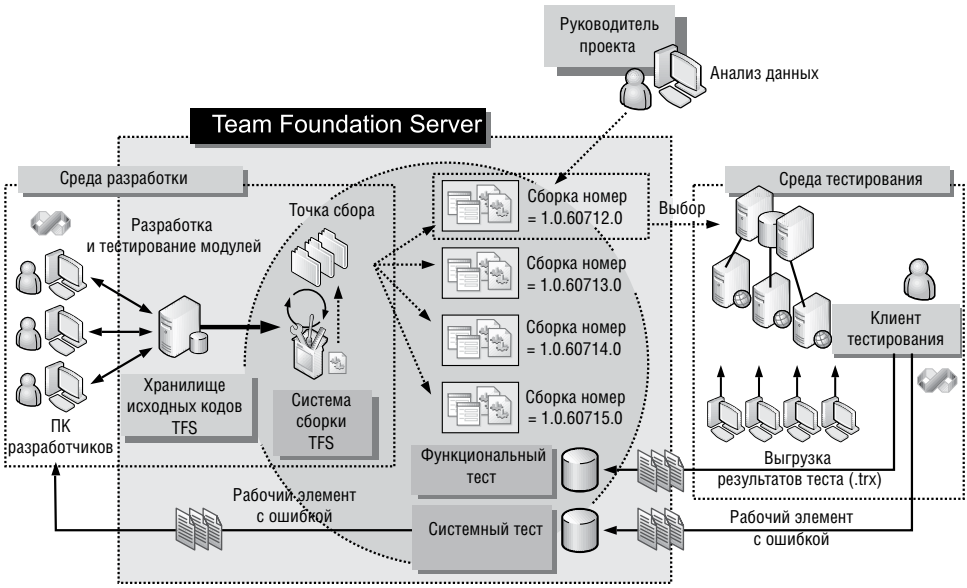


Рис. 1-1. *Логический документооборот Team Foundation Server*

Тестовая группа запускает результаты сборки в тестовой среде, проводя сочетание ручных и автоматических тестов. Результаты тестов хранятся в TFS и используются для организации обратной связи по вопросам качества сборки. Кроме того, тестовая группа может создавать рабочие элементы и ошибки (особый тип рабочего элемента), на которые группе разработчиков следует обратить внимание. Эти элементы позволяют группе тестирования отслеживать работу группы разработчиков.

Логическая организация работы в средах разработки, тестирования и производства

В крупных организациях, где имеется несколько групп разработчиков, у каждой из них есть собственный TFS с отдельными хранилищами и серверами сборки. На рис. 1-2 показан пример логического потока операций для двух групп разработчиков, передающих сборки в объединенную группу испытаний.

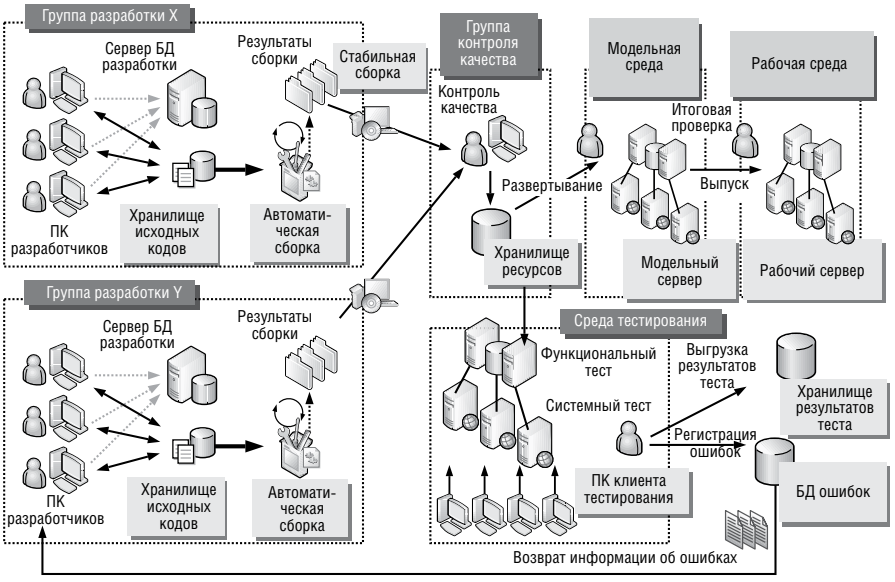


Рис. 1-2. *Логическая организация работ в двух группах разработчиков и одной группе комплексного тестирования*

Каждая группа разработки по расписанию передает сборки в точку сбора, например, в общий сетевой ресурс. Группа тестирования извлекает сборки оттуда и проводит их испытания, определяя качество сборок. Когда контроль качества пройден, приложение развертывается на модельном сервере для итогового тестирования и проверки пользователями. После этого приложение развертывается на рабочем сервере.

Процесс разработки

Работая над проектом ПО, разработчики вовлечены в несколько ключевых взаимодействий с TFS. Например, разработчик взаимодействует с TFS следующими способами:

- Осуществляет доступ к ошибкам и задачам TFS, чтобы выяснить, какую работу ему нужно сделать. Рабочие элементы могут назначаться разработчику менеджером проекта, другим разработчиком или испытательной группой.
- Использует обозреватель исходного кода (Source Control Explorer) VSTS для получения доступа к хранилищу исходных кодов TFS и загружает

последнюю версию исходного кода в локальную рабочую область или на свой компьютер.

- Выполнив назначенную задачу, снова помещает код в БД управления исходным кодом.

Размещение кода может запустить процесс непрерывной сборки при помощи Team Build. Если сборка завершилась неудачей, создается новый рабочий элемент для отслеживания ошибки.

Процесс тестирования

Тестировщик взаимодействует с TFS следующими способами:

- Извлекает результат плановой сборки из точки сбора.
- При помощи различных инструментов VSTS выполняет ручное и автоматическое тестирование, включая проверку безопасности, производительности и работы в веб.
- Выгружает результаты испытаний в БД тестирования для последующего использования.
- Регистрирует ошибки, выявленные в ходе тестирования, как новые рабочие элементы TFS.
- Разрешает существующие ошибки, если они устранены в последней сборке.

Физические среды разработки и тестирования

Количество компьютеров в средах разработки и тестирования зависит от размера групп и объема проектов. На рис. 1-3 изображена типичная физическая среда разработки и тестирования.

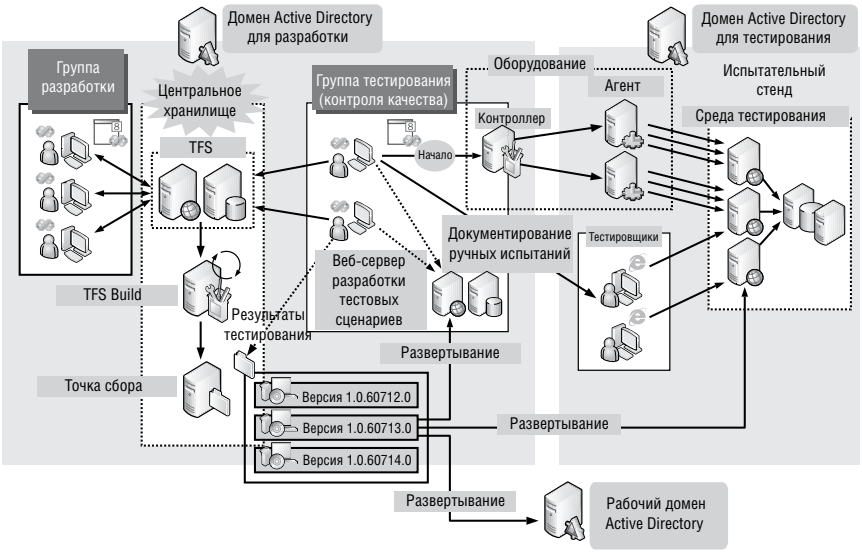


Рис. 1-3. *Физическая среда разработки и тестирования*

Среда разработки

Среда разработки служит для поддержки процессов разработки и сборки и содержит следующие компьютеры:

- сервер Team Foundation Server;
- сервер сборок;
- сервер для сбора результатов работы сервера сборок;
- рабочие станции разработчиков.

Если группа разработчиков осуществляет удаленный доступ к TFS или размер ее столь велик, что отрицательно сказывается на производительности центрального сервера TFS, для повышения эффективности работы вы можете также настроить TFS-прокси.

Среда тестирования

Среда тестирования состоит из одной или нескольких рабочих станций, на которых установлен продукт Visual Studio Team Edition for Software Testers. Он используется для управления циклом тестирования, а также для выполнения функционального тестирования, системного тестирования, тестирования производительности и веб-тестирования. Члены группы используют TFS для управления рабочими элементами, ошибками и результатами тестов.

В среду тестирования также может включаться продукт Visual Studio Team Test Load для проверки производительности.

Резюме

Продукты VSTS и TFS предназначены для поддержки цикла разработки ПО и объединяют различные его аспекты, например, контроль качества исходного кода, наблюдение за ходом выполнения работы, составление отчетов, управление проектом и автоматизацию сборки.

TFS играет определяющую роль в обеспечении сотрудничества между группами тестировщиков и разработчиков. Группа разработчиков взаимодействует с TFS на протяжении цикла разработки, осуществляет доступ к системе контроля за исходным кодом, а также к ошибкам и рабочим элементам, чтобы выяснить, какие от нее требуются действия. Группа тестирования взаимодействует с TFS для запуска тестов, выгрузки результатов тестов и регистрации ошибок.

Дополнительные ресурсы

- Дополнительные сведения об основах TFS вы найдете в статье «Team Foundation Server Fundamentals: a Look at the Capabilities and Architecture» по адресу *http://msdn2.microsoft.com/en-us/library/ms364062(vs.80).aspx*.
- Обзор продукта Team Foundation содержится в документации к Team Foundation на веб-сайте Microsoft MSDN® по адресу *http://msdn2.microsoft.com/en-us/library/ms181232(vs.80).aspx*.

Глава 2

Архитектура Team Foundation Server

В этой главе

- Архитектура продуктов Microsoft® Visual Studio® Team System (VSTS) и Team Foundation Server (TFS).
- Компоненты, составляющие уровни клиента, приложений и данных.
- Различия между односерверной и многосерверной разработкой.

Обзор

В этой главе вы познакомитесь с архитектурой TFS и основными топологиями разработки. Логически TFS обладает трехуровневой архитектурой, состоящей из клиентского уровня (client tier), уровня приложений (application tier) и уровня данных (data tier). Клиенты TFS взаимодействуют с уровнем приложений посредством различных веб-служб. Уровень приложений пользуется различными базами данных Microsoft SQL Server™ на уровне данных.

Вы вольны установить уровень приложений и уровень данных как на одном физическом сервере, так и на разных серверах. В основном, выбор определяется размером группы. Односерверное развертывание более уместно в группах, насчитывающих менее 50 членов, хотя достаточно мощный сервер способен поддерживать и до 400 пользователей. Многосерверное развертывание позволяет увеличить количество пользователей до 2000.

Из этой главы вы узнаете об основных компонентах TFS и об их взаимодействии друг с другом, а также о назначении каждого из этих компонентов и о том, каким образом обычно производится их развертывание.

Если вы не знакомы с TFS, сначала прочтите главу 1. В ней описано, как группы разработчиков и тестировщиков взаимодействуют с TFS, чтобы упростить сотрудничество и повысить эффективность разработки ПО.

Архитектура Team Foundation Server

В TFS использована логическая трехуровневая архитектура, разделяющаяся на *клиентский* уровень, а также уровни *приложений* и *данных*. Клиенты TFS взаимодействуют с уровнем приложений посредством различных веб-служб. В свою очередь, уровень приложений поддерживается различными базами данных на уровне данных. Компоненты уровня TFS и их взаимодействие проиллюстрированы на рис. 2-1.

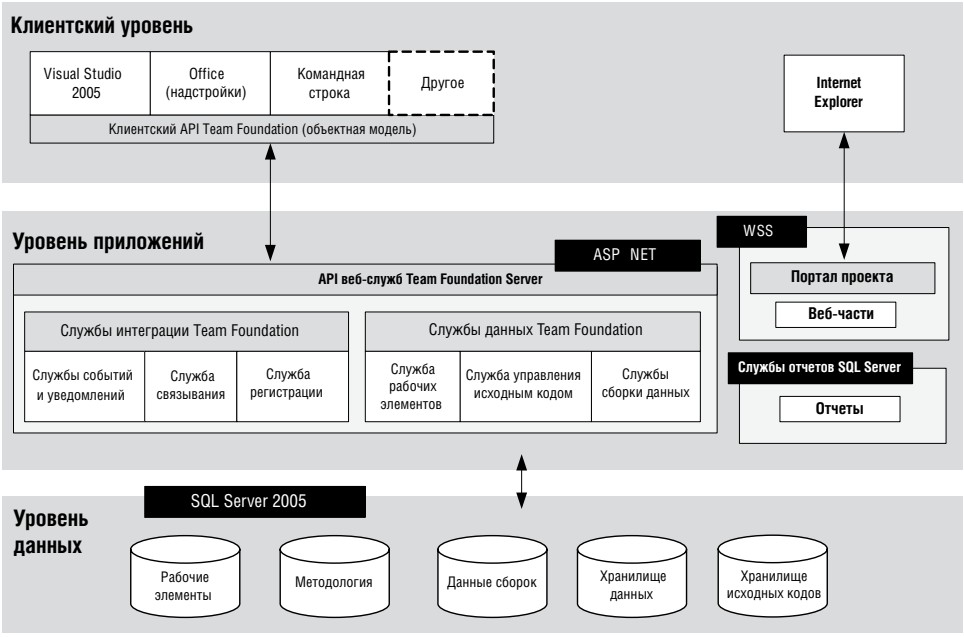


Рис. 2-1. *Компоненты и уровни TFS*

Клиентский уровень

Клиентский уровень состоит из следующих компонентов:

- **Объектная модель Team Foundation Server** Открытый интерфейс API для взаимодействия с TFS, используется для создания клиентских приложений, обменивающихся данными с TFS.
- **Компоненты Visual Studio Industry Partners (VSIP)** Инструменты сторонних поставщиков, надстройки и языки для использования в Visual Studio.
- **Интеграция с Microsoft Office** Набор надстроек для Microsoft Office Excel® и Microsoft Office Project, позволяющих запрашивать и обновлять рабочие элементы в базе данных TFS Work Item Tracking. Особенно полезен для менеджеров проекта, уже широко использующих эти инструменты.

- **Инструменты командной строки** Инструменты, позволяющие взаимодействовать с TFS из командной строки. В основном используются для работы с функциями контроля качества исходного кода, также полезны для автоматизации повторяющихся процессов и при планировании заданий.
- **Политики возврата после правки (check-in policy)** Расширяемый механизм проверки кода в процессе возврата после правки.

Уровень приложений

На уровне приложений клиентскому уровню предоставляется доступ к веб-службам ASP.NET. Использование этих веб-служб в интеграторах от независимых разработчиков не предполагается; здесь они приведены только для полноты картины. Службы сгруппированы в следующие коллекции:

- Team Foundation Data Services;
- Team Foundation Integration Services.

Службы данных Team Foundation

Эти веб-службы отвечают, в первую очередь, за операции с данными на уровне данных. К ним относятся следующие веб-службы:

- **Version Control** Используется на клиентском уровне для выполнения различных функций TFS по контролю качества исходного кода и для взаимодействия с БД контроля качества исходного кода.
- **Work Item Tracking** Используется на клиентском уровне для создания, обновления и направления запросов к рабочим элементам в БД Work Item Tracking.
- **Team Foundation Build** Используется на клиентском уровне и в оболочке MSBuild для выполнения процесса сборки.

Службы интеграции Team Foundation

Эти веб-службы обеспечивают функциональные возможности интеграции и автоматизации и не взаимодействуют с уровнем данных. К ним относятся следующие веб-службы:

- **Registration** Используется для регистрации других служб TFS, сохраняет информацию в регистрационной БД. Информация используется службами для обнаружения друг друга и определения способа взаимодействия.
- **Security** Состоит из службы групповой безопасности (group security service) и службы проверки подлинности (authorization service). Служба групповой безопасности управляет всеми пользователями и группами TFS. Служба проверки подлинности обеспечивает авторизацию доступа к TFS.
- **Linking** Содержит инструменты для создания слабых связей — «ссылок» (link) — между элементами данных. Например, связь между рабочим элементом дефекта и исходным кодом, измененным для устранения

дефекта, устанавливается при помощи ссылки TFS.

- **Eventing** Запускает инструмент или службу для регистрации типов событий. Пользователь может подписаться на события и получать уведомление по электронной почте или с помощью вызова веб-службы. Например, можно использовать событие возврата после правки для запуска непрерывной сборки.
- **Classification** Работает вместе с веб-службой Linking и позволяет классифицировать артефакты TFS в соответствии с предопределенными таксономиями. Это облегчает поддержку объединенных отчетов даже для артефактов, которые не пользуются общей таксономией для упорядочивания своих данных. Например, если рабочие элементы упорядочены по группам, а тесты упорядочены по компонентам, вы также можете упорядочить тесты по группам, что позволит им фигурировать в отчете рядом с рабочими элементами.

Уровень данных

Прямой доступ из клиентских приложений к данным, хранящимся на уровне данных, в TFS не поддерживается. Все запросы к данным должны осуществляться через веб-службы на уровне приложений. Уровень данных TFS состоит из следующих хранилищ, соответствующих службам данных на уровне приложений.

- **Отслеживание рабочих элементов** В этом хранилище хранятся все данные, относящиеся к рабочим элементам.
- **Управление версиями** Здесь хранятся все данные, относящиеся к управлению исходным кодом.
- **Team Foundation Build** Здесь хранится вся информация, относящаяся к TFS Team Build.
- **Хранилище отчетов** Здесь хранится информация, относящаяся ко всем инструментам и функциям TFS. Хранилище отчетов облегчает создание отчетов, объединяющих данные от различных инструментов.

Топология развертывания

Развертывание TFS выполняется с использованием различных топологий — от односерверных до сложных многосерверных типологий.

Основные требования

Независимо от используемой топологии вам следует помнить о нескольких ключевых требованиях.

- Устанавливайте уровень приложений и уровень данных в одном и том же домене. При этом они могут находиться как на одном, так и на разных серверных узлах.
- TFS устанавливается на компьютеры под управлением Microsoft Windows Server™ 2003 SP1 или более поздней версии.

- Все веб-службы уровня приложений должны устанавливаться на одном сервере.
- Устанавливайте один экземпляр TFS на одном компьютере.
- Нельзя установить более одного экземпляра TFS на физический сервер.
- Не распределяйте БД TFS между несколькими серверами БД. Все проекты должны находиться в одной группе серверов Team Foundation и не могут быть распределены по группам.
- Для размещения портала проекта нельзя использовать существующую инфраструктуру Microsoft SharePoint® Portal Server. Рассмотрите возможность использования специализированного сервера для размещения на нем порталов TFS SharePoint.
- Не пытайтесь установить TFS на контроллер домена — это не поддерживается.
- При развертывании на двух серверах подготовьте учетные записи домена для служб TFS. Например, вам нужно будет создать учетные записи DOMAIN\TFSSERVICE и DOMAIN\TFSREPORTS.

Развертывание на одном сервере

Развертывание на одном сервере — простейшая топология, подходящая для групп разработчиков или пилотных проектов с числом пользователей не более 400. В этом подходе все компоненты уровня приложений и уровня данных устанавливаются на один сервер и доступ к ним осуществляется в пределах одного домена.

Если вам нужно установить компоненты испытательного стенда для проверки производительности, установите их на серверном узле или на одном или нескольких клиентах. На рис. 2-2 показана топология с одним сервером.

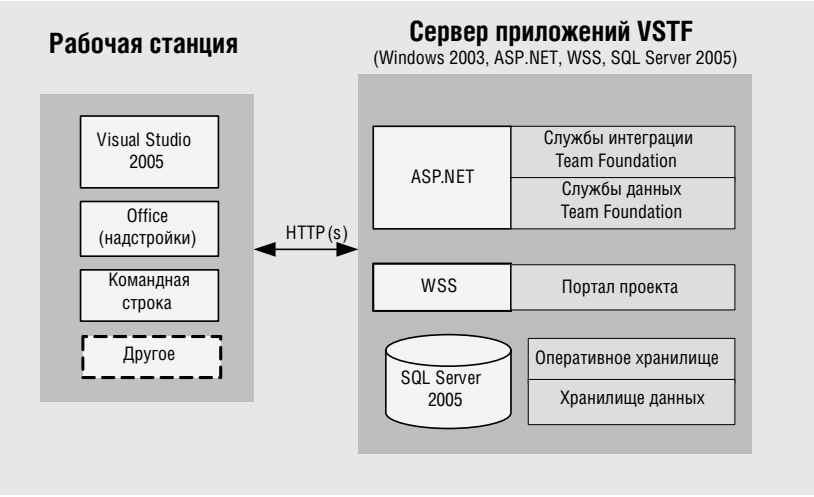


Рис. 2-2. *Односерверная топология*

Развертывание на раздельных серверах

Топология многосерверного развертывания используется в крупных группах разработчиков, насчитывающих до 2000 пользователей. В данной топологии уровень приложений устанавливается отдельно от уровня данных.

Вы можете установить службы Team Foundation Build Services на уровень приложений, однако в крупных группах рекомендуется выделять для сборки один или несколько специальных серверов. Если в проекте требуется проверка производительности, разверните испытательный стенд (контроллер и агенты) на дополнительных серверных узлах. На рис. 2-3 показана многосерверная топология.

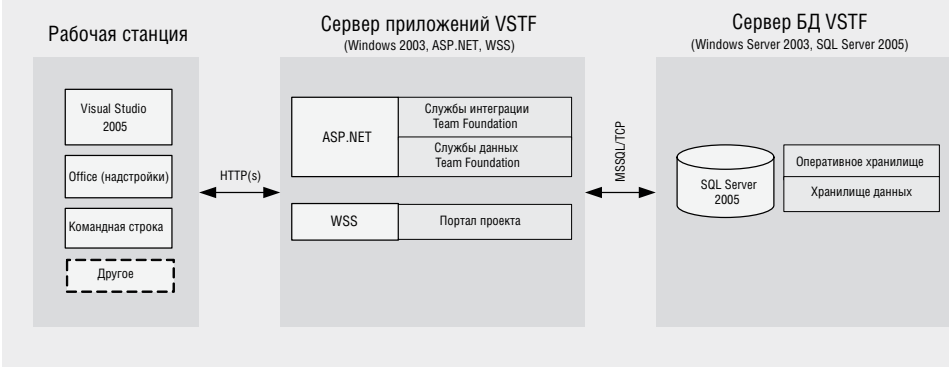


Рис. 2-3. *Многосерверная топология*

Резюме

Архитектура Team Foundation Server разделена на три уровня: клиентский, приложений и данных.

- На клиентском уровне находятся компоненты клиента, например, Team Explorer в Visual Studio 2005, интеграция с Microsoft Office и инструменты командной строки.
- На уровне приложений содержатся, например, службы управления версиями Team Foundation, службы отслеживания рабочих элементов и службы сборки.
- На уровне данных содержатся БД для хранения данных, необходимых для отслеживания рабочих элементов, управления версиями, групповой сборки и организации хранилища отчетов.

В продукте TFS поддерживаются односерверная и многосерверная топологии развертывания. В первом случае уровень приложений и уровень данных устанавливаются на одной машине. Односерверное развертывание оправдано для небольших групп или при выполнении пилотных проектов. При раздельном развертывании уровень приложений и уровень данных устанавливаются на отдельных серверах. Многосерверное развертывание полезно в крупных группах с большим числом пользователей.

Дополнительные ресурсы

- Дополнительные сведения об основах TFS вы найдете в статье «Team Foundation Server Fundamentals: a Look at the Capabilities and Architecture» по адресу *http://msdn2.microsoft.com/en-us/library/ms364062(vs.80).aspx*.
- Обзор продукта Team Foundation содержится в документации к Team Foundation на веб-сайте Microsoft MSDN® по адресу *http://msdn2.microsoft.com/en-us/library/ms181232(vs.80).aspx*.
- Дополнительную информацию о масштабируемости Team Foundation Server вы найдете в статье «Team Foundation Server Capacity Planning» по адресу *http://blogs.msdn.com/bharry/archive/2006/01/04/509314.aspx*.

ЧАСТЬ II

Управление исходным кодом

В этой части

- Структуризация проектов и решений для управления исходным кодом.
- Структуризация проектов и решений для управления исходным кодом в Team Foundation.
- Стратегии ветвления и слияния.
- Управление зависимостями в системе управления исходным кодом Visual Studio Team System.

ГЛАВА 3

Структуризация проектов и решений для управления исходным кодом

В этой главе

- Структурирование решений и проектов Microsoft® Visual Studio® Team System.
- Выбор между использованием одного решения или несколько решений.
- Определение оптимальной структуры для небольших, средних и крупных команд.

Обзор

В этой главе рассматриваются различные возможности структурирования файлов решений и проектов Visual Studio для командной разработки. Файлы решений (.sln) используются в Visual Studio для группирования файлов взаимосвязанных проектов Visual Studio (.csproj и .vbproj). От выбора модели структурирования проектов и решений зависит многое, например, насколько легко члены групп разработчиков смогут записывать решения и проекты в базу управления исходным кодом и извлекать их из нее.

Если вы работаете над небольшим проектом, для размещения всех файлов вам достаточно будет единственного решения. Если вы разрабатываете ПО с большим количеством проектов, воспользуйтесь несколькими решениями для группирования взаимосвязанных проектов, соответствующих различным аспектам функциональности общего проекта. В некоторых сценариях в дополнение к этим решениям может потребоваться и единый глобальный файл решения для объединения всех файлов проекта.

Стратегии структурирования решения и проекта

Для структурирования файлов решений и проектов чаще всего используются три стратегии:

- **Одиночное решение** Работая над небольшой системой, поместите все проекты в одиночное решение.
- **Решение с разделами** Работая над крупной системой, используйте для группирования связанных проектов несколько решений. Они должны логически объединять подмножества проектов, которые разработчик, скорее всего, будет изменять одновременно. Кроме того, создайте одно глобальное решение, содержащее все проекты. Этот метод позволяет сократить объем данных, извлекаемых из системы управления исходным кодом при работе над отдельным проектом.
- **Несколько решений** Если вы работаете над очень большой системой, требующей десятков и более проектов, используйте для работы над подсистемами несколько самостоятельных решений. Глобальное решение, содержащее все проекты, не создавайте.

В целом, следует придерживаться следующих рекомендаций:

- Используйте стратегию одиночного решения, даже если в результате получается решение слишком большое для загрузки его в Visual Studio.
- Используйте несколько решений, чтобы получить возможность отдельно рассматривать подсистемы вашего приложения.
- Разделяйте проект на несколько решений, чтобы сократить время, требующееся разработчикам на загрузку решения и его сборку.

Разрабатывая структуру проекта и решения, учитывайте следующие моменты:

- Каждый проект во время сборки генерирует файл сборки (assembly). Для начала определите, какие файлы сборки вы хотите создать, а затем на основании этой информации решите, какие проекты вам нужны. Это поможет вам распределить код по проектам.
- Начните с простейшего варианта — одиночного решения. Усложняйте структуру, только если это действительно необходимо.
- Проектируя структуру с несколькими решениями, сделайте следующее:
  - Учитывайте зависимости проектов. Старайтесь объединить в одном решении проекты, имеющие зависимости друг от друга. Это позволит использовать ссылки проекта в рамках решения. Использование ссылок проекта вместо файловых ссылок позволяет синхронизировать параметры конфигураций (Debug/Release) Visual Studio, а также отслеживать версии для определения момента очередной сборки проекта. Попытайтесь сократить количество ссылок проекта, указывающих в другие решения.

- Продумайте общее использование ресурсов. Размещайте в одном решении проекты с общим кодом.
- Учитывайте структуру групп. Компонуйте решения так, чтобы облегчить работу группам, занятым разработкой связанных проектов.

- Сохраняйте «плоскую» структуру проектов, чтобы удобнее было объединять их в решения, не меняя структуру файлов или папок системы управления исходным кодом.

Одиночное решение

Если вы работаете над небольшой системой, Вам, вероятно, стоит разместить все проекты в одном решении Visual Studio. Эта структура упрощает разработку, поскольку при открытии решения сразу становится доступным весь код. При такой стратегии легко устанавливать ссылки, поскольку все ссылки связывают проекты, находящиеся в одном решении. Возможно, при этом вам все равно потребуются файловые ссылки для указания на файлы сборки сторонних производителей, например, приобретенные компоненты, находящиеся за пределами решения. Метод одиночного решения проиллюстрирован на рис. 3-1.

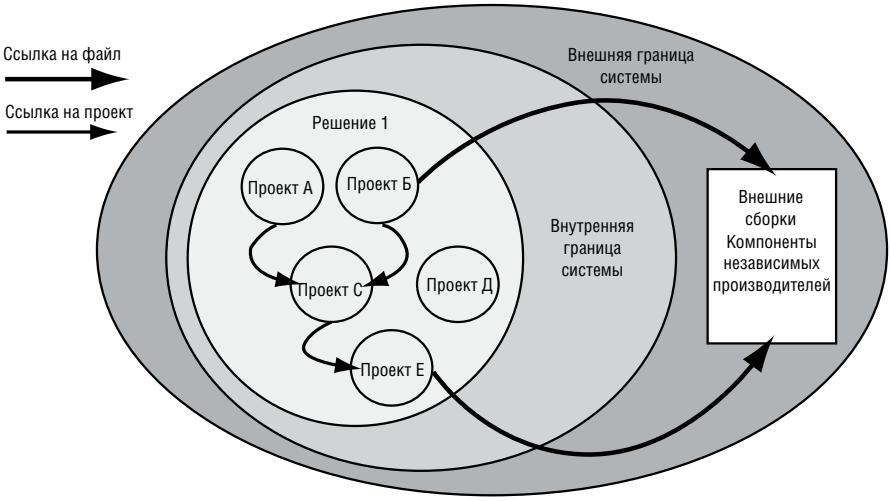


Рис. 3-1. *Метод одиночного решения*

Вот основные доводы к использованию этой структуры:

- Сценарии сборки просты.
- Легко составить карту зависимостей между проектами в пределах решения.

Такую структуру следует использовать, если все разработчики пользуются одним решением и имеют один и тот же набор проектов. Это не всегда возможно при разработке больших систем, где требуется упорядочивать проекты по подсистеме или функции.

Решение с разделами

Если вы работаете над крупной системой, подумайте об использовании нескольких решений, каждое из которых представляло бы подсистему вашего приложения. Такие решения позволяют разработчикам работать над небольшими частями системы, не загружая весь код всех проектов. Составляйте структуру решений таким образом, чтобы проекты, имеющие зависимости, группировались вместе. Это позволит использовать ссылки на проекты, а не ссылки на файлы. Возможно, стоит также создать файл основного решения, в котором содержались бы все ваши проекты. Это глобальное решение может применяться для сборки всего приложения.

Примечание В отличие от предыдущих версий, система Visual Studio 2005 основана на MSBuild. Это позволяет создавать структуры решений, не включая в них все проекты, на которые имеются ссылки и тем не менее производить сборку без ошибок. При условии, что сначала выполняется сборка основного решения, при которой происходит генерация двоичных файлов для каждого проекта, система MSBuild способна отследить ссылки проекта, выходящие за рамки решения, и успешно выполнить сборку. Это сработает только в том случае, если вы используете ссылки на проекты, а не на файлы. Созданные таким образом решения можно успешно собирать из командной строки сборки Visual Studio или из интегрированной среды разработки, но по умолчанию не в Team Build. Чтобы провести успешную сборку в Team Build, используйте основное решение, включающее в себя все проекты и зависимости.

При работе с несколькими решениями используйте плоскую файловую структуру во всех ваших проектах. Типичный пример — приложение, в котором имеется проект Microsoft Windows® Forms, проект ASP.NET, служба Windows и набор проектов для библиотек классов, которые используются некоторыми или всеми вышеперечисленными проектами.

Вы можете воспользоваться следующей плоской структурой применительно ко всем проектам:

- /Source
- /WinFormsProject
- /WebProject
- /WindowsServiceProject
- /ClassLibrary1
- /ClassLibrary2
- /ClassLibrary3
- Web.sln
- Service.sln
- All.sln

На рис. 3-2 проиллюстрирован метод решения с разделами.

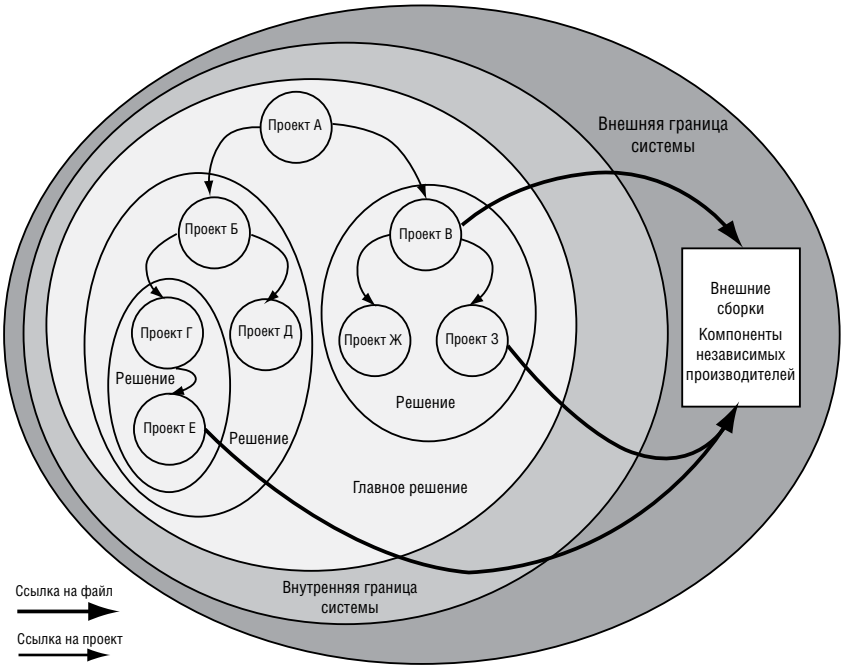


Рис. 3-2. *Метод решения с разделами*

Простота структуры обеспечивает гибкость и позволяет использовать решения для рассмотрения проектов с различных сторон. Физическую структуру папок решения очень трудно изменить, особенно если вы осознали, что вам необходимо использовать библиотеку классов из другого решения.

Причины, по которым следует использовать эту структуру, таковы:

- При загрузке и сборке частичных решений для приложения повышается производительность.
- Частичные решения могут применяться для создания представлений наборов проектов в зависимости от структуры подгрупп разработчиков или границ совместного использования кода.
- Главное решение можно использовать для сборки всего приложения.
- Вы легко составите схему зависимостей между проектами любого частичного решения.
- Логическое разделение решений уменьшает общую сложность работы. Например, новым разработчикам будет проще понять решение, разделенное на технологическую и функциональную части.

Есть и довод в пользу того, чтобы не увлекаться использованием данной структуры: увеличение затрат на сопровождение решений. Добавление нового проекта может привести к необходимости изменения многих файлов решения.

Несколько решений

Работая над очень крупным решением, требующим многих десятков проектов, немудрено натолкнуться на ограничения масштабируемости. В этом случае вам уже просто придется разделить приложение на несколько решений, но уже не создавая главное решение для всего приложения, поскольку ссылки внутри решений могут быть только ссылками на проекты. Ссылки, выходящие за пределы решения (например, на библиотеки независимых разработчиков или проекты в другом решении), являются файловыми ссылками. Это означает, что главного решения быть не может.

Вместо этого следует использовать сценарий, в котором указан порядок сборки решений. Одна из центральных задач обслуживания нескольких взаимосвязанных решений состоит в контроле за тем, чтобы разработчики случайно не создали циклические ссылки между решениями. Эта структура требует сложных сценариев сборки и явного отображения отношений зависимости. В Visual Studio полностью собрать подобное приложение уже невозможно. Придется воспользоваться непосредственно TFS Team Build или MSBuild.

Структура с несколькими решениями показана на рис. 3-3.

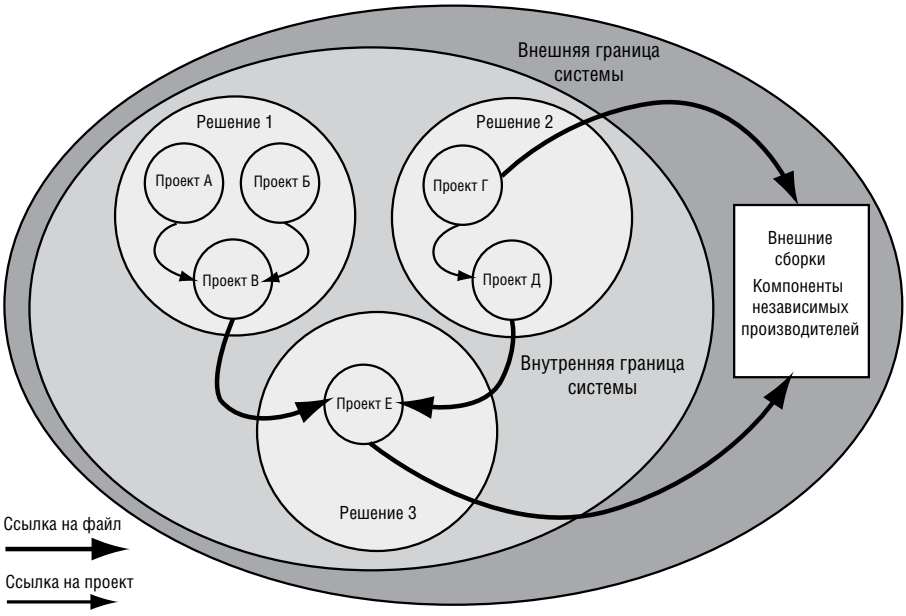


Рис. 3-3. *Структура с несколькими решениями*

Эту структуру следует использовать для очень крупных приложений, чтобы обойти проблемы с производительностью среды разработки Visual Studio и пределами масштабируемости.

Один из недостатков этой структуры — сложные сценарии сборки, способные обработать зависимости частичных решений путем соблюдения нужного порядка сборки решений.

Особенности крупных проектов

Большие группы разработчиков, как правило, отличаются следующими особенностями:

- Им требуется более сложная структура ветвления и слияния.
- Для них более характерно управление зависимостями решений и командных проектов.
- Для них более характерно наличие нескольких сборок компонентов и групп.

В большинстве крупных проектов хорошо работает структура решения с разделами, так как она одновременно обеспечивает гибкость и содержит единое решение, позволяющее собирать приложение целиком. Если ваше приложение настолько велико, что не вписывается в пределы масштабируемости, используйте метод нескольких решений.

Резюме

Используйте единое решение в небольших проектах, где нет необходимости разделять исходный код на отдельные частичные решения.

Используйте разделенные решения, чтобы логически объединить подмножества проектов, которые, скорее всего, будут изменяться разработчиком одновременно. Затем создайте одно глобальное решение, содержащее все проекты.

Используйте несколько решений, чтобы создать отдельные представления для подсистем и уменьшить время загрузки и сборки приложения.

Метод разделенных решений хорошо работает в большинстве крупных проектов, поскольку обеспечивает гибкость и содержит единое решение, позволяющее собирать приложение целиком.

Дополнительные ресурсы

- **Глава 4** В ней приведены важные соображения, которые следует учитывать при организации управления исходным кодом в TFS.
- **Глава 6** Структура проекта определяет доступные стратегии управления зависимостями в проектах и решениях. Управление зависимостями подробно описано в главе 6.
- **Перечисленные ниже разделы из части «Практикум»** В них подробно описаны процедуры, обсуждаемые в этой главе.
  - «Как структурировать приложения ASP.NET в Visual Studio Team Foundation Server».

  - «Как структурировать приложения Windows в Visual Studio Team Foundation Server».
  - «Как структурировать папки управления исходным кодом в Visual Studio Team Foundation Server».
- Дополнительную информацию о структуре проекта и решения (не имеющую прямого отношения к Team Foundation Server) вы найдете в статье «Team Development with Visual Studio .NET and Visual SourceSafe» по адресу *http://msdn2.microsoft.com/en-us/library/ms998208.aspx*.

ГЛАВА 4

Структуризация проектов и решений для управления исходным кодом в Team Foundation

В этой главе

- Повышение эффективности командной разработки в системе управления исходным кодом Microsoft® Visual Studio® Team Foundation Server (TFS) при помощи структурирования проектов.
- Синхронизация структуры папок на серверной и клиентской сторонах.
- Выбор стратегии модульного тестирования структуры.
- Создание структуры папок для различных сценариев ветвления.
- Знакомство с рабочей областью и сопоставлением локальных файлов с системой управления исходным кодом.
- Файлы, добавляемые в систему управления исходным кодом.

Обзор

Многие стандартные соглашения о папках Visual Studio, используемые при создании новых решений и проектов, не оптимальны для командной разработки и использования с системой управления исходным кодом TFS. Создавая новые проекты и решения Visual Studio, не спешите принять умолчания. Продумайте собственную локальную и серверную структуру папок.

В этой главе объясняется, как следует структурировать решения и проекты на компьютере разработчика (на клиентской стороне) и как структурировать папки в системе управления исходным кодом TFS (на серверной стороне).

Приведены примеры структур папок для приложений различных типов, включая Microsoft Windows® Forms, Smart Client и веб-приложения. В главе также рассказывается, как использовать рабочие области для управления сопоставлениями между клиентскими и серверными структурами папок.

Структура на стороне сервера

Большинство командных проектов содержат одно или несколько решений Visual Studio, каждое из которых содержит один или несколько проектов Visual Studio. Если для поддержки конкретных вариантов разработки вам необходимо ветвление, используйте для объединения проектов Visual Studio как на клиенте, так и на сервере корневую папку Main. Далее приведен пример структуры папок в системе управления исходным кодом TFS:

```
{
$MyTeamProject1
 /Main                                →Может содержать файлы решения (.sln)
      /Source
            /MyApp1                   →Файл MyApp1.sln
                  /Source             →Папка для всего исходного кода
                     /ClassLibrary1   →Файл ClassLibrary1.csproj
                     /MyApp1Web       →Файл Default.aspx
                  /UnitTests          →Основная папка для модульных тестов
                     /ClassLibrary1Tests  →Проект и код для тестирования
                     /MyApp1WebTests   →Проект и код для тестирования
         /SharedBinaries              →Общие ресурсы, например, библиотеки
         /SharedSource                →Общий исходный код
      /Docs                           →Документация продукта
      /Tests                          →Общая папка тестирования
         /FunctionalTests
         /PerformanceTests
         /SecurityTests
/TeamBuildTypes                       →Создаются автоматически Team Build
      /BuildType1
      /BuildType2
}
```

Папка **Main** служит для хранения файлов исходного кода и других связанных с ним компонентов, например, результатов сборки, проектной документации и данных тестирования. Папка приложения (**MyApp** в предыдущем примере) содержит файл решения Visual Studio (.sln), используемый для группирования связанных проектов Visual Studio. Каждый файл проекта (.vcproj или .vbproj) содержится в специальной подпапке проекта, расположенной в папке /Main/Source/MyApp1/Source. Модульные тесты, сопровождающие каждый проект, находятся в папке **UnitTests**. Вы можете поместить в папку Main дополнительные файлы решений (.sln) Visual Studio, что-

бы получить возможность работать со другими вариантами группирования проектов.

Папки **Docs** и **Test** используются для размещения дополнительных артефактов, связанных с командным проектом, включая документацию продукта и автоматические тесты.

Папка **TeamBuildTypes** создается автоматически, когда вы впервые генерируете сборку Team Build. Если вы хотите зарегистрировать тип командной сборки вручную, создайте эту папку самостоятельно, добавьте в нее файлы Team Build, и TFS автоматически распознает эту папку.

Подробнее о группировании проектов и структуре решений — в главе 3.

Хранение модульных тестов

Вы можете хранить модульные тесты в папке **UnitTests**, расположенной на одном уровне с папкой **Source**, как показано ниже.

```
{
   …
      /MyApp1                          →Файл MyApp1.sln
            /Source                    →Папка для всего исходного кода
                  /ClassLibrary1             →Файл ClassLibrary1.csproj
                  /MyApp1Web                 →Файл Default.aspx
            /UnitTests                 →Основная папка для модульных тестов
                  /ClassLibrary1Tests  →Проект и код для тестирования
                  /MyApp1WebTests      →Проект и код для тестирования
}
```

Этот вариант предполагает, что модульные тесты играют важную роль. Однако это происходит в ущерб совместимости ветвей на уровне проекта. Ниже приведена альтернативная структура:

```
{
   /MyApp1
      /Source
            /ClassLibrary1
            /ClassLibrary1Tests
            /MyApp1Web
            /MyApp1WebTests
}
```

Ниже приводятся доводы «за» и «против» каждого варианта.

Папка UnitTests на одном уровне с папкой Source

- За — все модульные тесты находятся в одном месте.
- За — переносимый код отделен от непереносимого.
- За — в процессе сборки можно легко запустить модульные тесты всех проектов.

- Против — разработчикам сложнее запускать модульные тесты только для своих проектов.
- Против — модульные тесты не будут включаться в ветвление исходного кода.

Своя папка UnitTests в каждом проекте

- За — разработчики могут легко запускать модульные тесты для конкретного проекта.
- За — при ветвлении модульные тесты будут включены в разветвленные папки, то есть, останутся связанными с исходным кодом в каждой ветви.
- Против — в папке исходного кода смешиваются рабочий и тестовый коды.
- Против — как правило, во время сборки сложнее становится одновременно запустить все модульные тесты всех проектов.

Хранение документации

Папка Documentation предназначена для документации проекта. Решая, что хранить в системе управления исходным кодом и что хранить в библиотеке документов портала проекта Microsoft Windows SharePoint®, примите во внимание следующие соображения:

- Используйте SharePoint для внутренних документов команды — сценариев использования, требований, плановой и проектной документации.
- Используйте систему управления исходным кодом TFS для документации, поставляющейся заказчику, — руководств по установке и развертыванию, руководства пользователя, файлов справки.

Большинство документов представляют собой двоичные файлы, поэтому подумайте об использовании блокировки для предотвращения ручных слияний. Установив блокировку, вы будете получать уведомления о том, что файл открыт, и избежите ручного слияния.

Будьте осторожны при использовании SharePoint и внимательно следите за версиями документов. В SharePoint гораздо проще перезаписать изменения по сравнению с системой управления исходным кодом TFS, поскольку по умолчанию в SharePoint включена перезапись существующих файлов при их выгрузке.

Структура на стороне клиента

Структура локальных папок на рабочих станциях разработчиков должна повторять структуру папок на сервере. Чтобы правильно организовать все компоненты на рабочих станциях, разместите исходный код всех командных проектов в одной корневой папке, например, C:\DevProjects. Создайте в ней одну вложенную папку для каждого командного проекта, как показано в следующем примере:

```
{
C:\DevProjects                →Корневая папка для всех командных проектов
      \MyTeamProject1         →Папка проекта TeamProject1
      \MyTeamProject2         →Папка проекта TeamProject2
}
```

В каждой папке командного проекта разместите копию структуры папок приложения на сервере управления исходным кодом, как показано в следующем примере:

```
{
\MyTeamProject1                          →Папка проекта TeamProject1
      \Main                              →Содержит файлы .sln с проектами
            \Source
                  \MyApp 1               →Файл  MyApp1.sln
                     \Source
                        \ClassLibrary1   →Файл ClassLibrary1.csproj
                        \MyApp1Web       →Файл Default.aspx
                         \UnitTests      →Проекты и исходный код для
                                            модульного тестирования
                        \ClassLibrary1Tests
                        \MyWinApp1Tests
                  \SharedBinaries        →Общие ресурсы, например,
                                            библиотеки
                  \SharedSource          →Общий исходный код
            \Docs                        →Документация проекта
            \Tests                       →Общая папка для тестирования
                  \FunctionalTests
                  \PerformanceTests
                  \SecurityTests
}
```

> **Примечание** Структура на стороне клиента автоматически отражает структуру на сервере, если вы при помощи рабочей области сопоставляете корневую папку приложения с локальным компьютером. Однако в крупных проектах это может привести к увеличению времени загрузки рабочей области. Чтобы оптимизировать работу в крупных проектах, создавайте сопоставления рабочей области ниже уровня корневой папки для извлечения только файлов, требуемых для разработки.

Ветвление папок

Чтобы изолировать разработку при помощи ветвления, создайте дополнительные папки в папке Main или разместите в ней дополнительные файлы решений (.sln) Visual Studio, которые позволят разработчикам работать с различными группами проектов. Ветви в подпапках исходного кода Main

могут использоваться для текущего сопровождения выпусков продукта или параллельных потоков разработки.

В приведенном ниже примере в дополнение к корневой папке Main используется папка Development (ответвление Main), предназначенная для изоляции по функциям или по группам. Папка Releases предназначена для хранения ветвей выпусков (еще одно ответвление Main) и обеспечивает изоляцию выпущенных сборок, требующих текущего сопровождения, и текущего выпуска.

```
{
$MyTeamProject1
      /Development
            /FeatureBranch1
                  /Source
                        /MyApp
            /FeatureBranch2
                  /Source
                        /MyApp
      /Main
            /Source
      /Releases
            /Release1 Сопровождение
                  /Source
                        /MyApp
            /Release2 Сопровождение
                  /Source
                        /MyApp
            /Release3 Изоляция текущего выпуска
                  /Source
                        /MyApp
}
```

> **Примечание** Не увлекайтесь ветвлением. Если потребуется, вы можете пометить выпуск и выполнить ветвление позже.

Подробнее о группировании проектов и структуре решений — в главе 3. Подробнее о сценариях ветвления и структурах связанных папок — в главе 5.

Знакомство с рабочими областями

Рабочая область (workspace) TFS является клиентской копией файлов и папок из системы управления исходным кодом TFS. По сути, рабочая область отображает папки системы управления исходным кодом в папки локальной системы. Допустим, вы изменяете файлы в рабочей области локального компьютера. Эти *незавершенные* (pending) изменения изолируются в рабочей области, пока вы

не вернете их на сервер как целое. Совокупность изменений, возвращаемых на сервер пакетом, называется *набором изменений* (changeset).

Одна рабочая область может содержать ссылки на несколько проектов. С помощью рабочей области вы также можете изолировать файлы или версии для персонального пользования. Рабочие области назначаются компьютеру или учетной записи пользователя. Поэтому на каждом компьютере, которым вы пользуетесь, рабочие области могут определяться по-своему. Кроме того, как пользователь вы можете иметь несколько рабочих областей на одном компьютере.

> **Примечание** Все физические положения в локальной файловой системе вы можете отобразить посредством только одной рабочей области. Однако вы вольны при помощи разных рабочих областей сопоставить любой папке *сервера* разные локальные папки.

Создание сопоставления рабочей области

Сопоставления являются рекурсивными, поэтому когда вы создаете новое сопоставление рабочей области и выполняете операцию Get Latest Version в корневой папке этой рабочей области, автоматически создается вся структура локальных папок, идентичная структуре на сервере.

Создавая сопоставления новых рабочих областей, имейте в виду следующее:

- Прежде чем в первый раз добавить решение в систему управления исходным кодом, убедитесь в том, что в локальной системе используется правильная структура папок.
- Впервые создавая сопоставление рабочей области для проекта и выполняя операцию Get Latest, убедитесь, что корневая папка проекта сопоставлена соответствующей локальной папке, например, C:\DevProjects\TeamProject1.

Где хранятся сопоставления рабочих областей?

Информация о рабочих областях хранится как на клиентском компьютере, так и на сервере. На клиентском компьютере она содержится в файле Version Control.config, расположенном в папке \Documents and Settings\[user]\Local Settings\Application Data\Microsoft\Team Foundation\1.0\Cache.

В файле VersionControl.config имя рабочей области сопоставлено с папкой на локальном компьютере. Конкретные папки системы управления исходным кодом с локальными папками в нем не сопоставляются. Эта информация содержится на сервере в нескольких таблицах (в частности, tbl_Workspace и tbl_workingfolder) в БД TfsVersionControl.

Сокрытие

Сокрытие (cloaking) используется для повышения производительности, а также для предотвращения извлечения части дерева исходного кода. Ниже описаны типичные ситуации, в которых используется сокрытие:

- Вы выполняете сборку проекта локально, и некая папка для сборки не требуется (например, папка с документацией).
- Вы работаете в команде крупного проекта и хотите извлекать только часть проекта.

В подобных случаях скрывайте папки, чтобы прекратить их извлечение клиентами. Для этого нужно внести изменения в рабочую область на клиентском компьютере, изменив статус рабочей папки с активного на скрытый. При этом нужно иметь в виду следующие соображения:

- Не скрывайте отдельные файлы. Весьма вероятно, что в будущем это обернется проблемами в сопровождении проекта.
- В больших проектах вместо создания нескольких рабочих областей включайте в сопоставление корневую папку и скрывайте вложенные папки.

Какие файлы включать в систему управления версиями?

Ниже приводится список основных типов файлов, которые следует добавлять в систему управления исходным кодом. Именно они добавляются, когда вы щелкаете **Add Solution to Source Control**.

- **Файлы решений (*.sln)** Список составляющих проектов, информация о зависимостях, а также о конфигурации сборки и поставщике системы управления исходным кодом.
- **Файлы проекта (*.csproj или *.vbproj)** Параметры сборки, ссылки на файлы сборки (по имени или пути) и опись файлов.
- **Метаданные проекта Visual Studio Source Control (*.vspscc) n** Связыва-ния проектов, списки исключений, имена поставщиков и другие метаданные системы управления исходным кодом.
- **Файлы конфигурации приложения (*.config)** XML-файлы конфигурации с индивидуальными настройками проекта и приложения для управления поведением приложения во время его выполнения. Для веб-приложений используются файлы Web.config, для остальных приложений — файлы App.config.

Примечание Во время выполнения система сборки Visual Studio копирует файл App.config в папку Bin проекта и переименовывает его в <имя_приложения>.exe.config. Для всех приложений, кроме веб-приложений, файл конфигурации в новый проект автоматически не добавляется. Если он вам нужен, добавьте его вручную. Убедитесь, что присвоили ему имя App.config, и поместите его в папку проекта.

- **Файлы исходного кода (*.aspx, *.asmx, *.cs, *.vb, ...)** Исходный код для разных приложений и языков.
- **Двоичные файлы (*.dll)** Если ваш проект зависит от двоичных файлов, например, от DLL-библиотек сторонних производителей, их также следует добавить в проект. Подробнее об управлении зависимостями — в главе 6.

Какие файлы не включать в систему управления исходным кодом?

Перечисленные ниже файлы индивидуальны для каждого разработчика, поэтому их не следует добавлять в систему управления версиями:

- **Файлы пользовательских параметров решения (*.suo)** Персональные настройки среды IDE Visual Studio для конкретного пользователя.
- **Файлы пользовательских параметров проекта (*.csproj.user или *.vbproj.user)** Персональные параметры проекта, относящиеся к разработке, а также при необходимости путь к файлам сборки, на которые есть ссылка в проекте.
- **Файлы WebInfo (*.csproj.webinfo или *.vbproj.webinfo)** Расположение виртуального корневой папки проекта. Эти файлы не добавляются в систему управления исходным кодом, чтобы отдельные разработчики могли определять для рабочей копии проекта различные виртуальные корневые папки. Впрочем, несмотря на наличие этой возможности, при разработке веб-приложений всем членам команды рекомендуется использовать согласованное расположение виртуальной корневой папки.
- **Результаты сборки** Сборки DLL, межоперационные сборки DLL и исполняемые файлы (EXE). Помните, что сборки, например, двоичные файлы сторонних производителей, которые не участвуют в процессе сборки, тем не менее, следует помещать в систему управления версиями.

Резюме

Для эффективной командной разработки необходимо структурировать проекты в системе управления исходным кодом TFS. Для объединения проектов Visual Studio используйте корневую папку Main, создавая в ней вложенные папки для хранения различных данных проекта — исходного кода, тестов, документации и типов командной сборки.

Используйте SharePoint для хранения внутренней документации, например, сценариев использования и проектной документации. Для документации продукта, которую вы планируете поставлять клиентам (руководств по установке и развертыванию, руководства пользователя, файлов справки), используйте систему управления исходным кодом TFS.

Синхронизируйте структуру папок на сервере и клиенте, чтобы сократить организационную путаницу. Для оптимизации работы в крупных про-

ектах создавайте сопоставления рабочих областей для корневой папки, чтобы обеспечить извлечение только нужных в работе файлов.

Дополнительные ресурсы

- **Перечисленные ниже разделы из части «Практикум»** В них подробно описаны процедуры, обсуждаемые в этой главе.
  - «Как создать дерево кода в Team Foundation Server».
  - «Как структурировать приложения ASP.NET в Visual Studio Team Foundation Server».
  - «Как структурировать приложения Windows в Visual Studio Team Foundation Server».
  - «Как структурировать папки управления исходным кодом в Visual Studio Team Foundation Server».
- Дополнительную информацию о системе управления исходным кодом Team Foundation вы найдете в статье «Using Source Code Control in Team Foundation» по адресу *http://msdn2.microsoft.com/en-us/library/ms364074(VS.80).aspx*.
- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.

ГЛАВА 5

Стратегии ветвления и слияния

В этой главе

- Обоснование необходимости ветвления.
- Выбор стратегии ветвления и слияния для проекта.
- Использование обычной стратегии ветвления в больших командах.
- Структуры папок для различных сценариев ветвления.

Обзор

В этой главе представлены стратегии ветвления и слияния для некоторых типичных сценариев. Как правило, ветвление нужно для поддержки выпусков или параллельных разработок.

Во многих простых сценариях ветвление не требуется. Вполне достаточно помечать сборки. Например, при помощи метки можно в любой момент заново создать сборку или выяснить, какие версии исходного файла были использованы при создании той или иной сборки. Следует подумать о ветвлении, если вам требуется изолирование параллельных групп для работы над разными, но перекрывающими функциями или для поддержки выпуска.

Чтобы в целом познакомиться с ветвлением и слиянием, читайте главу от начала до конца. Так вы узнаете о различных стратегиях ветвления и различных сценариях, при которых ветвление имеет смысл. Если вас интересует конкретный сценарий, перейдите к соответствующему разделу.

Сценарии ветвления и слияния

Ниже представлены примеры сценариев, в которых может потребоваться создание ветвей и выполнение слияний:

- Если вы постоянно сталкиваетесь с проблемой сломанных сборок, создайте ветвь разработки, чтобы изолировать параллельные варианты разработки.
- Если при выполнении некоторых функции или взаимодействии некоторых групп возникают проблемы с устойчивостью, создайте для этих функций или групп отдельные ветви в папке разработки системы управления исходным кодом.

Не выполняйте ветвление без необходимости, поскольку оно добавит вам работы по обслуживанию дополнительного дерева исходного кода и по слиянию. Большинство групп разработчиков, создающих коммерческие приложения, работают по короткому циклу выпуска и не нуждаются в ветвлении. Команды разработчиков, работающие с более продолжительными циклами, например, независимые поставщики ПО, чаще нуждаются в ветвлении.

При использовании одного потока разработки или осуществлении инкрементного и непрерывного выпуска ветви следует создавать ветви, только если изменения часто приводят к сбоям и вносят неустойчивость в процесс разработки.

Типичные сценарии

Ниже приведены наиболее типичные сценарии ветвления:

- **Сценарий 1 — без ветвей.** Команда работает только из главного дерева исходного кода. В этом случае вы не создаете ветвей и не нуждаетесь в изоляции. Данный сценарий соответствует, главным образом, мелким и средним командам, не требующим изолирования групп, функций или выпусков.
- **Сценарий 2 — ветвь выпуска.** Команда создает ветви для текущего сопровождения выпуска. Это наиболее распространенная ситуация. В этом случае ветвь создается для обеспечения устойчивости очередного выпуска до его выхода, а затем опять сливается с главным деревом исходного кода.
- **Сценарий 3 — ветвь сопровождения.** Команда создает ветвь для обслуживания предыдущей сборки, чтобы делать это, не нарушая устойчивости текущих сборок. Обратное слияние изменений ветви сопровождения с главным деревом производится при необходимости. Допустим, оно не нужно, если вы вносите для группы клиентов специфические оперативные изменения, который не хотите включать в главную сборку.
- **Сценарий 4 — ветвь функции.** Команда создает ветви, основываясь на функциях. То есть, вы создаете ветвь разработки, выполняете в ней нужную работу, а затем производите слияние с главным деревом исходного кода. Вы можете создавать самостоятельные ветви для работы над отдельными функциями, окончательная доработка которых будет происходить параллельно.
- **Сценарий 5 — ветвь группы.** Ветви создаются для изолирования подгрупп, чтобы они могли работать, не мешая друг другу или двигаясь параллельно к различным контрольным точкам.

Вы можете столкнуться как с одним, так и с несколькими из этих сценариев. Используйте их в качестве ориентира.

Примеры папок и их назначение

В этом разделе приведены примеры папок, создаваемых в системе управления исходным кодом Microsoft® Visual Studio® Team Foundation Server (TFS) при структурировании дерева кода по сценариям ветвления и слияния.

- Папка **Main** содержит главное дерево исходного кода. Здесь соединяются изменения из других ветвей.
- Папка **Development** является ответвлением **Main** и используется для изолирования текущей разработки. Подобные ветви могут создаваться временно, например, для проверки рискованных изменений, без влияния на **Main**.
- Папка **Releases** содержит ветви выпусков, которые уже осуществлены, и сейчас осуществляется поддержка клиентов. Она обеспечивает изоляцию текущей разработки, которая ведется в ветви Development, а также содержит ветвь текущего выпуска, которая отделена от **Main** и хранит версию, заблокированную до выпуска. В этой ветви вы готовите ПО к выпуску, тогда как остальные продолжают работать в ветви **Development** над новыми функциями.

В следующих разделах на конкретных примерах показано использование ветвления в каждом из описанных сценариев.

Сценарий 1 — без ветвей

Этот сценарий применяется, в основном, в небольших группах, где не нужна изолированная разработка. Помечая сборки, вы можете извлекать исходный код, соответствующий конкретной сборке. В ветвлении нет нужды, поскольку вы можете работать напрямую из папки **Main**. Ниже приводится структура папок в сценарии без ветвления:

```
{
My Team Project
      Main
            Source
}
```

Сценарий 2 — ветвь выпуска

В этом сценарии команда создает ветвь для стабилизации выпуска, а затем, после выпуска программы, производит слияние ветви с главным деревом исходного кода. Ниже приводится структура папок в сценарии ветвления выпусков:

```
}
My Team Project
      Main                        →Главная ветвь сборки
            Source
      Releases
```

```
            Release 1            →Ветвь выпуска
                  Source
}
```

Сценарий 3 — ветвь сопровождения

В этом сценарии вы создаете ветвь для сопровождения, чтобы не нарушить устойчивость текущих сборок. Далее показана структура папок с ветвлением для сопровождения. Она очень похожа на структуру с ветвлением для выпуска, однако сохраняется на более продолжительное время:

```
}
   My Team Project
         Main                        →Главная ветвь сборки
               Source
         Releases                    →Контейнер для ветвей сопровождения
               Release 1             →Ветвь сопровождения
                     Source
                     Другие папки ресурсов
               Release 2             →Ветвь сопровождения
                     Source
                     Другие папки ресурсов
}
```

Сценарий 4 — ветвь функции

В этом сценарии вы создаете ветвь разработки на основе функций продукта, выполняете в ней работу, а затем производите слияние с главным деревом исходного кода. Ниже приводится структура папок в сценарии ветвления с целью разработки функциональных особенностей:

```
{
   My Team Project
         Development             →Изолированный контейнер ветвей разработки
               Feature A         →Ветвь функции
                     Source
               Feature B         →Ветвь функции
                     Source
               Feature C         →Ветвь функции
         Main                    →Главная ветвь сборки
               Source
}
```

Сценарий 5 — ветвь группы

Этот сценарий похож на сценарий ветвления по функциям, за исключением того что здесь вы организуете ветви разработки не по функциям конечного

продукта, а в соответствии с подгруппами основной команды разработчиков. Иногда группы и функции точно соответствуют друг другу, но в других случаях группа может работать над несколькими функциями. Ниже приводится структура папок в сценарии ветвления по группам:

```
{
   My Team Project
         Development          →Изолированный контейнер ветвей разработки
               Team 1               →Ветвь группы
                     Feature A      →Изолированная ветвь разработки
                           Source
                     Feature B      →Изолированная ветвь разработки
                           Source
               Team 2               →Ветвь группы
                     Feature A      →Изолированная ветвь разработки
                           Source
                     Feature B      →Изолированная ветвь разработки
                           Source
         Main                       →Главная ветвь сборки
               Source
}
```

Логическая структура

Логическая структура выражает для каждой ветви отношения «родитель – потомок». Она может отличаться от физической структуры, отображаемой в Source Control Explorer. Например, в показанной выше физической структуре **Development** и **Main** являются папками одного уровня, тогда как с точки зрения логики **Development** является дочерней папкой **Main**.

На рис. 5-1 показан пример логической структуры ветвлений и слияний.

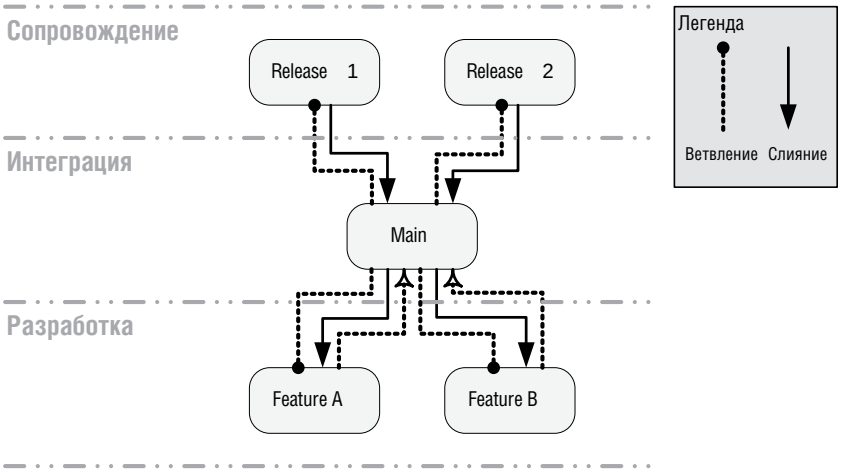


Рис. 5-1. *Логическая связь ветвлений и слияний*

Сценарий выпуска

На рис. 5-2 представлена типичная последовательность действий при ветвлении для выпуска.

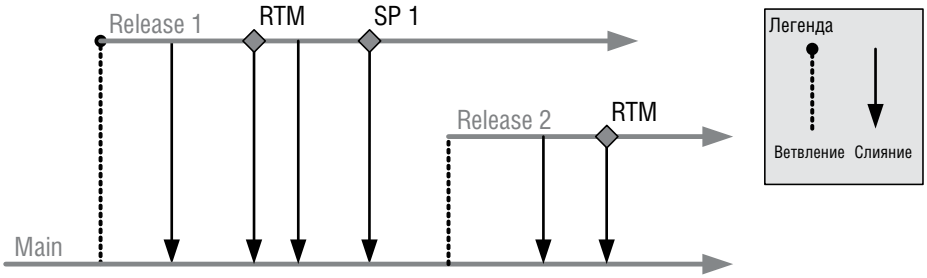


Рис. 5-2. *Последовательность действий при ветвлении для выпуска*

Последовательность действий такова:

1. Когда выпуск готов к изоляции, в Main создается ветвь Release 1.
2. Периодически проводятся слияния с Main, чтобы переместить серьезные исправления в главную ветвь сборки.
3. В ветви RTM выпуск помечается с последующей слиянием с Main.
4. Выпущен пакет обновлений SP1. Сборка помечается, выполняется слияние изменений с Main.
5. Ветвь Release 1 оставляется для поддержки SP1 и для последующих пакетов обновлений.

   Процесс повторяется для следующих выпусков.

Примечание Успешное применение ветвления для выпуска требует, чтобы вы точно выбирали ветвь, в которой будут применены исправления. При выпуске как простых исправлений, так и пакета обновлений следует сначала применять их в ветви текущего выпуска, а затем выполнить слияние с Main, чтобы исправления применялись и в последующих выпусках.

Сценарий изолированной разработки

На рис. 5-3 представлена типичная последовательность действий при ветвлении для изолирования разработки.

Последовательность действий такова:

1. Создается ветвь Feature A для изолирования разработки первой функции.
2. Создается ветвь Feature B для изолирования разработки второй функции.
3. Каждая группа выполняет слияние с **Main** по мере достижения контрольных вех, когда функцию можно объединять с главной сборкой для обеспечения доступа к ней других групп.

4. Каждая группа по графику проводит слияние с Main с целью извлечения результатов работы других групп и уменьшения числа конфликтов при проведении последующих слияний.
5. Когда функция готова, производится слияние изменений с Main, а ветвь функции удаляется.

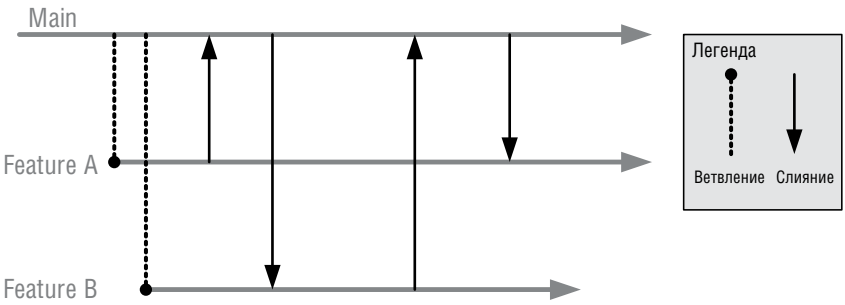


Рис. 5-3. *Последовательность действий при ветвлении для изолирования разработки*

Различные аспекты ветвления

Проводя ветвление, имейте в виду следующее:

- Не следует выполнять ветвление, если команда разработчиков не нуждается в одновременной работе над одними и теми же наборами файлов. Если у вас нет уверенности по этому поводу, пометьте сборку и позже при необходимости создайте из нее ветвь. Слияние ветвей — длительный и сложный процесс, особенно при наличии существенных различий.
- Организуйте деревья ветвей таким образом, чтобы слияния выполнялись только вдоль иерархии (вверх и вниз по дереву), но не поперек иерархии. Ветвление «по горизонтали» требует осуществления слияния без основы (baseless merge), при котором чаще приходится разрешать конфликты вручную.
- Основу иерархии ветвей составляют родительские и дочерние ветви, структура которых может отличаться от физической структуры кода на диске. При планировании слияний учитывайте именно логическую, а не физическую структуру ветвей.
- Не выполняйте слишком глубокого ветвления. Чем оно глубже, тем больше тратится времени на выполнение слияний и разрешение конфликтов. Это может отрицательно сказаться на графике работы по проекту и затянуть исправление ошибок.
- Выполняйте ветвление на высоком уровне и включайте в него файлы конфигурации и исходного кода.
- Постоянно развивайте структуру ветвления.

- Слиянием и разрешением конфликтов занимается один или несколько разработчиков. Код, полученный в результате слияния, должен быть всесторонне протестирован, так как нередки случаи, когда неверное слияние приводит к потере устойчивости сборки.
- Слияние «по горизонтали» особенно сложно и требует разрешения вручную многих конфликтов, которые в ином случае могли бы быть разрешены автоматически.

Решение о проведении ветвления в конечном итоге сводится к сравнительной оценке затрат на непрерывное разрешение конфликтов в реальном времени и на их периодическое разрешение при слиянии ветвей.

Особенности крупных проектов

Большие команды разработчиков с длительными циклами разработки, как правило, отличаются от небольших команд по следующим признакам:

- Требуют более сложной структуры ветвления и слияния.
- Чаще вынуждены управлять зависимостями между решениями и проектами.
- Чаще содержат несколько сборок для конкретных функций и групп.

В крупных командах чаще выполняется ветвление как по группам, так и по функциям. Как правило, в них также используются ветви для интеграции внешних зависимостей. Сложная структура ветвления увеличивает промежуток времени между изменением в ветви разработки и переносом этого изменения в главную ветвь сборки. Поэтому, создавая ветви, следует тщательно продумать стратегию слияния.

Далее перечислены соображения, которые нужно принимать в расчет, определяя, будут слияния проводиться по плану или при возникновении определенных событий:

- Обратная интеграция, например, слияние из ветви разработки в главную ветвь сборки, обычно планируется заранее.
- Прямая интеграция, например, при слиянии из главной ветви сборки в ветвь разработки, обычно, управляется событием, например, достижением вехи в разработке функции. Это происходит, когда группа, отвечающая за ветвь разработки, считает, что она готова выполнить слияние изменений из родительской ветви.

Корректируйте стратегию ветвления и слияния столь же часто, как вы производите сборки. Более глубокая структура ветвления требует дополнительного времени на перенос изменений из дочерних ветвей в главную ветвь сборки. Далее перечислены признаки приближения связанных с этим проблем:

- Исправленные в ветвях ошибки слишком долго распространяются «вверх», до главной ветви сборки, из-за чего возникают сбойные сборки.
- Функциям требуется слишком много времени для распространения до главной сборки, из-за чего пропускаются вехи.

- Слияние изменений между различными ветвями занимает слишком много времени.

Если в вашей команде возникли эти проблемы, подумайте над уменьшением глубины ветвления.

Далее приведен пример сложной структуры ветвления:

```
{
   My Team Project
      Development       →Контейнер для изолирования активных ветвей
разработки
         Feature A      →Изолированная ветвь разработки
            Source
         Feature B      →Изолированная ветвь разработки
            Source
      Main              →Главная ветвь компоновки и сборки. Здесь сходятся
все изменения.
         Source
         Папки с другими ресурсами
      Releases          →Контейнер для текущего выпуска и ветвей
сопровождения
         Release 2      →Активная ветвь сопровождения
            Source
            Папки с другими ресурсами
         Release 3      →Активная ветвь сопровождения
            Source
            Папки с другими ресурсами
         Release 4      →Ветвь с кодом, заблокированным до выпуска
            Sourse
            Папки с другими ресурсами
      Архив
         Release 1      →Старый выпуск в архиве
            Source
            Папки с другими ресурсами
}
```

В данную структуру включены:

- ветви функций, позволяющие нескольким группам работать изолированно;
- главная ветвь для сбора изменений всех групп, а также исправлений из ветвей выпусков;
- ветвь выпуска, используемая для изоляции текущего выпуска;
- ветвь сопровождения предыдущего выпуска, для которого все еще осуществляется активная поддержка;
- архивные ветви для старых выпусков, сопровождение которых прекращено.

Резюме

Ветви создаются для изоляции выпусков, для сопровождения предыдущего выпуска или для изоляции параллельной разработки. Не следует выполнять ветвление без особых на то причин.

Создавая ветви, логически структурируйте деревья так, чтобы слияния выполнялись только «по вертикали» (вверх и вниз по дереву), но не «по горизонтали». Слияние «по горизонтали» требует больше времени и чревато ошибками.

В больших командах структура ветвления более сложна. Готовьтесь к увеличению интервала времени, необходимого для переноса изменений из ветви разработки в главную ветвь сборки.

Дополнительные ресурсы

- **Раздел «Руководство по управлению исходным кодом»** В нем вы найдете дополнительные рекомендации по ветвлению и слиянию.
- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о ветвлении и слиянии в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

ГЛАВА 6

Управление зависимостями в системе управления исходным кодом Visual Studio Team System

В этой главе

- Управление зависимостями в системе управления исходным кодом Microsoft® Visual Studio Team System.
- Создание ссылок на проекты и файлы сборок из различных решений одного проекта.
- Создание ссылок на проекты и файлы сборок из других проектов.
- Создание ссылок на сборки сторонних производителей.
- Управление ссылками веб-служб в среде командной разработки.
- Управление ссылками на базы данных в среде командной разработки.

Обзор

В этой главе говорится о том, как обрабатывать зависимости внутри одного решения, а также между различными решениями Visual Studio. Продуманный подход к управлению зависимостями в среде командной разработки — необходимое условие повышения устойчивости сборки и сокращения затрат на обслуживание системы управления исходным кодом.

Зависимости связывают проекты, внешние файлы сборки, веб-службы и базы данных. С течением времени зависимости неизбежно меняются, что не может не затронуть порядок сборки приложения. Грамотное управление зависимостями повышает эффективность процесса сборки, одновременно делая сборки максимально безупречными.

Сценарии и решения

При управлении зависимостями, как правило, реализуются следующие сценарии:

1. Вы хотите установить ссылку на сборку, сгенерированную другим проектом в том же решении.
2. Вы хотите установить ссылку на сборку, сгенерированную проектом в другом решении.
3. Вы хотите установить ссылку на файл сборки, содержащийся в другом командном проекте.
4. Вы хотите установить ссылку на файл сборки стороннего производителя.

Ссылка на файл сборки другого проекта в том же решении

Чтобы создать ссылку на другую сборку того же решения Visual Studio, воспользуйтесь ссылкой на проект Visual Studio. С ее помощью вы дадите Visual Studio указание автоматически выполнять некоторые действия, например, синхронизировать конфигурации сборки (Debug/Release), отслеживать версии, а также при необходимости выполнять сборку компонентов.

Ссылка на файл сборки проекта в другом решении

Чтобы создать ссылку на сборку, сгенерированную проектом из другого решения Visual Studio, вы можете воспользоваться двумя способами:

- Используйте файловые ссылки для указания на двоичный файл сборки.
- Добавьте в решение проект Visual Studio (файлы проекта и исходного кода), а затем используйте ссылку на проект.

Файловые ссылки менее надежны, чем ссылки на проект. Они не подчиняются правилам конфигурации сборки и не отслеживаются зависимостями сборки Visual Studio. Поэтому при изменении файла сборки, на который указывает ссылка, Visual Studio не будет автоматически проводить сборку проекта.

В качестве альтернативы вы можете создать для внешнего проекта ветвь в вашем решении, выполнить сборку двоичного файла, а затем воспользоваться ссылкой на проект. Ссылка будет более надежна, хотя вам придется регулярно проводить слияния из ветви исходного кода в целях синхронизации изменений.

Ссылка на файл сборки из другого командного проекта

Есть два способа совместного использования исходным кода или двоичных файлов в нескольких проектах:

- **Ветвление** Вы создаете в текущем решении ветвь для исходного кода из другого командного проекта. При этом конфигурация, унифицирующая исходный код из общего расположения и вашего проекта, создается на сервере.

- **Сопоставление рабочей области** Вы отображаете исходный код из другого командного проекта в рабочую область на компьютере разработки. При этом конфигурация, унифицирующая исходный код из другого проекта и вашего проекта, создается на клиенте.

Ветвление более предпочтительно, потому что при этом зависимости хранятся на сервере управления исходным кодом. Сопоставление рабочей области реализуется только на клиентской стороне. Это значит, что для успешной сборки приложения каждый разработчик должен создать сопоставления на своем компьютере, а также на сервере сборки.

Ветвление приводит к добавлению еще одного слияния, но позволяет обновлять двоичный код более явным образом.

Ссылка на файл сборки от стороннего производителя

Этот сценарий очень похож на сценарий создания ссылок между командными проектами, за исключением того что в нем общим является только двоичный, но не исходный код. Выбор между ветвлением и рабочими областями проводится по тем же основаниям, но с поправкой на то, что издержки, скорее всего, будут меньше, поскольку сборки сторонних производителей, как правило, не подвержены частым изменениям.

Ссылки на проекты

Допустим, у вас имеется проект Visual Studio, например, командный проект, содержащий код общей библиотеки, который используется несколькими командными проектами. Вы можете управлять им как изнутри, так и при помощи отдельного командного проекта.

На рис. 6-1 показано, как выглядит структура папок системы управления исходным кодом TFS во втором случае.

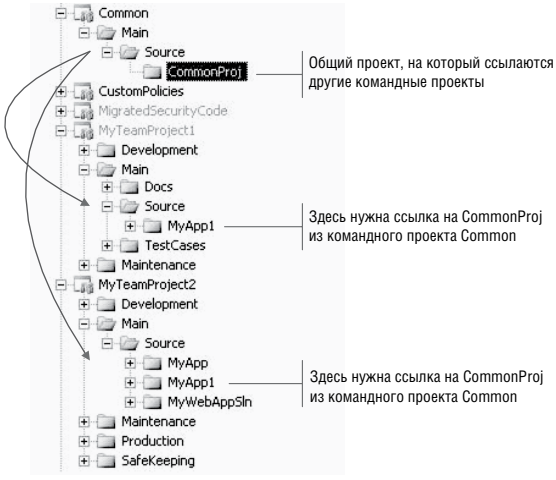


Рис. 6-1. *Структура папок общего проекта Common*

Получить доступ к общему проекту из своего командного проекта вы можете двумя способами:

- ветвление;
- сопоставление рабочей области.

Ветвление

Ветвление — предпочтительный способ в большинстве сценариев совместного использования исходного кода. Оно позволяет задействовать общий код в вашем проекте, включив его в систему управления исходным кодом вашей группы. В этом случае вы создаете в своем командном проекте ветвь для исходного кода из общего расположения. Конфигурация, объединяющая исходный код из общего расположения и ваш проект, создается на сервере.

Синхронизация изменений исходного кода выполняется в процессе слияния ветвей, и решение об извлечении изменений из общего исходного кода становится более явным и управляемым, хотя и приводит к дополнительным накладным расходам. Кроме того, упрощается использование Team Build: сопоставление производится на серверной стороне, а сопоставление на клиенте, которое нужно было бы копировать на сервер сборки, отсутствует.

Допустим, у вас есть два командных проекта TeamProject1 и $Common, причем проект $Common является совместно используемым. Вы создаете ветвь, ведущую из общего расположения в проект, где есть ссылка на него. Структура папок TFS должна быть похожа на структуру, показанную на рис. 6-2.

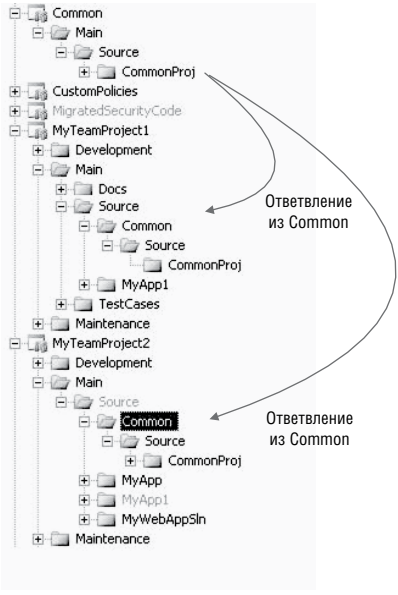


Рис. 6-2. *Использование ветвления*

Сопоставление рабочей области должно выглядеть примерно так:

Папка системы управления исходным кодом	Локальная папка
$/MyTeamProject1/Main/Source/	C:\MyTeamProject\Main\Source

Примерная структура папок рабочей области на стороне клиента показана на рис. 6-3.

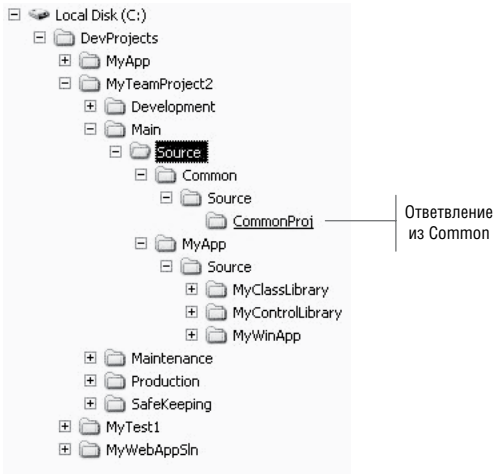


Рис. 6-3. *Сопоставление рабочей области на клиентской стороне*

Сопоставление рабочей области

Чтобы изменения кодов общего ресурса немедленно становились доступными всем разработчикам без появления издержек, связанных с ветвлением и слиянием, отобразите код из общего проекта в рабочую область на компьютере разработчика. Конфигурация, объединяющая исходный код из общего расположения и ваш проект, создается на клиентской стороне.

Преимущество этого метода состоит в том, что изменения в общем проекте учитываются каждый раз, когда вы копируете последнюю версию исходного кода в вашу рабочую область. Однако это усложняет использование Team Build, поскольку сопоставление рабочей области производится на клиенте.

Допустим, у вас есть два командных проекта $MyTeamProject и $Common, причем $Common — проект с общим исходным кодом. Эти проекты совместно используют общий путь на клиентском жестком диске. Структура папок рабочей области на стороне клиента будет похожа на структуру, показанную на рис. 6-4.

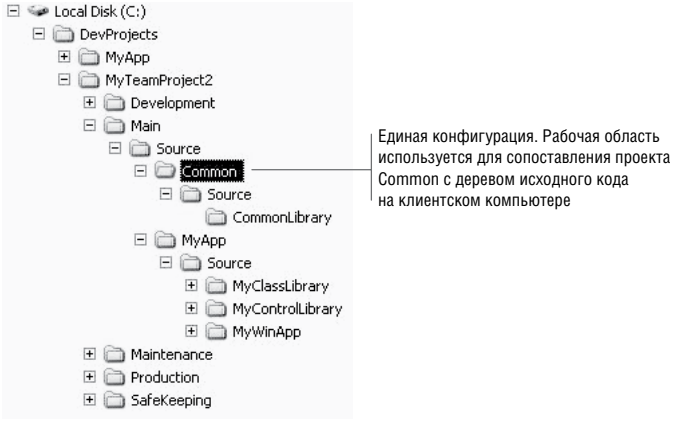


Рис. 6-4. *Использование сопоставления рабочей области*

Сопоставление рабочей области выглядит примерно так:

Папка системы управления исходным кодом	Локальная папка
$/MyTeamProject2/Main/Source/	C:\DevProjects\MyTeamProject2\Main\Source\
$/Common	C:\DevProjects\MyTeamProject2\Main\Source\Common

Дополнительную информацию вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Ссылки на файлы сборки сторонних производителей

Если вам нужно создать ссылку на сборку за пределами текущего набора проектов, например, на библиотеку стороннего производителя, причем вы не можете или не хотите создавать ветвь из того проекта в ваш проект, и нет возможности использовать ссылку на проект, вы должны создать файловую ссылку.

Работа с общими двоичными файлами похожа на работу с общим исходным кодом. Решите, где вы будете хранить двоичные коды и как ваша группа должна осуществлять доступ к ним. Если у вас есть двоичные файлы, используемые в нескольких командных проектах, вы вольны управлять проектами как из командного проекта группы-владельца, так и из проекта, специально созданного для совместно используемых двоичных кодов.

Группам, пользующимся общими двоичными кодами, доступны те же два варианта, что и для ссылок на проект:

- ветвление;
- сопоставление рабочей области.

Ветвление

В этом сценарии вы выполняете ветвление двоичных кодов из общего расположения в ваш командный проект. Конфигурация, объединяющая двоичные коды из общего расположения и из проекта, создается на серверной стороне.

Отличие заключается в том, что все изменения двоичных кодов, например, новые версии, учитываются в процессе слияния между ветвями. Решение об учете измененных общих двоичных кодов принимается намного более явно.

Допустим, у вас есть два командных проекта $TeamProject1 и $Common, приме Common содержит общие двоичные файлы. Создайте ветвь из общего расположения в проект, ссылающийся на него. Структура папок TFS должна выглядеть примерно так, как показано на рис. 6-5.

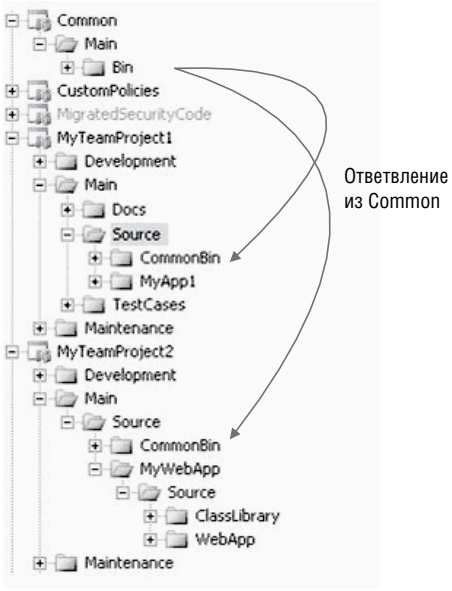


Рис. 6-5. *Ветвление из общего проекта Common*

Сопоставление рабочей области должно выглядеть примерно так:

Папка системы управления исходным кодом	Локальная папка
$/MyTeamProject1/Main	C:\MyTeamProject1\Main

Структура папок рабочей области на стороне клиента должна выглядеть примерно так, как показано на рис. 6-6.

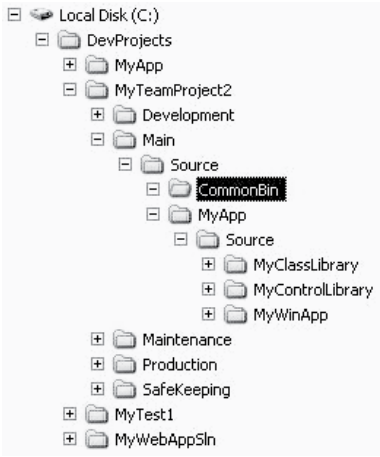


Рис. 6-6. *Структура папок рабочей области на стороне клиента*

Сопоставление рабочей области

В этом случае в командном проекте, пользующемся общими двоичными кодами, вы создаете ссылки на двоичные коды в общем расположении рабочей области на компьютере разработчика. Конфигурация, объединяющая двоичные коды из общего расположения и из проекта, создается на стороне клиента.

Преимущество этого способа состоит в том, что изменения в общих двоичных кодах учитываются каждый раз, когда вы извлекаете последнюю версию исходного кода в рабочую область.

Например, если у вас есть проекты $TeamProject1 и $Common, причем $TeamProject1 ссылается на двоичные коды из $Common, структура папок рабочей области на стороне клиента должна напоминать структуру, представленную на рис. 6-7.

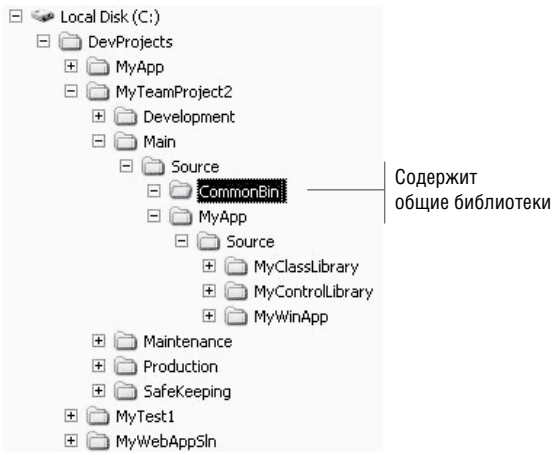


Рис. 6-7. *Хранение общих библиотек*

Сопоставление рабочей области должно выглядеть примерно так:

Папка системы управления исходным кодом	Локальная папка
$/MyTeamProject2/Main/	C:\DevProjects\MyTeamProject2\Main\
$/Common/Main/Bin	C:\DevProjects\MyTeamProject2\Main\Source\CommonBin

Дополнительную информацию вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Руководство по ссылкам на проекты и файлы сборок

Создать ссылку на файл можно одним из двух способов:

- Чтобы создать ссылку на сборку .NET Framework, выберите сборку в списке, отображенном на вкладке .**NET** диалогового окна **Add Referen-ces**.
- Используйте кнопку **Browse** в диалоговом окне **Add References**.

Такие сборки, как System.XML.dll, расположены в глобальном кеше сборок (Global Assembly Cache, GAC). Непосредственно сослаться на сборку внутри GAC вы не можете. Когда вы выбираете сборку на вкладке .**NET** диалогового окна **Add References**, на самом деле вы ссылаетесь на копию сборки в папке %windir%\Microsoft.NET\Framework\<версия>\.

Ссылки на проекты предпочтительнее ссылок на файлы. При управлении ссылками сборок, имейте в виду следующие соображения:

- Везде, где только возможно, используйте ссылки на проект.
- Используйте ссылки на файлы только при необходимости.
- Для ссылок на проект и на файл используйте **Copy Local = True**.

Автоматическое отслеживание зависимостей

Каждый раз, когда вы выполняете сборку локального проекта, система сравнивает дату файла сборки, на который имеется ссылка, с датой рабочей копии на компьютере разработчика. Если сборка, на которую имеется ссылка, оказалась новее, «свежая» версия копируется в локальную папку. Одно из преимуществ данного способа состоит в том, что ссылка на проект, установленная разработчиком, не блокирует DLL-библиотеку на сервере и никак не мешает процессу сборки.

Ссылки на веб-службы

В простых системах, где все проекты содержатся внутри одного командного проекта, разработчики рано или поздно обзаводятся локальными копиями всех веб-служб, потому что они определены проектами Visual Studio внутри

командного проекта. Когда вы впервые открываете решение из системы управления исходным кодом, все проекты (включая все веб-службы) устанавливаются локально. Если веб-служба добавляется к решению другим разработчиком, она устанавливается локально при очередном обновлении решения из системы управления исходным кодом. В этом сценарии не нужно публиковать веб-службы на центральном веб-сервере внутри командной среды.

В более крупных системах веб-службы могут быть опубликованы на веб-сервере с централизованным доступом при помощи IIS (Internet Information Server). В локальной установке веб-служб нуждаются не все разработчики. Они могут получить доступ к нужной веб-службе из своих клиентских проектов, но для этого потребуется соответствующим образом настроить ссылку.

Используйте в ссылках на веб-службы динамические URL

Если вы хотите обращаться к веб-службе, необходимо добавить в ваш проект ссылку на нее. При этом генерируется прокси-класс, посредством которого вы взаимодействуете с веб-службой. Код прокси изначально содержит статические URL веб-служб, например, http://localhost или http://SomeWeb-Server.

> **Важно!** В решении для веб-служб, выполняющихся на вашем компьютере, всегда используйте http://localhost, а не http://ИмяМоего Компьютера, чтобы обеспечить работоспособность ссылки на всех компьютерах.

Статический URL, встроенный в прокси, как правило, не совпадает с URL, который требуется в средах производства и тестирования. Обычно URL меняется по мере того, как ваше приложение проходит путь от разработки к тестированию и производству. Есть три способа решения этого вопроса:

- Программно установить URL веб-службы во время создания экземпляра прокси-класса.
- Задать для свойства **URL Behavior** ссылки на веб-службу значение **Dynamic**. Это предпочтительный способ. При установке для свойства значения **Dynamic** в прокси-класс добавляется код, извлекающий URL веб-службы из пользовательского раздела файла конфигурации приложения — Web.config для веб-приложения или SomeApp.exe.config для приложений Windows.
- Сгенерировать прокси с помощью инструмента командной строки WSDL.exe, задав параметр /urlkey. При этом происходит примерно то же самое, что и при установке свойства **URL Behavior**, но в этом случае URL хранится в разделе <applicationSettings> файла конфигурации приложения.

Использование динамического URL позволяет создавать пользовательский файл конфигурации, который будет перекрывать основной файл конфигурации приложения. Это дает разработчикам и членам группы тестирования возможность временно перенаправить ссылку веб-службы в другое расположение.

Как использовать динамический URL и файл конфигурации пользователя

Задайте для свойства **URL Behavior** значение **Dynamic**, чтобы добиться максимальной гибкости конфигурации в средах разработки и производства. По умолчанию при добавлении веб-ссылки Visual Studio присваивает свойству именно это значение. Чтобы проверить это, выполните следующие действия:

1. В обозревателе **Solution Explorer** разверните список веб-ссылок.
2. Выделите по очереди каждую ссылку из списка.
3. Убедитесь, что свойству **URL Behavior** присвоено значение **Dynamic.**

При первом добавлении ссылки Visual Studio автоматически генерирует файл App.config. Этот файл содержит ссылку на веб-службу и параметры конфигурации. Он выглядит примерно так:

```
<configuration>
   <configSections>
      <sectionGroup name="applicationSettings" type="System.Configuration.
ApplicationSettingsGroup, System, Version=2.0.0.0, Culture=neutral, PublicK
eyToken=b77a5c561934e089" >
         <section name=" YourProject.Properties.Settings" type="System.
Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"
/>
      </sectionGroup>
   </configSections>
   <applicationSettings>
      <YourProject.Properties.Settings>
         <setting name="SomeService_ localhost _Service"
serializeAs="String">
            <value>http://localhost/someservice/Service.asmx</value>
         </setting>
      </YourProject.Properties.Settings>
   </applicationSettings>
</configuration>
```

В этом файле есть новый раздел конфигурации, используемый прокси-классом. В нем содержится стандартный адрес веб-службы, заданный Visual Studio при создании прокси. Стандартный URL находится в файле Settings. Designer.cs. Чтобы просмотреть этот файл, выполните следующие действия:

1. В обозревателе Solution Explorer щелкните веб-службу правой кнопкой.
2. Выберите команду **View in Object Browser**. В окне Object Browser найдите вариант **YourProject.Properties**, где YourProject — имя проекта, в котором содержится ссылку на веб-службу.
3. Разверните узел **YourProject.Properties)** и дважды щелкните **Settings**. Откроется файл Settings.Designer.cs, в котором есть примерно такая строка:

```
[global::System.Configuration.DefaultSettingValueAttribute("http://
localhost:/webservice/Service.asmx")]
```

Это стандартное значение, используемое для URL веб-службы, если не найдено соответствующей конфигурационной информации.

Зачастую требуется изменить URL вызываемой веб-службы. Например, может понадобиться сначала тестирование работы с локальной веб-службой, а затем тестирование с версией веб-службы, работающей в тестовой среде. Желательно, чтобы окончательный URL веб-службы продукта отличался от URL, используемого во время разработки. Значение URL должно быть определено в пользовательском файле конфигурации, на который ссылается главный файл App.config. Имя пользовательского файла конфигурации произвольно. В следующем примере мы для ясности будем называть его User.config.

1. В обозревателе Solution Explorer щелкните правой кнопкой проект, в котором содержится ссылка на веб-службу, и выберите команды **Add** и **New Item**.
2. Выберите вариант **Application Configuration File**, задайте имя файла User.config и щелкните **Add**.
3. Скопируйте элемент <*YourProject.Properties.Settings*> из файла конфигурации приложения (App.config) в файл User.config. В этом файле должен присутствовать только элемент, который требуется изменить на время выполнения. Удалите директиву <?xml> и элемент <configuration>, если таковые имеется, как показано в следующем примере:

```
<YourProject.Properties.Settings>
   <setting name="SomeService_localhost_Service" serializeAs="String">
      <value>http://localhost/someservice/Service.asmx</value>
   </setting>
</YourProject.Properties.Settings>
```

Разработчик должен отредактировать содержимое файла User.config, указав в нем нужный URL. Теперь нужно сделать так, чтобы система использовала данные конфигурации из файла User.config, а не из файла App.config. Для этого следует исправить файл App.config следующим образом:

1. Добавьте атрибут **configSource="user.config"** в элемент <*YourProject.Properties.Settings*> главного файла конфигурации приложения. Когда во время выполнения будет считываться этот раздел, произойдет перенаправление на пользовательский файл конфигурации.
2. Удалите содержимое элемента <*YourProject.Properties.Settings*>.

Теперь файл App.config должен выглядеть так:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
      <sectionGroup name="applicationSettings" type="System.Configuration.
ApplicationSettingsGroup, System, Version=2.0.0.0, Culture=neutral, PublicK
eyToken=b77a5c561934e089" >
```

```
        <section name="YourProject.Properties.Settings" type="System.
Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"
/>
      </sectionGroup>
   </configSections>
   <applicationSettings>
      <yourProject.Properties.Settings configSource="user.config">
      </YourProject.Properties.Settings>
   </applicationSettings>
</configuration>
```

В предыдущем примере *YourProject* — имя проекта, содержащего ссылку на веб-службу.

> **Важно!** Если вы задали атрибут **configSource**, пользовательский файл конфигурации обязательно должен существовать и содержать только элемент <*YourProject.Properties.Settings*>. Добавляя атрибут **config Source ="user.config"** в основной файл конфигурации, не забудьте удалить из элемента <*YourProject.Properties.Settings*> содержимое XML.

Используя файл User.config, сделайте следующее:

- Убедитесь, что развернули файл User.config вместе с кодом приложения. Это делается в обозревателе Solution Explorer. Щелкните правой кнопкой файл User.config, выберите команду **Properties** и присвойте свойству **Copy To Output Directory** значение **Copy if newer**.
- Не добавляйте файл User.config в систему управления исходным кодом. При этом каждая группа разработчиков и тестировщиков сможет использовать собственный URL, задав его в собственном файле User.config. Система управления исходным кодом может содержать другие файлы User.config, например, для тестирования и промышленного использования, управление которыми должно осуществляться пользователями, ответственными за соответствующие среды. Эти файлы User.config следует хранить не в составе проектов веб-служб, а в других областях системы управления исходным кодом.
- Храните глобальный файл User.config в системе управления исходным кодом. Он может либо содержать только корневой элемент (не элемент <setting>), либо задавать расположение веб-службы по умолчанию. Чтобы система конфигурирования работала, файл User.config должен обязательно существовать.

Совет По умолчанию при добавлении решения пользовательский файл конфигурации автоматически добавляется в систему управления исходным кодом. Чтобы отказаться от этого, во время первого возврата файла после правки сбросьте флажок напротив User.config. Чтобы файл никогда не попал в систему управления исходным кодом, щелкните его правой кнопкой в обозревателе Solution Explorer и выберите вариант **Undo Pending Changes**.

Важно! Если файл User.config с определением URL содержит только корневой элемент и в нем нет элемента <setting>, прокси-класс веб-службы использует стандартный URL, указанный в файле Settings.Designer.cs. Это значение определяется атрибутом **DefaultValueSettings** и выглядит так:

```
[global::System.Configuration.DefaultSettingValueAttribute("http://
localhost/webservice/Service.asmx")]
```

Важно! Если в веб-приложении используется пользовательский файл конфигурации, любые изменения, внесенные в этот файл, по умолчанию приводят к автоматическому перезапуску приложения. Чтобы этого не происходило, добавьте атрибут **restartOnExternalChanges="false"** в раздел configuration, как показано в следующем примере:

```
<configSections>
  <sectionGroup name="applicationSettings" type="System.
Configuration.ApplicationSettingsGroup, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">
     <section name="Test.Properties.Settings" type="System.
Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermissio
n="false" restartOnExternalChanges="true"/>
  </sectionGroup>
</configSections>
```

Если вы добавите этот атрибут в раздел configuration файла Web.config, изменения во внешнем файле User.config не будут приводить к перезапуску веб-приложения, но будут ему видны.

Важно понимать, что при использовании этого механизма файл User.config обязательно должен существовать. Необходим ответственный, в чьи обязанности будет входить обеспечение правильной работы среды во время создания сборок для тестовых и рабочих выпусков. Соответствующий файл User.config, входящий в сборку, должен извлекаться из системы управления исходным кодом и копироваться в положение, где его сможет найти MSBuild.

Ссылки на базы данных

Ссылками на базы данных в форме строк подключения также можно управлять при помощи внешнего файла конфигурации. Преимущество данного способа состоит в том, что каждый разработчик может легко задать строку подключения в собственном файле конфигурации User.config. Изменения, внесенные одним разработчиком, например, перенаправление подключения в локальную БД для модульных тестов, не затронут остальных разработчиков.

Пользовательские файлы конфигурации можно также использовать для настройки специфических параметров среды, например, тестовой, задавая в User.config подключение к тестовой БД.

Процедура настройки ссылок на БД похожа на процедуру для ссылок на веб-службы, за исключением того что в прокси веб-службы содержится код для извлечения из файла конфигурации URL веб-службы. В случае БД вы должны предоставить код, позволяющий читать строку подключения.

Строки подключения в файлах конфигурации

Далее объясняется, как сохранить строку подключения к БД в пользовательском файле конфигурации и затем ссылаться на нее.

Чтобы использовать пользовательский файл конфигурации для хранения связывающих строк, выполните следующие действия:

1. Добавьте атрибут **configSource="user.config"** в элемент <connectionStrings> главного файла конфигурации приложения.

```
<configuration>
   <connectionStrings configSource="user.config"/>
</configuration>
```

2. Чтобы перекрыть настройки из главного файла конфигурации приложения, создайте файл User.config в той же папке, что и файл конфигурации приложения. Добавьте в него элемент <connectionStrings>. Обратите внимание, что приведенная далее строка подключения ссылается на локальную базу данных.

```
<connectionStrings>
   <add name="DBConnStr" connectionString="server=localhost;Integrated
Security=SSPI;database=Accounts"/>
</connectionStrings>
```

3. Внутри проекта используйте следующий код для получения строки подключения из пользовательского файла конфигурации. В нем использовано статическое свойство **ConnectionStrings** класса **System.Configuration.ConfigurationManager**. В приложении WinForm вы должны явно добавить ссылку на **System.Configuration.dll**.

```
using System.Configuration;
```

```
private string GetDBaseConnectionString()
{
   return ConfigurationManager.ConnectionStrings["DBConnStr"].
ConnectionString;
}
```

4. Убедитесь, что развернули файл User.config вместе с кодом приложения. Это делается в обозревателе Solution Explorer. Щелкните правой кнопкой файл User.config, выберите команду **Properties** и присвойте свойству **Copy To Output Directory** значение **Copy if newer**.

Не добавляйте файл User.config в систему управления исходным кодом. При этом каждая группа разработчиков и тестировщиков сможет использовать собственную строку подключения к БД, задав его в собственном файле User.config. Система управления исходным кодом может содержать другие файлы User.config, например, для тестирования и промышленного использования, управление которыми должно осуществляться пользователями, ответственными за соответствующие среды. Эти файлы User.config следует хранить не в составе проектов БД, а в других областях системы управления исходным кодом.

В системе управления исходным кодом должен иметься файл User.config для каждой из используемых сред, например, производственной и тестовой, с указанием строки подключения к соответствующей БД.

Совет По умолчанию при добавлении решения пользовательский файл конфигурации автоматически добавляется в систему управления исходным кодом. Чтобы отказаться от этого, во время первого возврата файла после правки сбросьте флажок напротив User.config. Чтобы файл никогда не попал в систему управления исходным кодом, щелкните его правой кнопкой в обозревателе Solution Explorer и выберите вариант **Undo Pending Changes**.

Важно понимать, что при использовании этого механизма файл User.config обязательно должен существовать. Необходим ответственный, в чьи обязанности будет входить обеспечение правильной работы среды во время создания сборок для тестовых и рабочих выпусков. Соответствующий файл User.config, входящий в сборку, должен извлекаться из системы управления исходным кодом и копироваться в положение, где его сможет найти MSBuild.

Резюме

Управлять проектами или сборками сторонних производителей можно с помощью ветвления или сопоставления рабочей области. Ветвление — предпочтительный метод, так как при этом зависимости хранятся на сервере управления исходным кодом. Использование ветвей позволяет самостоятельно принимать решение об учете обновленных двоичных или исходных кодов.

Метод сопоставления рабочей области реализуется только на клиентской стороне. Это означает, что каждый член группы должен создать сопоставление на своем компьютере, а также на сервере сборки, чтобы приложение можно было собрать. Этот метод обычно используется, когда изменения в двоичных или исходных кодах нужно немедленно учесть во время сборки.

Используйте динамические URL при создании ссылок на веб-службы. Для управления веб-службами используйте внешний файл конфигурации. Преимущество этого способа заключается в том, что каждый разработчик может задать ссылку на веб-службу в собственном файле конфигурации User.config. С помощью файла конфигурации можно также управлять ссылками на базы данных в форме строк подключения.

Дополнительные ресурсы

- **Раздел «Руководство по управлению исходным кодом»** В нем вы найдете дополнительные сведения о ссылках.
- Дополнительную информацию о ссылках на проекты вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о рабочих областях вы найдете в статье «Working with Source Control Workspaces» по адресу *http://msdn2.microsoft.com/en-us/library/ms181383(VS.80).aspx*.

ЧАСТЬ III

Сборки

В этой части

- Знакомство с Team Build.
- Настройка непрерывной интеграции при помощи Team Build.
- Настройка сборки по расписанию при помощи Team Build.

ГЛАВА 7

Знакомство с Team Build

В этой главе

- Введение в архитектуру Visual Studio Team System Team Build.
- Компоненты Team Build.
- Функциональность Team Build.
- Выбор стратегии сборки.
- Особенности стратегии групповой разработки при работе над крупным проектом.

Обзор

В этой главе объясняется, как автоматизировать процесс сборки при помощи Team Build. Описаны некоторые распространенные затруднения, и сравниваются различные подходы к сборке — от ежедневных плановых сборок до непрерывной интеграции.

Подсистема Team Build основана на системе Microsoft Build Engine (MSBuild) и позволяет извлекать исходный код для сборки, компиляции решения и (при необходимости) для запуска в процессе сборки инструментов модульного тестирования и анализа статического кода. Вы также можете поместить результат сборки на заданном общем ресурсе.

Исходный код, используемый для определенной сборки, Team Build помечает номером, чтобы в дальнейшем вы могли извлечь код, использованный для создания именно этой сборки. Если произойдет сбой, настройте Team Build на создание соответствующих рабочих элементов и на уведомление пользователей о произошедшей ошибке.

Архитектура

В этом разделе рассказывается об архитектуре Team Build и о логической последовательности рабочих операций.

Физическая архитектура

Физическая архитектура Team Build состоит из следующих компонентов:

- Мастер New Team Build Type Creation Wizard — клиентский компонент для создания новых типов сборок; доступен из Team Explorer.
- Обозреватель Team Build — клиентский компонент, позволяющий просматривать отчеты Team Build и информацию о выполнении сборки в Team Explorer.
- Веб-служба управления исходным кодом — компонент уровня приложения, используется службой сборки для доступа к исходному коду.
- Веб-служба Team Foundation Build — компонент уровня приложения, принимающий запросы от клиента и координирующий выполнение этапов сборки.
- Служба сборки (Build Service) — выполняет различные этапы сборки, получая инструкции от веб-службы Team Build. Ее можно разместить на отдельном сервере сборки или на сервере уровня приложений.
- Хранилище Team Foundation Build — компонент уровня данных, используется для хранения записей, относящихся к процессам Team Build.
- База данных исходного кода — компонент уровня данных для хранения исходного кода, доступ к которому во время выполнения процесса сборки осуществляет служба сборки.

Логика рабочего процесса

Логика процесса сборки Team Build проиллюстрирована на рис. 7-1.

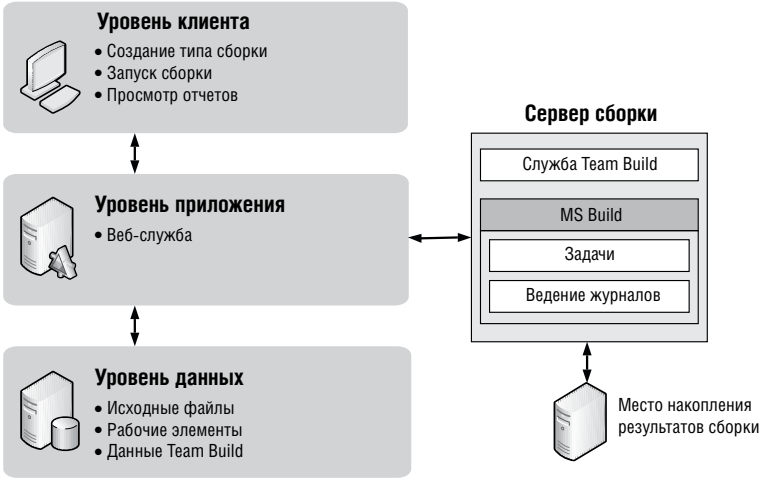


Рис. 7-1. *Логика рабочего процесса Team Build*

Подсистема Team Build интегрирована с сервером TFS на уровне приложения и взаимодействует с рабочими элементами, покрытием кода тестами, анализом кода, тестовыми данными и отчетами.

Управление сборкой осуществляется при помощи файла TFSBuild.proj. В нем определяются собираемые проекты, конфигурации, места накопления результатов, анализ кода и выполняемые тесты. Этот файл генерируется мастером Team Build Creation Wizard при создании сборки. Его можно редактировать непосредственно.

В Team Build используется система событий сервера TFS. События Team Build можно применять для создания собственных этапов сборки, а также для генерации уведомлений об изменении состояния сборки или о ее завершении.

Важные аспекты

Обратите внимание следующие ключевые моменты физической архитектуры Team Build:

- Team Foundation Build работает на основе MSBuild.
- Файл TFSBuild.proj содержит все параметры сборки. Он создается мастером Team Build Creation Wizard и допускает непосредственное редактирование.
- В файле TFSBuild.rsp содержатся параметры командной строки для MSBuild. Его также можно изменять.
- Собственные этапы сборки и уведомления создаются посредством событий **BuildStatusChangeEvent** и **BuildCompletionEvent**.
- TeamBuild интегрируется с рабочими элементами, охватом кода, анализом кода и тестовыми данными.

Принципы работы Team Build

Team Build состоит из службы Team Build Service, работающей поверх системы сборки MSBuild. Система MSBuild выполняет собственно сборку, а служба Team Build отвечает за взаимодействие с уровнем приложения TFS. Сборки Team Build создаются в клиенте Visual Studio. Разработчик может запускать их из клиента, при помощи события на сервере сборки или из командной строки, например, как запланированную задачу. Процесс сборки состоит из следующих этапов:

1. Получение исходного кода из системы управления исходным кодом и его помещение в каталог сборки.
2. Компиляция исходного кода и создание двоичных файлов.
3. Выполнение анализа кода (необязательный этап).
4. Создание рабочего элемента, если произошел сбой сборки.
5. Выполнение тестов (необязательный этап).
6. Оценка покрытия кода текстами (необязательный этап).
7. Запись сведений о сборке.
8. Копирование сборки в место накопления.

   После завершения сборки вам доступны следующие элементы:

- **Сведения о сборке** Просмотреть их можно из любого клиента или в отчетах.
- **Двоичные файлы сборки** Находятся в месте накопления.
- **Журнал сборки** Содержит сведения об ошибках и предупреждения.
- **Рабочий элемент** Если во время сборки произошел сбой, создается рабочий элемент, позволяющий отследить, где он возник.

Как определить стратегию сборки

Для определения стратегии сборки выполните следующие действия:

1. Выясните, кто будет пользоваться сборкой.
2. Просмотрите сценарии решения.
3. Выявите распространенные затруднения.

Определение потребителей сборки

Большинство команд разработчиков создают сборки для одного или нескольких типов потребителей:

- команды разработчиков;
- команды тестировщиков;
- внутренние контролеры;
- внешние контролеры бета-версии;
- заказчики.

У потребителей каждого типа свои требования к качеству сборок и частоте их выпуска. Как правило, их можно разделить на две группы: одним нужны плановые сборки, выпускаемые по расписанию, другим — сборки, оперативно создаваемые на основе происходящих событий. Плановые сборки обычно создаются ежедневно, но это может происходить как чаще, так и реже. Сборки, управляемые событиями, обычно инициируются возвратом исходного кода в систему и предназначены для быстрого предоставления команде разработчиков информации о качестве кода. Если у разработчиков возникли проблемы с неработающими сборками, попробуйте использовать модель возврата кода с контролем качества (gated check-in), в которой сборки тестируются, прежде чем разрешается их помещение в дерево исходного кода.

Сценарии сборки

Выбирайте сценарии решения, опираясь на степень их соответствия конкретной ситуации. Если вы сомневаетесь, используйте самый простой из возможных сценариев — плановую сборку — и усложняйте его только при необходимости. Ниже перечислены наиболее распространенные сценарии командных сборок:

- **Ежедневная сборка** Используется в большинстве команд для предоставления тестировщикам и другим потребителям согласованных двоичных файлов.

- **Ежедневная сборка и непрерывная интеграция** В некоторых командах для оперативного предоставления разработчикам информации о качестве кода используются сборки с непрерывной интеграцией — при каждом возврате исходного кода. Система непрерывной интеграции в чистом виде вполне подходит для небольших команд разработчиков, а вот крупные команды с частым возвратом исходного кода и продолжительной сборкой рискуют перегрузить сервер сборки. Если это случилось, попробуйте использовать скользящую сборку (rolling build). Сборки при этом создаются не так часто, но и время между возвратом кода и получением результата возрастает незначительно.
- **Несколько лабораторий сборки, в каждой из которых используются ежедневные сборки и непрерывная интеграция** В очень больших командах для повышения качества и своевременности сборки необходимы более сложные решения. Для сокращения количества сбоев можно использовать возврат кода с контролем качества, отклоняя ненадежный код, прежде чем он попадет в дерево. Команды с подразделениями могут использовать несколько серверов сборки (по одному для каждого подразделения), чтобы обеспечить концентрацию каждой группы на выполнении определенной задачи. Например, одно подразделение создает сборки интеграции, другое — сборки разработки.

Плановые сборки

Плановая сборка выполняется через определенные промежутки времени. Цель плановой сборки — создание согласованных надежных сборок, которые можно использовать для получения информации о качестве сборки. Плановые сборки обычно выполняются ежедневно, но могут выполняться реже или чаще, в зависимости от потребностей.

Чаще всего потребителями плановых сборок являются:

- тестировщики;
- внутренние контролеры сборки;
- внешние контролеры.

Для создания плановой сборки выполните следующие действия:

1. Создайте пакетный файл для выполнения сборки из командной строки.
2. Используйте планировщик Windows для выполнения пакетного файла по расписанию.

Более подробную информацию вы найдете в главе 9.

Непрерывная интеграция

При непрерывной интеграции сборка запускается каждый раз, когда происходит возврат кода после правки. Назначение такой сборки — максимально быстрое получение информации о качестве сборки. Потребителями непрерывной интеграции обычно являются команды разработчиков.

Выберите одну из следующих стратегий в зависимости от размера команды, размера сборки и частоты возврата кода:

- Сборка сразу после возврата кода.
- Скользящая сборка после указанного числа возвратов кода или через определенный период времени.

  Более подробную информацию вы найдете в главе 8.

Возврат кода с контролем качества

Возврат кода с контролем качества означает, что возвращаемый код должен удовлетворять определенным стандартам качества, чтобы его можно было добавить в дерево исходного кода. За счет выполнения ряда тестов снижается количество сбоев сборки и повышается ее качество.

Типичные проблемы

Существует несколько типичных сценариев, которые в Team Build по умолчанию не поддерживаются. Подумайте, не сталкивается ли ваша команда с какой-либо из следующих ситуаций.

- **Разработка проекта с внешними зависимостями** Если вы включаете внешние зависимости посредством ветвления, используйте ссылки проекта, чтобы сборка работала на сервере сборки без каких-либо дополнительных действий. Если для ссылок на внешние зависимости используется сопоставление рабочей области на клиентской стороне, сопоставление необходимо поддерживать на сервере сборки, чтобы она выполнялась успешно. Подробнее — в главе 6.
- **Разработка проекта установки** По умолчанию Team Build не поддерживает проекты установки. Для компиляции проекта установки и копирования двоичных файлов в место накопления используйте собственный послесборочный этап. Более подробную информацию вы найдете в статье по адресу *http://msdn2.microsoft.com/en-us/library/ms404859(VS.80).aspx*.
- **Разработка приложений .NET 1.1** По умолчанию Team Build не поддерживает приложения .NET 1.1. Сборки .NET 1.1 поддерживаются комплектом MSBuild Extras Toolkit for .NET 1.1 (MSBee), но при этом необходимо обновить проекты и решения до Visual Studio 2005. Если обновить проекты и решения нельзя, используйте для компиляции приложений .NET 1.1 собственный послесборочный этап. Скачать MSBee можно по адресу *http://www.codeplex.com/MSBee*. Подробнее о создании собственного этапа сборки для компиляции приложений .NET 1.1 — в блоге по адресу *http://blogs.msdn.com/nagarajp/archive/2005/10/26/485368.aspx* или в статье «Microsoft SDC DevEnv task» по адресу *http://www.codeplex.com/sdctasks*.
- **Разработка приложений Microsoft Visual Basic® 6.0** По умолчанию Team Build не поддерживает приложения Visual Basic 6.0. Для компиляции таких приложений используйте собственный послесборочный этап.

Подробнее — в статье «MSBuild task to build VB6» по адресу *http://freetodev.spaces.live.com/blog/cns!EC3C8F2028D842D5!261.entry*.

Особенности крупных проектов

При работе в крупных проектах следует учитывать следующие отличия больших команд:

- требуют более сложной структуры ветвления и слияния;
- часто требуют управления зависимостями между разными решениями и командными проектами;
- чаще возникает необходимость поддерживать несколько сборок для компонентов и команд разработчиков.

Работая в крупных командах разработчиков, учитывайте следующие соображения:

- В крупных командах сборка обычно занимает больше времени. Частота выполнения сборки с непрерывной интеграцией не должна превышать время сборки во избежание очередей и увеличения нагрузки на сервер сборки.
- Если в команде есть подразделения, используйте по одному серверу сборки для каждого подразделения. Это позволяет каждому подразделению сосредоточиться на выполнении определенной задачи, например, на интеграции или разработке.
- Подразделения, занимающиеся интеграцией, обычно работают только с плановыми сборками. Подразделения, занимающиеся разработкой, могут работать как с плановыми сборками, так и с непрерывной интеграцией.

Настройка сборки

Информация о сборке, например, сервер сборки, место накопления, каталог сборки, задается в файле TFSBuild.proj. Он содержит значительную часть сведений, необходимых для выполнения командной сборки, включая расположения сборки и указания на то, требуется ли выполнять статический анализ кода и модульные тесты. Чтобы изменить эту информацию, отредактируйте файл TFSBuild.proj, выполнив следующие действия:

1. Извлеките файл из системы управления исходным кодом.
2. Обновите информацию в файле.
3. Верните файл в систему управления исходным кодом.

При следующем выполнении сборки будут использованы обновленные данные. Подробнее о настройке сборки — в разделах «Настройка» руководства по сборке и практических рекомендациях по сборке в этой книге.

Резюме

Team Build основана на системе MSBuild. Она интегрируется с сервером TFS на уровне приложения и взаимодействует с рабочими элементами, покрытием кода тестами, тестовыми данными и отчетами.

Определяя стратегию сборки, необходимо учитывать требования к ней и запросы потребителей сборки. Обычно для получения согласованных сборок, необходимых тестировщикам и другим потребителям, применяют плановые сборки, а для быстрого получения информации о качестве сборки используют непрерывную интеграцию.

Дополнительные ресурсы

- Дополнительные сведения вы найдете в разделах «Как автоматически выполнять анализ кода при помощи Team Build», «Как настроить непрерывную сборку в Visual Studio Team Foundation Server», «Как настроить плановую сборку в Visual Studio Team Foundation Server».
- Дополнительную информацию о Team Build вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181710(vs.80).aspx*.

ГЛАВА 8

Настройка непрерывной интеграции

В этой главе

- Введение в непрерывную интеграцию.
- Настройка непрерывной интеграции в Microsoft® Visual Studio® Team System Team Build.
- Оптимизация непрерывной интеграции.

Обзор

В этой главе рассказывается, как настраивать непрерывную интеграцию при помощи Team Build и Microsoft Visual Studio Team Foundation Server (TFS). Непрерывная интеграция используется для оперативного получения информации о качестве сборки немедленно после возврата исправленного кода в систему. При командной разработке оперативность очень важна, особенно если исправления объединяются с изменениями, внесенными другими разработчиками, и приводят к сбою в сборке. Непрерывная интеграция позволяет быстро исправлять код, не мешая работе других членов команды. Тем самым повышаются качество и согласованность сборки.

По умолчанию Team Foundation Server не поддерживает непрерывную интеграцию. Однако ее можно добавить при помощи расширений Microsoft.

Основные сведения о возможностях автоматизации и планового выполнения сборок приводятся в главе 7.

Стратегии непрерывной интеграции

Непрерывная интеграция (continuous integration) — процесс создания сборок при каждом возврате кода в систему управления исходным кодом.

Перечислим различные стратегии непрерывной интеграции:

- сборка при каждом возврате;
- скользящая сборка после определенного числа возвратов;
- скользящая сборка по истечении заданного промежутка времени;
- скользящая сборка после определенного числа возвратов или по истечении заданного промежутка времени.

Сборка при каждом возврате

Выполнение сборки после каждого возврата кода — простейшая стратегия непрерывной интеграции, позволяющая получать информацию о качестве кода наиболее оперативным способом. Однако, если возвраты происходят слишком часто и сервер сборки не успевает их обрабатывать, следует использовать другой подход, при котором сборка выполняется после определенного числа возвратов или по истечении определенного периода времени. Чтобы принять обоснованное решение, определите следующие параметры:

- время выполнения командной сборки;
- среднее время между возвратами кода;
- интервал времени, в котором возвраты осуществляются особенно часто.

Если время, необходимое для выполнения сборки, превышает среднее время между возвратами, сборки начнут выстраиваться в очередь, поскольку одна сборка еще не будет выполнена, когда произойдет возврат, инициирующий выполнение следующей сборки. Чрезмерная длина очереди сборок может повлиять на производительность сервера и заблокировать начало других сборок, например, плановых. Выявите временное «окно», во время которого часто выполняются возвраты, и определите, достаточно ли у вас времени, чтобы очередь успела очиститься после окончания периода высокой нагрузки. Подробнее — в разделе «Как настроить непрерывную сборку в Visual Studio Team Foundation Server».

Сборка после определенного числа возвратов

Если в результате создания сборок после каждого возврата сервер оказался перегружен, попробуйте выполнять сборку через определенное число возвратов. Это самый простой сценарий скользящего выполнения сборки, но у него есть серьезный недостаток. Поскольку должно произойти определенное число возвратов, прежде чем сервер выполнит сборку, последний возврат рабочего дня почти гарантированно сборку не инициирует, а значит, информация о качестве кода будет получена с задержкой.

Сборка по истечении определенного промежутка времени

Создание сборок через определенные промежутки времени после возврата — более совершенный способ. Он позволяет гарантированно получить сборку с

небольшой задержкой относительно возврата. Учтите, что с каждой сборкой может быть связано разное число возвратов — от одного до нескольких. Чем больше возвратов, тем сложнее определить, который из них вызвал сбой.

Сборка по обоим условиям

Создание сборок по обоим условиям — после определенного числа возвратов или по истечении определенного промежутка времени — позволяет обеспечить максимальную согласованность сборок, по-прежнему не перегружая сервер. Используйте скользящую сборку, если при непрерывной интеграции возникает большая очередь сборок, обработка которой требует много времени. Используйте средний интервал между возвратами, чтобы задать число возвратов, которое должен получить сервер, прежде чем выполнить сборку. Используйте таймаут, чтобы гарантировать выполнение сборки даже в том случае, если получено небольшое число возвратов.

Чтобы определить оптимальный интервал выполнения сборки, разделите среднее время между возвратами на длительность процесса сборки. Допустим, сборка занимает 10 минут, а среднее время между возвратами — 5 минут. Установите выполнение сборки после двух возвратов и таймаут 10 минут. При этом сборка в любом случае завершится к тому времени, когда будет инициирована следующая сборка. Если сервер все равно чрезмерно перегружается, эти значения можно увеличить.

Непрерывная интеграция в TFS

По умолчанию в Team Foundation Server 2005 не включено решение непрерывной интеграции, но в нем имеется среда, позволяющая реализовать такое решение самостоятельно.

Подробнее о настройке непрерывной интеграции в TFS рассказывается в разделе «Как настроить непрерывную сборку в Visual Studio Team Foundation Server». В качестве примера там используется решение, предоставленное командой разработчиков Visual Studio Team System. Оно устанавливает веб-службу, работающую от имени учетной записи, имеющей доступ к серверу TFS. Сервер TFS способен отправлять сообщения электронной почты или вызывать веб-службу при наступлении определенного события. Предлагаемое решение непрерывной интеграции использует механизм событий для связывания веб-службы с событием **CheckinEvent**. В результате каждый раз, когда происходит возврат, веб-служба инициирует выполнение сборки.

Резюме

Используйте непрерывную интеграцию, чтобы выполнять сборку каждый раз, когда разработчик возвращает код в базу данных. Хотя Team Build по умолчанию не предоставляет решения с использованием непрерывной интеграции, можно настроить сборку и реализовать собственное решение.

В зависимости от требований к проекту непрерывная интеграция настраивается различными способами:

- выполнение сборки при каждом возврате;
- выполнение сборки после определенного числа возвратов или по истечении определенного периода времени (используется тот способ, условие которого выполнено раньше).

Дополнительные ресурсы

- Дополнительные сведения вы найдете в разделе «Как настроить непрерывную интеграцию в Visual Studio Team Foundation Server».
- Дополнительную информацию о решении с использованием непрерывной интеграции Visual Studio Team System вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx*.
- Установочный MSI-файл решения с использованием непрерывной интеграции Visual Studio Team System вы найдете по адресу *http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi*.
- Подробнее о разработке и непрерывной интеграции при помощи Team Foundation Server читайте по адресу *http://msdn.microsoft.com/msdnmag/issues/06/03/TeamSystem/default.aspx*.

ГЛАВА 9

Настройка плановой сборки

В этой главе

- Введение в плановую сборку.
- Настройка плановой сборки в Microsoft® Visual Studio® Team System Team Build.

Обзор

В этой главе объясняется, как настраивать плановое выполнение сборок в Team Build и Microsoft Visual Studio Team Foundation Server (TFS). Плановые сборки используются для автоматизации процесса создания согласованных двоичных файлов. Они чаще всего применяются группами тестирования, внутренними контролерами и внешними контролерами бета-версий.

Плановые сборки — простейшая форма автоматизации сборок. Плановая сборка может выполняться каждый час, каждый день, каждую неделю или с любым другим интервалом.

Основные сведения о возможностях автоматизации и планового выполнения сборок приводятся в главе 7. Если вас беспокоит нестабильность сборок, вызванная качеством кода, подумайте об использовании непрерывной интеграции. Подробнее — в главе 8.

Выбор частоты плановой сборки

Частот создания сборок — одно из самых важных решений, принимаемых при настройке плановых сборок. Сборки могут выполняться каждый час, каждый день или каждую неделю.

Ежечасные сборки

Если в проекте достаточно возвратов, чтобы значительные изменения происходили в течение часа, но непрерывная интеграция не используется, настройте ежечасное создание сборок. Это позволит разработчикам быстро получить информацию о качестве кода. Ежечасные сборки можно также предоставлять тестировщикам и другим членам команды.

Ежедневные сборки

Это наиболее распространенная частота сборки. При этом разработчики каждое утро получают готовую к тестированию сборку со всеми вчерашними изменениями.

Еженедельные сборки

В крупных и сложных проектах сборка может длиться днями, и тогда лучшим вариантом становится еженедельная сборка. При этом команда тестировщиков в начале каждой недели будет получать сборку, включающую все изменения, внесенные на прошлой неделе.

Плановая сборка на сервере TFS

Компонент Team Build сервера TFS не поддерживает плановое создание сборок при помощи пользовательского интерфейса. Воспользуйтесь для этого планировщиком задач Windows.

Чтобы создать плановую сборку, выполните следующие действия:

1. Создайте командную строку TFSBuild.

```
TfsBuild start <<TeamFoundationServer>> <<TeamProject>> <<BuildTypeName>>
```

2. Введите эту строку в пакетный файл. Чтобы его можно было запустить из командной строки Windows, укажите полный путь к файлу TFSBuild.exe, например:

```
"C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\TFSBuild" start
<<TeamFoundationServer>> <<TeamProject>> <<BuildTypeName>>
```

3. Создайте запланированную задачу Windows, которая будет выполнять этот пакетный файл с нужным интервалом.

Резюме

Используйте расписание для создания согласованных сборок, которые можно передавать тестировщикам и другим потребителям. Team Foundation Server не располагает пользовательским интерфейсом для планирования сборок. Для выполнения сборки в определенное время запустите утилиту командной строки TFSBuild при помощи планировщика Windows.

Плановую сборку можно выполнять ежечасно, ежедневно, еженедельно или с любым другим интервалом.

Дополнительные ресурсы

- Дополнительные сведения вы найдете в разделе «Как настроить плановую сборку в Visual Studio Team Foundation Server».
- Дополнительную информацию о настройке плановой сборки вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181727(VS.80).aspx*.

ЧАСТЬ IV

Работа в больших проектах

В этой части

- Рекомендации по работе в больших проектах.

ГЛАВА 10

Рекомендации по работе в больших проектах

В этой главе

- Логическая последовательность операций в большом проекте Microsoft® Visual Studio® Team System Team Foundation Server (TFS).
- Оптимизация системы управления версиями и сборок в больших группах.
- Работа системы управления версиями с большими проектами.
- Изменение стратегии ветвления и слияния при работе с большими проектами.
- Изменение стратегии сборки в больших проектах.

Обзор

В данной главе подробно рассматриваются принципы разработки масштабных проектов с использованием TFS. Как правило, отличие большого проекта от небольшого состоит в следующем:

- в больших проектах требуется более сложная структура ветвления и слияния;
- в больших проектах обрабатывается больше зависимостей между решениями и проектами;
- часто в большом проекте обслуживается несколько сборок для компонентов и групп.

Например, в большом проекте может понадобиться поддержка нескольких ветвей, чтобы различные группы могли параллельно разрабатывать разные функции. В этом сценарии, вероятнее всего, придется работать с зависимостями между решениями и проектами, совместно используя веб-службы и базы данных. Каждой группе, возможно, придется работать с собственным

сервером сборки и выкладывать сборки в определенное место накопления.

Эта глава адресована тем, кто участвует в управлении, поддержке и работе над крупномасштабными проектами. Рекомендации по работе с большими проектами можно также найти в главах 3, 5 и 7. В этой главе все рекомендации по работе над большими проектами сведены воедино.

Логическая последовательность операций в больших проектах

На рис. 10-1 представлена типовая схема совместной работы нескольких подгрупп над созданием сложного приложения и проиллюстрированы ключевые моменты работы в большом проекте:

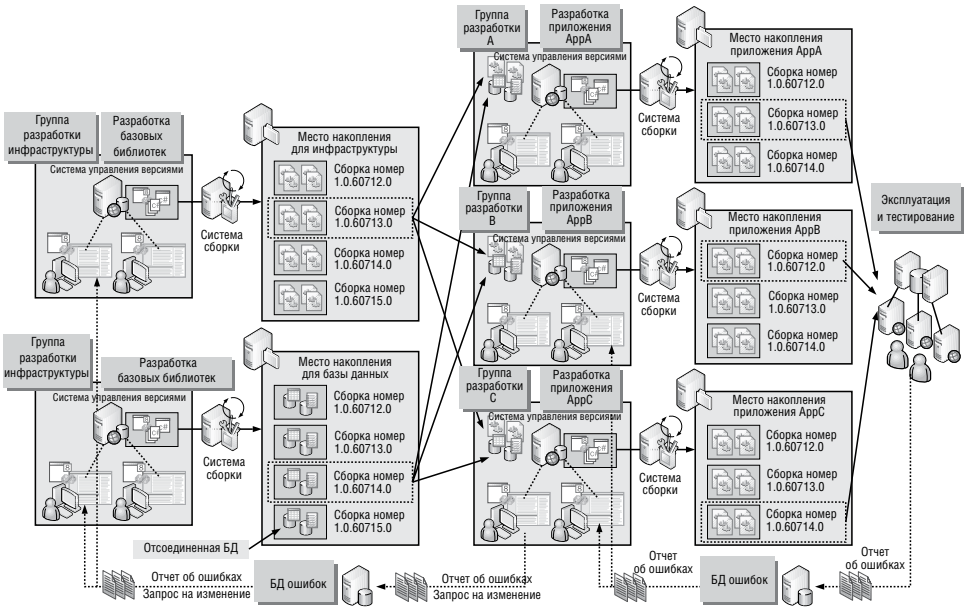


Рис. 10-1. *Сценарий работы с большим проектом*

- Каждая подгруппа использует собственный сервер сборки и выкладывает сборки в собственное место накопления.
- Группы разработки приложения извлекают сборки, предоставляемые группами разработки компонентов, и используют их уже как часть своих решений, включая в собственные сборки.
- Тестирование компонентов и интеграционное тестирование выполняются для каждой сборки. Дефекты регистрируются в соответствующей базе данных для отслеживания дефектов.

Рекомендации по управлению версиями

При работе над большим решением, включающим десятки проектов, могут возникнуть проблемы с его масштабируемостью решения (.sln) Visual Studio. В этом случае следует распределять управление версиями по подсистемам или подприложениям, используя несколько файлов решений и не создавая главного решения для всего приложения. На рис. 10-2 продемонстрирован подход с использованием нескольких решений, который обычно применяется в больших группах.

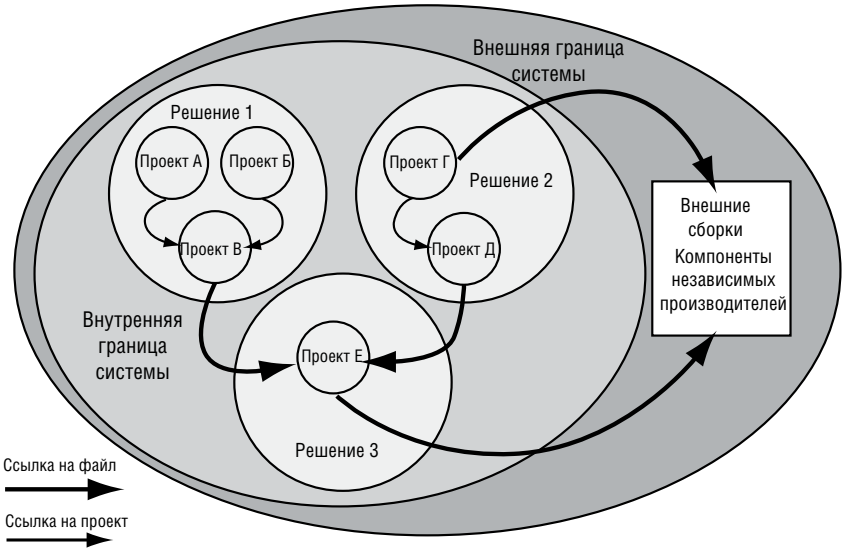


Рис. 10-2. *Подход с использованием нескольких решений*

В этом сценарии все ссылки внутри решения являются ссылками на проекты. Ссылки на проекты вне решений (например, на библиотеки сторонних производителей или проекты другого решения) являются файловыми ссылками. Это означает, что можно обойтись без «главного» решения, используя вместо него сценарий, «понимающий» порядок сборки решений.

Одна из главных задач при обслуживании структуры с несколькими решениями — обеспечить отсутствие циклических ссылок между решениями, что требует сложных сценариев сборки и явного отображения зависимостей. При такой структуре невозможно целиком собрать приложение в Visual Studio; приходится прибегать к TFS Team Build.

Подробнее о подходе с использованием нескольких решений рассказывается в главе 3.

Рекомендации по ветвлению и слиянию

В больших группах часто используется ветвление, как по группам, так и по компонентам. Кроме того, одна или несколько ветвей создаются для интеграции внешних зависимостей. Поскольку структура ветвления в больших группах более глубока, увеличивается промежуток времени между изменением в коде ветви разработки и переносом этого изменения в главную ветвь. Поэтому, создавая ветви, необходимо особенно тщательно продумывать стратегию слияния.

Выбирая между выполнением слияния по графику или под управлением событий, учитывайте следующие соображения.

- Обратная интеграция, например, перенос изменений из ветви разработки в главную ветвь, обычно выполняется по расписанию.
- Прямая интеграция, например, перенос изменений из главной ветви в ветвь разработки, обычно управляется неким событием (например, завершением этапа разработки компонента). Это событие происходит, когда группа, ответственная за ветвь разработки, считает, что готова перенести изменения из своей ветви.

Необходимо найти компромисс между стратегией ветвления и слияния и предполагаемой частотой выполнения сборок. Глубокая структура ветвления приводит к более продолжительному интегрированию дочерних ветвей в главную ветвь исходного кода. Признаки неверного выбора стратегии ветвления таковы:

- Исправления в сбойной сборке слишком долго добираются до главной ветви.
- Срываются сроки разработки, потому что на распространение внесенных изменений в главную ветвь уходит слишком много времени.
- На слияние изменений между ветвями требуется слишком много времени.

Если в вашей команде возникли эти проблемы, подумайте об уменьшении глубины структуры ветвления. Ниже приведен пример структуры ветвления в большой группе:

My Project

- **Development** — контейнер для активных ветвей разработки
  - **Team 1**
    - ▲ **Feature A** — изолированная ветвь разработки
      - ▼ **Source**
    - ▲ **Feature B** — изолированная ветвь разработки
      - ▼ **Source**
  - **Team 2**
    - ▲ **Feature A** — изолированная ветвь разработки
      - ▼ **Source**

        - **Feature B** — изолированная ветвь разработки
            - **Source**
- **Main** — главная ветвь интеграции и сборки. Здесь сводятся воедино все изменения.
    - **Source**
    - **Папки других ресурсов**
- **Releases** — контейнер для выпусков
    - **Release 4** — ветвь с кодом на этапе подготовки к выпуску, внесение изменений в который запрещено
        - **Source**
    - **Release 2** — активная ветвь обслуживания
        - **Source**
    - **Release 3** — активная ветвь обслуживания
        - **Source**
- **Архив**
    - **Release 1** – Ветвь архивного хранения старой версии
        - **Source**

В эту структуру включены:

- ветви компонентов, обеспечивающие изолированную работу нескольких групп;
- главная ветвь, в которой собираются изменения от всех групп, а также исправления ошибок из ветвей выпусков;
- ветвь версии, готовой к выпуску, используемая для блокировки текущей версии продукта;
- ветвь обслуживания старой версии, которая все еще активно поддерживается;
- ветви с архивом старых версий, которые больше не обслуживаются.

Рекомендации по выполнению сборки

Скорее всего, в больших группах и время сборки также будет большим. Интервал между сборками при использовании непрерывной интеграции должен быть меньше продолжительности одной сборки, чтобы не создавать очередь и не перегружать сервер сборки. Система сборки в большой группе обычно организована следующим образом:

- Если группа использует ветвление по группам или компонентам, с каждой ветвью связан собственный сервер сборки. Это позволяет ориентировать каждую сборку на конкретную цель ветви, например, на интеграцию или разработку.

- Периодичность сборки определяется потребностями группы. Инфраструктура сборки разработана для обеспечения этих потребностей.
- Сначала выбирается периодичность сборки, затем на ее основании определяется периодичность слияний.
- Плановые сборки обычно используются в ветвях интеграции.
- В ветвях разработки используется сочетание непрерывной интеграции и плановых сборок. Сборки, полученные в процессе непрерывной интеграции, передаются группе тестирования интеграции. Результаты плановой сборки в ветви разработки поступают к соответствующей группе тестирования, а результаты непрерывной сборки в этой же ветви позволяют группе разработки оценить качество кода.

Другой ключевой вопрос: нужно ли выполнять сборку, тестирование и контроль качества в каждой точке интеграции? Все это можно отложить до объединения всех изменений в главной ветви интеграции. Идеальный вариант: использовать для каждой ветви отдельный сервер сборки, а также пороги качества, группы разработки и тестирования. Однако в случае отсутствия одной из составляющих для упрощения структуры можно отказаться от контроля качества или удалить некоторые ветви.

Резюме

При работе с большими решениями, включающими десятки проектов, могут возникнуть проблемы с масштабируемостью решения (.sln) Visual Studio. В этом случае следует разбить структуру системы управления версиями на подсистемы или подприложения и использовать несколько файлов решения.

В больших группах структура ветвления более сложная, поэтому следует готовиться к увеличению задержки между внесением изменений в ветвь разработки и их распространением в главную ветвь.

Время сборки в больших группах обычно больше. Периодичность сборки при использовании непрерывной интеграции должна быть меньше длительности сборки. Это позволит избежать больших очередей сборок и перегрузки сервера сборки. Если проблемы с производительностью сервера сборки все равно сохраняются, рекомендуется использовать отдельный сервер сборки для каждой ветви системы управления версиями.

Дополнительные ресурсы

- Подробнее о системе управления версиями Team Foundation читайте в статье «Using Source Code Control in Team Foundation» по адресу *http://msdn2.microsoft.com/en-us/library/ms364074(VS.80).aspx*.
- Подробную информацию о создании рабочей области вы найдете в статье «How To - Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.

- Введение в ветвление и слияние вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Подробнее о ветвлении читайте в статье «How To - Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Подробнее о слиянии читайте в статье «How To - Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробное описание методов ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.
- Подробнее о Team Build читайте в статье «Overview of Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms181710(vs.80).aspx*.

ЧАСТЬ V

Управление проектами

В этой части

- Введение в управление проектами.
- Рабочие элементы.

ГЛАВА 11

Введение в управление проектами

В этой главе

- Возможности Microsoft® Visual Studio® Team System по управлению проектами.
- Стратегия создания командного проекта.
- Создание и управление проектами в Visual Studio Team System.

Обзор

Эта глава познакомит вас с возможностями управления проектами из арсенала Visual Studio Team System. Вы узнаете, как с их помощью решать типичные проблемы и задачи, возникающие при управлении проектами по разработке ПО.

В Visual Studio Team System и Team Foundation Server (TFS) включены инструменты, шаблоны и отчеты, помогающие отслеживать и поддерживать процессы разработки ПО. Они упрощают обмен информацией в рамках команды, автоматизируют передачу необходимых данных различным членам команды, упрощают распределение и отслеживание рабочих элементов, например, задач, облегчают контроль за показателями состояния проекта.

Введение в управление проектами

Планирование проектов обычно выполняется по следующей схеме:

1. **Концептуальное описание проекта** На этом этапе формируется представление о том, какой конечный результат проекта хотят получить все заинтересованные стороны.

2. **Формулирование сценариев** На этом этапе с участием заказчика формулируется начальный набор сценариев использования ПО. Заказчик оценивает готовые сценарии с позиции собственных интересов.
3. **Формирование набора функциональных возможностей для реализации выбранных сценариев** На этом этапе сценарии разбиваются на отдельные компоненты, важные с точки зрения заказчика. Благодаря этому появляется возможность обсуждать ожидания заказчика в отношении этих компонентов.
4. **Формирование набора рабочих элементов** Сценарии и компоненты разбиваются на конкретные задачи. Завершение рабочего элемента означает реализацию соответствующей функции или сценария.
5. **Распределение задач по областям** Области выделяются на основе компонентов или в зависимости от организации конкретной команды.
6. **Создание плана работ** На этом этапе либо составляется полный график выполнения работ, либо производится разбиение работы на итерации.

Типичные проблемы управления проектами

Сегодня в распоряжении руководителя проекта имеется множество разнообразных инструментов для управления процессом разработки ПО. Как правило, эти инструменты практически никак не интегрированы с инструментами, используемыми для разработки ПО. Отсутствие интеграции и взаимодействия между инструментарием и командами разработчиков — главная проблема руководителя проекта. Обычно ему приходится сталкиваться со следующими трудностями:

- **Работа с разрозненными источниками информации** Инструменты для управления проектами обычно используются изолированно, что приводит к появлению разрозненных источников информации, которые не так просто объединить. Кроме того, обычно бывает сложно объединить данные по управлению проектом с данными, предоставляемыми другими членами команды, например, учесть при формировании показателей ошибки, выявленные изолированной системой отслеживания ошибок.
- **Сложности со сбором показателей проекта** Показатели проекта очень важны для контроля за его состоянием, принятия обоснованных решений и получения ответа на фундаментальные вопросы, например, «Будет ли проект поставлен вовремя и будет ли выполнен бюджет?». Отвечая на ключевые вопросы, руководитель проекта обычно полагается на данные Microsoft Office Project или системы отслеживания ошибок, используемой группами разработки и тестирования. Объединить данные, предоставляемые этими несопоставимыми системами, сложно, на это уходит много времени, к тому же, в процессе сведения данных легко допустить ошибки. Для большинства показателей, формируемых инструментарием,

не существует унифицированной системы хранения и доступа. Создание отчетов — это обычно напряженный ручной труд, связанный с многократным копированием и вставкой информации из разных источников.

- **Сложности с выполнением требований** Зачастую работа, запланированная для группы разработки, требования заказчика и основные нефункциональные требования к создаваемой системе существенно отличаются друг от друга. Еще одна «пропасть» разделяет запланированный и фактический объемы. Из-за этих несоответствий теряются важные данные, что приводит к невыполнению требований.
- **Управление процессами и изменения в них** Объяснить команде, каких процессов она должна придерживаться, очень сложно. Еще сложнее, не снизив производительности команды, внести изменения в процессы для решения возникающих проблем.
- **Недостаток учитываемого обмена информацией и отслеживания задач** Взаимодействие и слаженность команды обычно обеспечиваются при помощи совещаний и списков задач, позволяющих правильно выбрать приоритеты. Отслеживать процесс выполнения отдельной задачи может быть очень сложно. Также руководители проектов часто тратят массу ценного времени на извлечение информации о состоянии работ из разных планов и списков. Члены команды тратят время на отчеты о состоянии и обновление различных документов и форм.
- **Контроль качества** Предсказать количество и серьезность ошибок в создаваемом ПО сложно, поэтому обычно плановые оценки и затраты основываются на «оптимистических предположениях». Эти оценки традиционно делаются с запасом, величина которого зависит от степени уверенности руководителя в текущем состоянии проекта.

Система VSTS призвана помочь в решении многих традиционных проблем, с которыми сталкиваются руководители проектов. Интегрированные в нее инструменты помогают командам совершенствовать процесс разработки ПО, а руководителям проектов — лучше контролировать этот процесс.

Функции управления проектами в Team Foundation Server

Основные функции Visual Studio Team System по управлению проектами таковы:

- **Управление процессами** Система управления процессами Team Foundation Server включает руководство по процессу Microsoft Solution Framework (MSF), а также шаблоны процессов, благодаря которым новые командные проекты обеспечены типами рабочих элементов, отчетами, порталом проекта SharePoint и настройками системы управления исходным кодом.

- **Безопасность и разрешения** В новые проекты включены стандартные группы и разрешения, соответствующие типичным ролям команды разработки.
- **Централизованное управление рабочими элементами** Рабочие элементы, в том числе ошибки, риски, задачи, сценарии и требования к качеству обслуживания (quality of service, QoS), централизованно регистрируются, управляются и обслуживаются в БД рабочих элементов TFS. Централизованное хранение упрощает доступ и просмотр этих элементов всеми членами команды.
- **Интеграция с Microsoft Office Excel® и Microsoft Office Project** Благодаря интеграции с Office Excel и Office Project руководители проектов могут работать с хранилищем рабочих элементов и расписанием, используя хорошо знакомые инструменты.
- **Показатели и составление отчетов** В TFS включена возможность создания и отображения отчетов, которые превращают рабочие данные, например, рабочие элементы, результаты сборки и результаты тестирования, в показатели, хранящиеся в хранилище данных TFS. Благодаря встроенным отчетам у вас есть возможность запрашивать разнообразные показатели состояния и качества проекта.
- **Порталы проекта** Для каждого командного проекта TFS создает ассоциированный портал Microsoft Windows SharePoint®. Портал используется для управления документацией проекта, быстрого просмотра ключевых отчетов и оценки текущего состояния проекта.

Преимущества

Управляя проектами из TFS, вы получаете следующие преимущества:

- централизованное управление;
- возможность отслеживать взаимосвязи рабочих элементов;
- интегрированное планирование и составление графика проекта;
- расширенные возможности управления процессами;
- расширенные возможности обмена информацией и взаимодействия внутри команды;
- подробные отчеты о ходе выполнения работ.

Создание проектов и управление ими в Team Foundation Server

В целом, создание проекта в Team Foundation Server разделяется на следующие этапы:

1. **Выбор шаблона процесса** Можно использовать стандартный шаблон или создать собственный.

2. **Создание командного проекта** В основу проекта ляжет выбранный шаблон.
3. **Добавление в командный проект групп и членов**.
4. **Разработка итерационного цикла проекта** Включает создание итераций.
5. **Преобразование сценариев в рабочие элементы TFS**.
6. **Выбор сценариев, которые должны быть завершены в каждой итерации**.
7. **Определение требований QoS** Связывание требований со сценариями.
8. **Разбиение сценариев с целью обеспечить разработчиков управляемыми рабочими элементами** Разработчики предварительно оценивают сроки выполнения для каждого рабочего элемента.
9. **Создание графика проекта** График создается средствами Microsoft Project и добавляется в Team Project.
10. **Определение критериев приемки** Критерии завершенности рабочих элементов согласовываются с требованиями QoS.
11. **Определение требований к отчетам** Определяются ключевые показатели выполнения проекта и оговариваются требования к отчетам.

Подробнее о создании и управлении проектом рассказывается в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.

Стратегии командных проектов

Чтобы начать работать с TFS, необходим, по крайней мере, один командный проект. Когда руководитель проекта создает новый проект, одновременно создается веб-сайт проекта с библиотеками документов, в которых содержатся шаблоны документов и отчеты. Также создается база данных рабочих элементов для отслеживания всех работ по проекту. Устанавливается шаблон методологии, определяющий правила, политики, группы доступа и запросы для всех работ по проекту. В системе управления исходным кодом создается ветвь исходного кода.

Выбирая структуру командного проекта, необходимо руководствоваться требованиями к ПО. Структура проекта может меняться в процессе работы. Существует три основных стратегии создания нового проекта, которые могут использоваться по отдельности или в сочетании друг с другом:

- **один проект на приложение;**
- **один проект на выпускаемую версию;**
- **один проект на команду.**

Один проект на приложение

Это самая распространенная стратегия создания командных проектов. Такой подход уместен при разработке как крупных, так и небольших приложений, а также для параллельной разработки нескольких версий одного приложения. При таком подходе для каждого приложения, находящегося в разработке, создается по одному проекту.

Один проект на выпускаемую версию

Этот подход удобен для больших команд, работающих над долгосрочными проектами. Новый проект создается после выпуска каждой новой версии, и все начинается с нуля. При этом не приходится беспокоиться о переносе ресурсов предыдущего проекта, включая рабочие элементы. Кроме того, этот подход позволяет улучшить шаблоны процессов или использовать новые на основании приобретенного опыта и знаний.

Один проект на команду

Этот подход удобен в больших проектах, над которыми работают несколько команд, там, где особенно важны централизованное управление и мониторинг процесса разработки. При таком подходе отдельный проект создается для каждой команды. Каждая команда соотносится с последовательностями операций, которые определяются типами рабочих элементов TFS, и располагает единым блоком для создания отчетов.

Управление процессами

Благодаря VSTS управление циклом разработки ПО стало неотъемлемой частью инструментария для его разработки. Благодаря тому что процессы цикла разработки интегрированы в VSTS, обмен рабочими элементами между членами команды можно в значительной степени автоматизировать.

Шаблоны процессов

В VSTS для определения набора инструкций и артефактов, например, руководств по процессу, шаблонов документов, стандартных рабочих элементов и т. д., используются шаблоны процессов. Шаблон процесса — это независимый набор инструкций, описывающих методологию разработки ПО для команд разработчиков. В шаблон процесса включаются следующие элементы:

- **Руководство по процессу** Предоставляется для каждого шаблона и содержит контекстно-зависимую информацию, справку и указания членам команды, нуждающимся в помощи и понимании определенной деятельности. Руководство по выполнению процесса интегрировано в Visual Studio Help System.

- **Шаблоны документов** Позволяют членам команды единообразно создавать новые документы (спецификации, оценки рисков и планы проектов).
- **Рабочие элементы и последовательность операций** У рабочих элементов имеется собственный набор полей и правил, определяющих последовательность операций рабочего элемента и распределение работ между членами команды.
- **Группы безопасности** Используются для определения прав по управлению и изменению отчетов, результатов работы, например, исходного кода и документации, и рабочих элементов. Администрировать группы безопасности проекта может его руководитель, для этого ему не обязательно быть администратором Windows.
- **Политики возврата после правки** Используются для применения правил и порогов качества ко всему коду, возвращаемому в систему управления исходным кодом. Например, можно поставить условие, что весь возвращаемый код должен отвечать определенному критерию, скажем, соответствовать корпоративным стандартам написания кода, или должен подвергаться модульному тестированию. Более подробно о создании и настройке политик возврата изменений рассказывается в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- **Отчеты** Используются для мониторинга исполнения процессов и текущего состояния проекта. В VSTS встроено множеством отчетов, включая отчеты о качестве кода, о соблюдении графика работ, об эффективности тестирования и др. Можно создавать собственные отчеты и настраивать существующие.

Шаблоны процессов MSF Agile и MSF CMMI

В комплекте VSTS поставляются два шаблона процессов:

- **MSF for Agile Software Development** Простой шаблон для небольших или неформальных проектов по разработке ПО. Он основывается на сценариях и действиях по обстоятельствам. Ориентирован на конкретный проект и его исполнителей.
- **MSF for CMMI® Process Improvement** Предназначен для более серьезных проектов по разработке ПО. Расширяет функциональность шаблона MSF Agile, предоставляя поддержку аудита, верификации и формальных процессов, опираясь на процесс и соответствие процессу. Ориентирован на организацию.

Если предоставляемые шаблоны недостаточно отражают требования конкретного процесса и методологии, добавьте в систему новые шаблоны или настройте стандартные. Подробнее о настройке шаблонов рассказывается в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.

Безопасность и разрешения

При создании проекта в TFS в нем независимо от выбранного шаблона создаются четыре группы, каждая из которых по умолчанию располагает теми или иными правами, определяющими, что разрешается делать членам этой группы:

- Project Administrator — администратор проекта;
- Contributor — участник;
- Reader — читатель;
- Build Services — сервисы сборки.

Чтобы максимально удовлетворить требованиям по безопасности конкретной организации, в проекте можно создавать группы доступа самостоятельно. Это эффективный способ предоставить группе пользователей в командном проекте конкретный набор полномочий. Всегда давайте группе лишь минимально необходимые права и вводите в нее только тех пользователей или другие группы, которым эти права необходимы.

Создав группу, вы добавляете ее в список, предоставляете ей соответствующие права и включаете в нее членов. По умолчанию вновь созданная группа командного проекта не получает никаких прав.

Управление рабочими элементами

Рабочие элементы используются в качестве блоков работы для обмена информацией и координации совместной деятельности в команде. Выбранный шаблон процесса предоставляет исходный набор типов рабочих элементов. Руководители проекта создают и организуют дополнительные рабочие элементы, которые необходимы в проекте. Рабочий элемент может описывать задачу, риск, сценарий, ошибку или требование QoS. Для вящей управляемости можно устанавливать связи между рабочими элементами, например, связать определенную задачу с соответствующим ей сценарием или требованием QoS.

В шаблоне процесса имеются описания типов рабочих элементов, включая набор полей для каждого типа. Правильный выбор шаблона очень важен, поскольку в ходе выполнения проекта изменить его уже нельзя. При необходимости шаблон процесса можно дополнительно настроить, включив в него все необходимые типы рабочих элементов, отсутствующие в базовом шаблоне.

В шаблонах MSF Agile и MSF CMMI предусмотрен стандартный набор рабочих элементов с задачами, достаточными для начала процесса разработки ПО.

Шаблон MSF Agile

В этом шаблоне имеются следующие типы рабочих элементов:

- **Сценарий** (scenario) Используется для представления взаимодействия пользователя с приложением. Описывает конкретные шаги, необходимые для достижения цели. Сценарии должны быть конкретными, поскольку возможных способов действия может быть несколько.
- **Задача** (task) Используется для представления блока работы. У каждой роли свои требования к задачам. Например, разработчик использует для распределения работ задачи разработки.
- **Требование QoS** (Quality of Service requirement) Документирует характеристики системы, например, производительность, нагрузку, доступность, устойчивость к нештатным условиям эксплуатации, специальные возможности и удобство обслуживания.
- **Ошибка** (bug) Используется для информирования о потенциальной проблеме в системе.
- **Риск** (risk) Используется для выявления и управления рисками в проекте.

Шаблон MSF CMMI

В этом шаблоне имеются следующие типы рабочих элементов:

- **Требование** (requirement) Фиксирует требования, определенные на этапе сбора требований.
- **Запрос на изменение** (change request) Фиксирует любые запросы на внесение изменений, возникающие после этапа сбора требований.
- **Проблема** (issue) Проблемы, исправление которых необходимо отслеживать.
- **Задача** (task) Используется для представления блока работы. У каждой роли свои требования к задачам. Например, разработчик использует для распределения работ задачи разработки.
- **Рецензия** (review) Блок работы по составлению отзывов, например, рецензии исходного кода, проекта и пр.
- **Ошибка** (bug) Используется для информирования о потенциальной проблеме в системе.
- **Риск** (risk) Используется для выявления и управления рисками в проекте.

Интеграция с Microsoft Project

В VSTS и приложении Team Explorer имеются расширения для Microsoft Project. В крупных проектах, где задействовано большое количество ресурсов, для работы с графиком работ по проекту в TFS можно использовать Microsoft Office Project. Можно, например, управлять и планировать работы, назначать их, распределять и отслеживать, а затем, когда результаты го-

товы к использованию другими членами команды, публиковать их в базе данных рабочих элементов.

Подробнее об этом — в статье «Working with Work Items in Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms244368(VS.80).aspx*.

Интеграция с Microsoft Excel

В VSTS и приложении Team Explorer имеются расширения для Microsoft Excel. В крупных проектах, где задействовано большое количество рабочих элементов, можно использовать интеграцию с Excel: создавать рабочие элементы в электронной таблице Excel и загружать их в базу данных рабочих элементов для использования другими членами команды.

Подробнее — в статье «Working with Work Item Lists in Microsoft Excel» по адресу *http://msdn2.microsoft.com/en-us/library/ms181694(VS.80).aspx*.

Контроль выполнения работ и отчеты

Отчеты из комплекта TFS помогут вам оперативно оценить состояние проекта, качество разрабатываемого ПО и этап, на котором находится проект. Отчеты формируются на основе данных из хранилища TFS и объединяют показатели, поступающие от рабочих элементов, системы управления исходным кодом, тестирования и сборок.

Например, с помощью отчетов можно выяснить темпы работы команды на основании ее фактической деятельности. Задача системы создания и отображения отчетов TFS — предоставить вам интегрированные данные по всем компонентам VSTS, чтобы руководители проекта и члены команды могли оценить состояние проекта и предпринять соответствующие действия для его успешного выполнения.

Отчеты, доступные по умолчанию, определяются используемым шаблоном проекта, но существует также возможность создания собственных отчетов. Содержимое и использование каждого отчета из шаблона процесса, объясняется в руководстве для этого шаблона. Team Foundation Server основан на Microsoft SQL Server™ 2005 и использует SQL Server для хранения всех данных, связанных с рабочими элементами, атрибутами качества, тестированием, результатами тестирования и результатами сборок. Для объединения и анализа этих данных и создания отчетов в TFS используются службы SQL Server Analysis Services. Отчеты, созданные шаблоном процесса или отдельными членами команды с помощью Microsoft Office Excel или Visual Studio 2005 Report Designer, доступны при помощи SQL Server 2005 Reporting Services и портала SharePoint команды.

Подробнее о настройке отчетов — в разделе «Как создать собственный отчет в Visual Studio 2005 Team Foundation Server» этой книги.

Резюме

Team Foundation Server предоставляет такие возможности управления проектами, как централизованное управление рабочими элементами, управление процессами, управление безопасностью и разрешениями, сбор показателей проекта и составление отчетов. Все это упрощает управление проектами по разработке ПО средствами Visual Studio.

Управление циклом разработки ПО стало неотъемлемой частью инструментария для разработки ПО. В TFS включены шаблоны процессов MSF Agile и MSF CMMI, поддерживающие две разные методики разработки. Можно изменять предоставляемые шаблоны процессов или создать собственный шаблон, удовлетворяющий потребностям конкретной команды.

Дополнительные ресурсы

- Подробнее об управлении и организации проектов по разработке ПО с использованием VSTS читайте в статье «Visual Studio 2005 Team System: Software Project Management» по адресу *http://msdn2.microsoft.com/en-us/library/aa302181.aspx*.
- Более подробную информацию об использовании Microsoft Office Excel для управления проектами вы найдете в статье «Working with Work Item Lists in Microsoft Excel» по адресу *http://msdn2.microsoft.com/en-us/library/ms181694(VS.80).aspx*.
- Более подробную информацию об использовании Microsoft Office Project для управления проектами вы найдете статье «Working with Work Items in Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms244368(VS.80).aspx*.

ГЛАВА 12

Рабочие элементы

В этой главе

- Назначение и структура рабочих элементов.
- Последовательность операций рабочего элемента.
- Настройка рабочих элементов под конкретные требования команды.

Обзор

Эта глава познакомит вас с рабочими элементами и объяснит, как использовать их для управления проектами по разработке ПО. Каждый рабочий элемент представляет единицу работы, которую должна выполнить группа разработки. Набор типов рабочих элементов определен в шаблоне процесса, выбираемом при создании нового проекта.

Для отслеживания работ в текущем проекте можно создавать рабочие элементы любого из доступных типов. Стандартные типы рабочих элементов и их поведение определены в шаблонах процессов. Любой аспект типа рабочего элемента можно настроить согласно конкретным потребностям команды.

Сценарии и решения

Рабочие элементы (work item) — основной инструмент руководителя проекта и ведущих разработчиков для отслеживания работ по проекту: что уже сделано и что еще предстоит сделать. Члены команды используют рабочие элементы для отслеживания собственной последовательности задач и для распределения работ, например, в форме ошибок или задач.

Обычно рабочие элементы в проектах используются в следующих целях:

- Формирование требований пользователей или требований QoS к приложению.

- Контроль соответствия процессов разработки и тестирования этим требованиям.
- Создание задач разработки, представляющих действия, которые необходимо выполнить для реализации компонентов и функций приложения.
- Создание ошибок для представления дефектов в реализации компонентов и функций приложения.
- Сортировка ошибок и задач для расстановки приоритетов и распределения между членами команды.
- Отслеживание задач разработки для оценки темпов продвижения к завершению кода.
- Отслеживание ошибок и других показателей качества для определения качества приложения и его готовности к поставке.

Использование рабочих элементов в проекте определяется тем, какие типы рабочих элементов заданы в проекте. Описания рабочих элементов хранятся в шаблоне процесса, выбранном при создании проекта. Можно выбрать один из двух стандартных шаблонов — MSF Agile или MSF CMMI — или настроить рабочие элементы соответственно конкретным требованиям. Типы рабочих элементов в каждом шаблоне перечислены в предыдущей главе.

Структура рабочего элемента

Каждый тип рабочего элемента можно описать следующим образом:

- Он имеет назначение и предполагаемое использование. Например, ошибки используются для отслеживания дефектов качества, задачи — для отслеживания запланированных работ, требования QoS — для описания аспектов, не связанных с функциональностью, например, требований к безопасности, производительности и т. д.
- С ним связана последовательность операций, описываемая посредством состояний и переходов. Например, возможны состояния «Opened», «Resolved» и «Closed».
- У него есть набор полей, например, поля Priority (приоритет), Status (состояние) и Iteration (итерация), которые можно задавать, опрашивать и использовать при создании отчетов.

Последовательность операций рабочего элемента

С каждым рабочим элементом связана определенная последовательность операций (workflow), представляющая все возможные состояния рабочего элемента, а также переходы между ними. Каждое состояние обычно ассоциируется с ролью TFS. Например, когда тестировщик открывает в MSF Agile новую ошибку, соответствующий рабочий элемент переходит в состояние

Active. Когда разработчик исправляет ошибку, состояние рабочего элемента меняется на **Resolved**. Когда тестировщик подтверждает исправление ошибки, состояние меняется на **Closed**.

Примеры последовательностей операций

Далее приведены примеры последовательностей операций для двух самых распространенных типов рабочих элементов.

Задача MSF CMMI

Задача MSF CMMI может находиться в следующих состояниях:

- **Proposed** Предложена разработчиком, тестировщиком или архитектором.
- **Active** Принята ведущим разработчиком или руководителем.
- **Resolved** Решена разработчиком.
- **Closed** Протестирована и закрыта тестировщиком.

На рис. 12-1 показаны состояния и возможные переходы между ними.

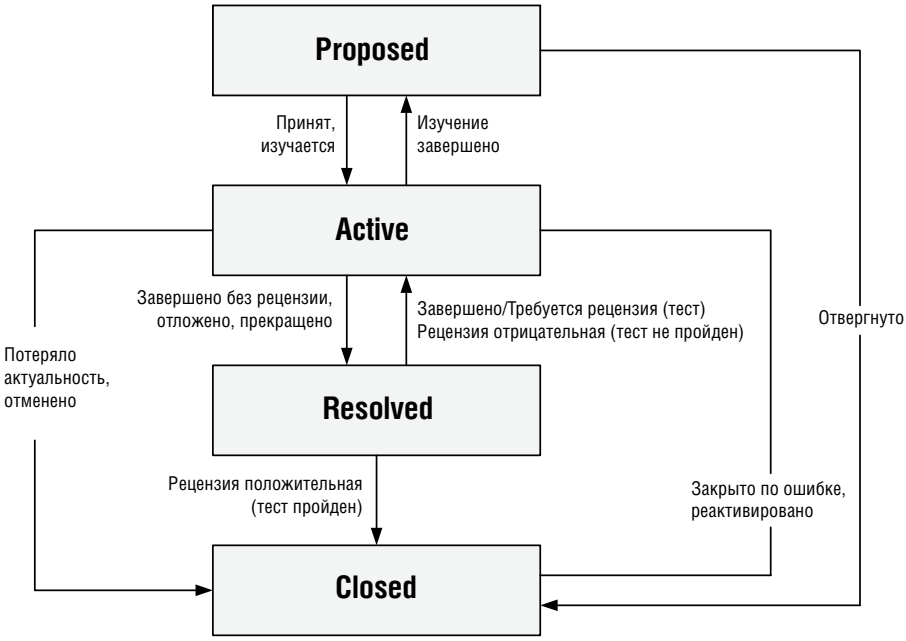


Рис. 12-1. *Переходы между состояниями рабочего элемента MSF CMMI*

Ошибка MSF Agile

Ошибка MSF Agile может находиться в следующих состояниях:

- **Active** Открыта тестировщиком.
- **Resolved** Устранена разработчиком.

- **Closed** Протестирована и закрыта тестировщиком.

На рис. 12-2 показаны состояния и возможные переходы между ними.

Рис. 12-2. *Переходы между состояниями рабочего элемента MSF Agile*

Настройка рабочих элементов

Существует несколько ситуаций, в которых может потребоваться изменение типов рабочих элементов MSF Agile или MSF CMMI:

- В рабочем элементе нет поля, которое необходимо в процессе разработки.
- Последовательность операций рабочего элемента не соответствует схеме работы команды.
- Необходим новый тип рабочего элемента.

Для разрешения этих проблем в TFS можно сделать следующее:

- Добавить (или удалить) тип рабочих элементов.
- Изменить поля существующих рабочих элементов.
- Изменить состояния и переходы существующих рабочих элементов.

Подробнее о настройке рабочих элементов — в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.

Резюме

С помощью рабочих элементов руководители проектов и команд отслеживают работы, которые должны быть выполнены в ходе проекта. Рабочие элементы используются для создания задач разработки, ошибок реализации, требований пользователей и требований QoS. Кроме того, их можно использовать для контроля за выполнением требований и определения качества приложения и его готовности к поставке.

В шаблоны процессов MSF Agile и MSF CMMI включен набор стандартных типов рабочих элементов. Имеется также возможность настроить предлагаемые или создать новые типы рабочих элементов для удовлетворения требований текущего процесса.

Дополнительные ресурсы

- Подробную информацию о рабочих элементах вы найдете в статье «Managing Team Foundation Work Items» по адресу *http://msdn2.microsoft.com/en-us/library/ms181314(VS.80).aspx*.
- Скачать шаблон процесса MSF CMMI и ознакомиться с предлагаемыми в нем типами рабочих элементов можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=12A8D806-BB98-4EB4-BF6B-FB5B266171EB&displaylang=en*.
- Скачать шаблон процесса MSF Agile и ознакомиться с предлагаемыми в нем типами рабочих элементов можно по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=EA75784E-3A3F-48FB-824E-828BF593C34D&displaylang=en*.

ЧАСТЬ VI

Шаблоны процессов

В этой части

- Знакомство с шаблонами процессов.
- Проекты MSF для быстрой разработки ПО.

ГЛАВА 13

Знакомство с шаблонами процессов

В этой главе

- Назначение, содержимое и структура шаблона процесса.
- Основные отличия между шаблонами процессов MSF Agile и MSF CMMI.
- Настройка шаблона процесса в соответствии с потребностями вашей команды.

Обзор

В этой главе обсуждается роль, которую шаблоны процессов играют в Microsoft Visual Studio® 2005 Team Foundation Server (TFS). В ней определены основные признаки и основные отличия двух поставляемых шаблонов: MSF Agile and MSF CMMI.

Процесс разработки ПО весьма сложен и включает в себя многочисленные операции на самых разнообразных уровнях. Перед группой разработчиков этот процесс предстает, как правило, в виде документации, но инструменты для реализации отсутствуют. Недостаток инструментария существенно усложняет изучение проекта группой разработчиков и его согласованное выполнение. Конечно, к услугам менеджера проекта различные средства для управления проектом, управления требованиями, отслеживания ошибок или управления обзорами, однако они редко хорошо взаимосвязаны. Несогласованность делает еще более трудной задачей соблюдение единой методологии в нескольких проектах. Усложняется также подготовка общего отчета, который позволил бы всей команде уяснить текущее состоя-

ние и успешность проекта. В результате анализ процесса становится ненадежным, а внесение улучшений в процесс — практически невозможным.

Visual Studio Team System (VSTS) и TFS предоставляют вам интегрированную среду, в которой поддерживается большинство операций, включаемых в процесс разработки ПО. В TFS методологии цикла разработки реализованы в виде шаблонов процессов. Шаблон процесса (process template) — это набор XML-файлов со спецификациями процессов и артефактов, составляющих методологию разработки. В этой главе объяснена архитектура шаблона процесса и ее компоненты. Прочитав ее, вы глубже разберетесь в использовании и настройке поставляемых шаблонов.

Если имеющиеся шаблоны не совсем соответствуют процессу разработки, принятому в вашей команде, создайте новый шаблон, отредактируйте существующий шаблон или выберите один из шаблонов, предлагаемых Microsoft Partners. Обзор последний вы найдете по адресу *http://msdn2.microsoft.com/en-us/teamsystem/aa718801*.

Шаблоны процессов MSF Agile и MSF CMMI

В комплект Team Foundation Server входят два шаблона процессов — MSF Agile и MSF CMMI, — описывающих два различных стиля разработки ПО. Используйте шаблон MSF Agile, если вы должны быстро разработать новое приложение. Он поможет реализовать MSF Agile разработку методом проб и ошибок и другие методики быстрой разработки. Опирайтесь на MSF CMMI, если следуете методологии Capability Maturity Model® Integration, предложенной Институтом разработки ПО (Software Engineering Institute, SEI). Это формальный процесс, направленный на усовершенствование существующих процессов.

Возможности двух этих шаблонов различны. Например, в них по умолчанию создаются разные типы отчетов и рабочих элементов. Оба шаблона допускают простую настройку согласно потребностям вашего проекта.

Руководство по настройке процессов

Создаваемый вами проект вовсе необязательно впишется в шаблоны процессов из комплекта VSTS: начиная с отсутствия нужного типа рабочего элемента и заканчивая совершенно иной методологией процессов. Допустим, если вы используете методологию SCRUM, а в текущем шаблоне процесса не упоминаются спринты (sprint), вам придется дополнить существующий шаблон или даже заменить его.

Архитектура шаблона процесса

В архитектуре шаблона процесса три основных компонента:

- Надстройки шаблона процесса.
- XML-файлы определения процессов.
- Мастер **New Team Project.**

Надстройки шаблона процесса

Надстройками шаблона процесса (process template plug-in) называются компоненты, запускаемые при создании нового проекта команды. Надстройка настраивает нужные файлы и конфигурирует данные в определенной области шаблона. В комплект TFS входят следующие надстройки:

- **Classification** Определяет начальную итерацию и области.
- **Groups and Permissions** Определяет начальные группы безопасности команды проекта и их разрешения.
- **Windows SharePoint Services** Определяет портал проекта на базе шаблона сайта Microsoft Windows SharePoint®, а также файлы шаблонов и руководство процесса.
- **Work Item Tracking** Определяет начальные типы рабочих элементов, запросы и экземпляры рабочих элементов.
- **Reports** Определяет начальные отчеты и настраивает сайт отчетов.
- **Version Control** Определяет начальные разрешения безопасности по управлению версиями и заметки для возврата после правки.

Для настройки шаблона процесса вы можете отредактировать любой файл определения надстройки. При необходимости в процессе настройки шаблона файлы определения надстроек можно даже удалять, за исключением надстройки Classification.

XML-файлы определения процессов

XML-файлы *определения процесса* — это набор XML-файлов с описаниями задач, которые необходимо выполнить, чтобы правильно настроить новый проект для процесса. Если вы создаете проект при помощи мастера **New Team Project**, он сам запустит все необходимые надстройки. Каждая надстройка считывает из соответствующего XML-файла определения процесса список задач, которые она должна выполнить. При помощи XML-файлов определения процесса вы задаете настройки и параметры, которые должны применяться надстройками. Доступны следующие XML-файлы:

- **XML-файл отслеживания рабочих элементов** Файл Workitems.xml хранится в подпапке Work Item Tracking иерархии папок шаблона процесса. В нем задаются типы рабочих элементов, запросы рабочих элементов и экземпляры рабочих элементов.

- **Work Item Types** Определяют правила, поля, состояния и переходы для рабочего элемента (например, задача, ошибка, требование), который предполагается отслеживать в ходе выполнения проекта.
- **Work Item Queries** Используются для поиска конкретных групп рабочих элементов, например, задач или активных ошибок. Запросы рабочих элементов задаются в файлах WIQ (work item query) в подпапке Queries папки Work Item Tracking иерархии папок шаблона процесса.
- **Work Item Instances** Начальный набор экземпляров рабочих элементов, создаваемый по умолчанию одновременно с проектом.

- **XML-файл классификации** Файл Classification.xml хранится в подпапке Classification иерархии папок шаблона процесса. Он состоит из двух частей: итерации и области.
  - **Iterations** Здесь определяется, сколько раз команда проекта повторит некий набор базовых действий (например, планирование, разработку, тестирование). Итерации затрагивают запросы и отчеты рабочих элементов, поскольку рабочие элементы группируются по итерациям.
  - **Areas** Собственно организация работ в команде проекта. Например, команда может организовать свою деятельность, отталкиваясь от продукта или отдельного компонента и создав область интерфейса, область приложения и область базы данных. Области также используются, чтобы сгруппировать рабочие элементы для запросов и отчетов.
- **XML-файл Windows SharePoint Services** Файл WssTasks.xml хранится в подпапке Windows SharePoint Services иерархии папок шаблона процесса. Здесь задаются три основные задачи: шаблон сайта, библиотеку документов, а также папки и файлы.
  - **Site Template** Шаблон сайта, на основе которого будет создан портал проекта. Шаблон также должен быть доступен на TFS SharePoint Portal. Шаблоны сайтов в шаблоны процессов не включаются.
  - **Document Libraries** После создания портала проекта вы можете также создать дополнительные библиотеки документов.
  - **Folders and Files** После создания портала проекта вы можете задать создание дополнительных папок, а также указать конкретные файлы для копирования, например, файлы шаблона.
- **XML-файл управления версиями** Файл VersionControl.xml хранится в подпапке Version Control иерархии папок шаблона процесса. В нем задаются начальные разрешения безопасности по управлению версиями, заметки для возврата после правки, а также требуется или нет единоличное редактирование.
  - **Check-in Notes** Здесь задается использование заметок для возврата после правки (check-in note). В этих заметках разработчик, возвращая код после правки, указывает, как внесенные в код изменения соотно-

сятся с общим процессом. Например, в заметке можно написать, что изменение было внесено как часть обзора безопасности, а также подробно описать его.

- **Exclusive Check-out** Здесь задается возможность одновременного использования одного и того же файла несколькими пользователями.
- **Permissions** Определяет, какие действия группам безопасности и отдельным пользователям разрешается выполнять в рамках управления версиями.

- **XML-файл отчетов** Файл ReportsTasks.xml хранится в подпапке Reports иерархии папок шаблона процесса. В нем определены начальные отчеты команды проекта.
  - **Reports Site** На сайт отчетов указывает ссылка Reports на домашней странице портала проекта.
  - **Folders** На сайте отчетов можно создавать папки. Созданная папка будет отображена на сайте проекта и в папке Reports обозревателя Team Explorer.
  - **Reports** Используется для добавления отчетов при помощи файлов .rdl.
- **XML-файл групп и разрешений** Файл GroupsandPermissions.xml хранится в подпапке Groups and Permissions иерархии папок шаблона процесса и используется для задания начальных групп безопасности в команде проекта.
  - **Groups** Здесь задается новая группа безопасности TFS.
  - **Permissions** Здесь задаются разрешения для каждой созданной группы.

Мастер New Team Project

Мастер **New Team Project** используется для создания новых командных проектов при помощи надстроек и XML-файлов определения процесса.

Способ настройки

Чтобы настроить шаблон процесса, выполните следующие действия:

1. Изучите шаблоны процесса TFS и выберите тот, который максимально отвечает потребностям вашей организации.
2. Загрузите выбранный шаблон.
3. Настройте различные компоненты шаблона.
4. Выгрузите настроенный шаблон обратно в TFS.
5. Убедитесь, что внесенные изменения соответствуют принятому процессу.

Эта общая последовательность используется в составе следующих подходов к настройке шаблонов процесса:

- **Ручная настройка XML-файлов** Ручная настройка чревата ошибками, но с другой стороны позволяет управлять тонкими деталями настройки. Подробнее — в статье «Customizing Process Templates» по адресу *http://msdn2.microsoft.com/en-us/library/ms243782(VS.80).aspx*.
- **Редактор Process Template Editor Tool из комплекта Power Tools** В последнюю версию комплекта усовершенствований, инструментов и утилит командной строки Visual Studio 2005 Team Foundation Server Power Tool входит программа с графическим интерфейсом для просмотра и редактирования шаблонов процесса. Подключившись к TFS, вы можете воспользоваться этим редактором для настройки определений типов рабочих элементов и глобальных списков активного проекта. Подробнее — в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server».

Основные настройки

Здесь описаны основные компоненты, которые, как правило, редактируются при настройке процесса:

- **Группы и разрешения** В стандартные шаблоны включен набор групп, которым уже назначены различные разрешения. Если группы по умолчанию и их разрешения не соответствуют требованиям вашего процесса, вы вольны их изменить или создать новые группы. Также можно добавить пользователя в группу, исключить пользователя из группы, назначить группе разрешение или отозвать его.
- **Заметки и политики возврата исходного кода после правки** В стандартные шаблоны включен набор заметок и политик возврата после правки. Если стандартные заметки не соответствуют требованиям вашего процесса, вы вправе добавить или удалить поля заметок, а также сделать некоторые из полей обязательными. Если вашим требованиям не соответствуют политики, создайте новые политики, а также отредактируйте или удалите имеющиеся.
- **Области и итерации** В стандартных шаблонах нет структуры классификации ни для областей, ни для итераций. Вы можете настроить их согласно конкретным требованиям процесса. Рекомендуется использовать в качестве основы для областей компоненты или возможности проекта. Итерация — это цикл повтора конкретного набора базовых действий (например, планирования, разработки или тестирования).
- **Портал команды** В стандартных шаблонах имеется портал команды по умолчанию, призванный стать средоточием коммуникаций между членами команды и другими сотрудниками организации. Вы можете изменить внешний облик портала, его действие и содержимое в согласии с требованиями процесса.

- **Руководство по процессу** В стандартные шаблоны включено соответствующее руководство, в котором объясняются роли, формы, отчеты и организация работы в команде проекта. Настраивая шаблон процесса согласно собственным требованиям, обязательно отредактируйте руководство, отразив изменения в различных компонентах.
- **Отчеты** В стандартных шаблонах имеется набор отчетов по умолчанию. Если они вам не подходят, создайте собственные отчеты на базе требований процесса.
- **Типы и запросы рабочих элементов** В стандартные шаблоны включены набор типов рабочих элементов, экземпляры рабочих элементов по умолчанию и запросы. Если они не соответствуют требованиям вашего процесса, внесите в типы рабочих элементов необходимые изменения, например:
  - Добавьте новые типы рабочих элементов.
  - Удалите существующие типы рабочих элементов.
  - Добавьте экземпляры по умолчанию для типов рабочих элементов.
  - Удалите экземпляры по умолчанию для типов рабочих элементов.
  - Создайте собственные открытые или закрытые запросы.

  Можно также вносить изменения в конкретный тип, например:
  - добавить поля;
  - переименовать поля;
  - задать допустимый диапазон значений поля;
  - изменить состояния и допустимые переходы состояний;
  - сделать поля обязательными или доступными только для чтения;
  - сделать одно поле зависимым от другого;
  - автоматически заполнять значения полей;
  - изменять способ представления информации на форме;
  - изменять соответствие между полем и столбцом Microsoft Office Project.

Как происходит настройка?

Процедура настройки шаблона разделяется на следующие этапы:

1. Пользователь запускает мастер **New Team Project**.
2. Мастер запрашивает:
   - имя проекта;
   - название шаблона, который нужно использовать при создании проекта.

   Отображаемые окна мастера зависят от используемых надстроек. Например, если в шаблон процесса не включена надстройка Windows SharePoint

Services, не будут отображаться окна для ввода информации о портале проекта.

3. Когда пользователь введет всю необходимую информацию и щелкнет **Finish**, мастер вызывает надстройки для выполнения действий по созданию проекта. Порядок вызова надстроек задается в XML-файлах определения процесса.
4. Мастер считывает инструкции в шаблоне процесса, а затем создает и настраивает конкретные элементы.

   Пользователю не нужно задавать никакие сведения о создаваемых типах рабочих элементов, поскольку все необходимые указания уже содержатся в шаблоне процесса.

Примечание Если в процессе создания проекта мастер столкнется с какими-либо осложнениями, вы увидите на экране сообщение с описанием проблемы и предложениями по ее устранению.

Резюме

Используйте шаблон процесса MSF Agile в проектах быстрой разработки ПО. Шаблон MSF CMMI предназначен для проектов создания ПО по методологии Capability Maturity Model Integration Института разработки ПО (Software Engineering Institute, SEI).

Основные компоненты архитектуры шаблона процесса — надстройки шаблона, XML-файлы определения процесса и мастер **New Team Project**.

Если стандартные шаблоны процесса не отвечают вашим потребностям, настройте их, вручную отредактировав XML-файлы определения процесса или воспользовавшись средством Process Editor Tool.

Чаще всего производится настройка групп и разрешений, заметок и политики возврата исходного кода после правки, областей и итераций, отчетов и определений типов рабочих элементов.

Дополнительные ресурсы

- Сведения о выборе шаблона процесса вы найдете в статье «Choosing a Process Template» по адресу *http://msdn2.microsoft.com/en-us/library/ms400752(vs.80).aspx*.
- Шаблон MSF CMMI доступен для загрузки по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=12A8D806-BB98-4EB4-BF6B-FB5B266171EB&displaylang=en*.
- Шаблон MSF Agile доступен для загрузки по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=EA75784E-3A3F-48FB-824E-828BF593C34D&displaylang=en*.

- Подробнее о настройке шаблона процесса читайте в статье «Process Template Customization Overview» по адресу *http://msdn2.microsoft.com/en-us/library/ms194945(VS.80).aspx*.
- Комплект Team Foundation Server Power Tools, включающий Process Template Editor, доступен для загрузки по адресу *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.
- Шаблоны процесса, предоставляемые партнерами Microsoft Partners, описаны по адресу *http://msdn2.microsoft.com/en-us/teamsystem/aa718801*.

ГЛАВА 14

Проекты MSF Agile

В этой главе

- Обстоятельства применения шаблона процесса Microsoft® Solution Framework (MSF) for Agile Software Development (MSF Agile).
- Использование шаблона MSF Agile.
- Настройка шаблона MSF Agile согласно конкретным потребностям команды.

Обзор

Процесс, определенный шаблоном MSF Agile, основан на идеях быстрой (Agile) разработки ПО, а также на принципах и методиках MSF. Стратегия быстрой разработки ПО, реализованная в шаблоне, основана на прохождении нескольких итераций и на сборке приложений на базе сценариев. В шаблон включены инструкции и средства автоматизации, необходимые для поддержки командной разработки, включая управление конфигурацией, управление проектом, отслеживание рабочих элементов и портал проекта.

В главе объясняется организация работы в типичном проекте MSF Agile, приведены примеры команд, использующих процесс MSF Agile, описаны параметры шаблона по умолчанию и варианты настройки шаблона.

Организация работы в проекте MSF Agile

В шаблоне MSF Agile определен набор задач, которые выполняются в ходе итераций цикла разработки ПО различными ролями, например: бизнес-аналитиками, архитекторами, менеджерами проекта, разработчиками и специалистами по тестированию. Основные действия, связанные с каждой задачей, описаны в табл. 14-1.

Табл. 14-1. Задачи и действия MSF Agile

Задача	Действия
Создание журнала проекта	Общее описание проекта
Создание архитектуры решения	Создание прототипа архитектуры Определение интерфейсов Создание архитектуры инфраструктуры
Создание сценариев использования	Разбиение на сценарии Внедрение сценариев в TFS Расстановка приоритетов сценариев
Планирование итераций	Распределение сценариев и требований по задачам разработки и тестирования Оценка задач разработки и тестирования Составление графика и назначение задач разработки и тестирования
Итерация разработки	Написание кода для задач разработки Создание или обновление модульного теста для задачи разработки Запуск модульного теста и анализ кода
Итерация тестирования	Создание проверочных тестов Запуск тестовых сценариев, исследовательское тестирование Оформление ошибок для выявленных проблем
Обзор результатов итерации	Анализ выполнения задач итерации Анализ выявленных ошибок Сравнение полученных метрик с пороговыми значениями

Параметры MSF Agile по умолчанию

Когда вы создаете новый командный проект на основе шаблона MSF Agile, в главном окне Microsoft Visual Studio® отображается страница с общим описанием процесса. Это ваше первое знакомство с процессом MSF Agile. Доступ к этой информации можно также получить с домашней страницы портала проекта.

Разумеется, одним описанием дело не ограничивается. Вам доступна настройка рабочих элементов (например, сценариев, требований к качеству обслуживания, задач, ошибок и рисков), отчетов по проекту, ролей (групп и

разрешений), а также портала проекта. Далее перечислены основные компоненты шаблона MSF Agile:

- рабочие элементы;
- группы и разрешения;
- контроль исходного кода;
- области и итерации;
- отчеты;
- портал.

Далее компоненты шаблона MSF Agile описаны более подробно.

Рабочие элементы

В шаблон процесса MSF Agile включены следующие типы рабочих элементов:

- **Bug** Реальная или потенциальная проблема в приложении.
- **Risk** Возможное событие или условие, способное отрицательно сказаться на проекте.
- **Scenario** Конкретная «траектория» взаимодействия пользователя с создаваемой вами системой.
- **Task** Конкретный фрагмент работы, выполняемый членом команды.
- **Quality of Service Requirement** Нефункциональное требование, относящееся, например, к безопасности, производительности или управляемости.

При создании нового проекта на базе шаблона MSF Agile в нем для экономии вашего времени создаются следующие задачи, выполнить которые нужно при инициализации проекта.

- **Set up: Set Permissions** Добавление членов команды в одну из четырех групп безопасности: Build Services, Project Administrators, Contributors и Readers.
- **Set up: Migration of Source Code** Перенос существующего исходного кода из Microsoft Visual SourceSafe® при переносе в Microsoft Visual Studio Team Foundation Server существующего проекта. Необходимо сначала завершить перенос исходного кода и лишь потом открывать членам команды доступ к проекту.
- **Set up: Migration of Work Items** Перенося в TFS существующий проект, вы можете также перенести рабочие элементы (например, ошибки и задачи) из Clearquest или при помощи файла с разделителями-запятыми. Необходимо сначала завершить перенос рабочих элементов и лишь потом открывать членам команды доступ к проекту.
- **Set up: Set Check-in Policies** Настройка бизнес-правил или политики, связанных с возвратом исходного кода после правки.

- **Set up: Configure Build** Создание исходного дерева источников и настройка сборки на регулярной основе (как правило, ежедневно).
- **Set up: Send Mail to Users for Installation and Getting Started** Отправка членам команды сообщений электронной почты с информацией о том, к какому TFS-серверу им подключаться и с каким проектом работать.
- **Create Vision Statement** Создание общего описания проекта с указанием конечной цели, одобренной всеми заинтересованными лицами.
- **Set up: Create Project Description on Team Project Portal** Изменение описания проекта по умолчанию в соответствии с решаемой задачей, например, включение в него описания целей и конечного результата проекта.
- **Create Personas** Создание персон, символизирующих целевых пользователей системы. Их удобно применять, например, продумывая дизайн приложения.
- **Define Iteration Length** Определение цикла итераций проекта. Зависит от объема и сложности проекта.
- **Create Test Approach Worksheet including Test Thresholds** Цель этой задачи — разобраться в стратегии тестирования с самого начала итераций проекта. Это поможет вам составить более эффективное расписание тестирования и изначально более четко направить усилия разработчиков.
- **Brainstorm and Prioritize Scenarios List** Выявление основных сценариев использования и оценка их приоритетности.
- **Brainstorm and Prioritize Quality of Service Requirements List** Выявление нефункциональных QoS-требований, связанных, например, с безопасностью, производительностью и управляемостью.
- **Set up: Create Project Structure** Создание структуры проекта с выделением основных областей, над которыми будет работать команда разработчиков.
- **Create Iteration Plan** Распределение задач разработки по итерациям.

Отчеты

По умолчанию в шаблоне MSF Agile доступны следующие отчеты:

- **Bugs by Priority** Правильные ли найдены ошибки? В этом отчете сравниваются темпы выявления ошибок с высоким и низким приоритетом.
- **Bug Rates** Насколько эффективно выявляются, исправляются и закрываются ошибки? На диаграмме иллюстрируются тенденции в появлении и исправлении старых новых ошибок, а также текущие ошибки.
- **Builds** Каково качество сборки? В отчете приводится список доступных сборок, включая их качество и другие подробные сведения о них.

- **Project Velocity** Насколько быстро команда завершает свою работу? Из этого отчета вы узнаете, насколько быстро команда справляется с плановыми заданиями, а также о том, как меняется темп работы.
- **Quality Indicators** Каково качество ПО? В одном отчете собраны результаты испытаний, ошибки, покрытие кода и сведения о его изменчивости.
- **Load Test Summary** Результаты практических испытаний приложения.
- **Regressions** Список тестов, которые раньше выполнялись, а теперь — нет.
- **Reactivations** Сколько рабочих элементов было повторно активировано? Из этого отчета вы узнаете, какие рабочие элементы были закрыты или помечены как разрешенные преждевременно.
- **Related Work Items** Как рабочие элементы зависят друг от друга? В этом отчете приводится список рабочих элементов, связанных с другими рабочими элементами.
- **Remaining Work** Сколько работы осталось сделать и когда она будет завершена? Из этого отчета вы узнаете объем оставшейся и выполненной работы. Проанализировав имеющиеся тенденции, вы предскажете примерный срок готовности кода.
- **Unplanned Work** Сколько выполняется внеплановых работ? В этом отчете показаны полный объем работ и оставшийся объем работ с разделением на плановые и внеплановые операции.
- **Triage** Какие рабочие элементы нуждаются в утверждении? В этом отчете показаны все рабочие элементы до сих пор имеющие статус предложения.
- **Work Items** Какие рабочие элементы активны? В отчете приводится список активных рабочих элементов.
- **Work Items by Owner** Сколько работы назначено каждому члену команды? В отчете показаны рабочие элементы для каждого члена команды.
- **Work Items by State** Сколько имеется активных, разрешенных и закрытых рабочих элементов? Ответ вы узнаете из этого отчета.

Группы и разрешения

По умолчанию в шаблоне MSF Agile доступны следующие группы:

- **Readers** Членам этой группы проект доступен только для чтения.
- **Contributors** Членам этой группу разрешается добавлять, изменять и удалять элементы проекта.
- **Build Services** Членам этой группы разрешается сборка проекта. Она предназначена только для учетных записей служб.

- **Project Administrators** Членам этой группы разрешается выполнять в проекте любые действия.

Управление исходным кодом

В MSF Agile по умолчанию используются следующие параметры управления исходным кодом:

- **Коллективное редактирование** По умолчанию в MSF Agile разрешается одновременное редактирование одного и того же файла несколькими членами команды. Все возникающие при этом конфликты разрешаются при возвращении файла после правки.
- **Разрешения** По умолчанию назначены следующие разрешения по контролю за исходным кодом:
  - **Project Administrators** Имеют все доступные права.
  - **Build Services** Имеют право читать, откладывать изменения, возвращать код после правки, помечать, начинать сборку, редактировать сборку.
  - **Contributors** Имеют право читать, откладывать изменения, брать код на редактирование, возвращать код после редактирования, помечать, начинать сборку.
  - **Readers** Имеют только право чтения исходного кода.

Области и итерации

В стандартных шаблонах нет структуры классификации ни для областей, ни для итераций. Вы можете настроить их согласно конкретным требованиям процесса. Рекомендуется использовать в качестве основы для областей компоненты или возможности проекта. Итерация — это цикл повтора конкретного набора базовых действий (например, планирования, разработки или тестирования).

Практические примеры использования MSF Agile

В этом разделе приводятся практические примеры использования MSF for Agile Software Development командой Майкрософт *patterns & practices* и сторонней командой разработчиков.

Пример 1: команда *patterns & practices*

В следующем примере показано, как процесс MSF Agile используется при выполнении типичного проекта команды *patterns & practices*.

Новый проект с нулевой итерации

- Менеджер продукта:
  1. Совместно с заказчиками и заинтересованными лицами формулирует требования к проекту. Они записываются в документ Microsoft Office Word с именем Project Back Log.
  2. Создает декларацию проекта при помощи Microsoft Office PowerPoint®.
  3. Совместно с заказчиками и заинтересованными лицами проводит совещание по выявлению основных сценариев использования продукта, выделяя ключевые требования к продукту и определяя его общий облик.
  4. Совместно с менеджером проекта и другими заинтересованными лицами определяет приоритеты сценариев.
- Менеджер проекта:
  1. Преобразует сценарии в рабочие элементы TFS.
  2. Принимает решение о продолжительности цикла итерации в зависимости от объема проекта и возможностей команды.

Планирование до начала итераций

- Опираясь на приоритеты, менеджер проекта принимает решение, над какими сценариями будет вестись работа в ходе итерации.
- Менеджер продукта и менеджер проекта формулируют требования качества обслуживания (Quality of Service, QoS) для сценария. Требования QoS привязываются к сценариям.

Планирование итераций

- Менеджер проекта:
  1. Совместно с разработчиками и другими членами команды разбивает сценарии на задачи для разработчиков.
  2. Переносит задачи разработки в TFS и связывает их со сценариями.
  3. Определяет критерии завершения для каждой из задач разработки.
  4. Разделяет требования QoS на задачи тестирования.
  5. Переносит задачи тестирования в TFS и связывает их с QoS-требованиями.
  6. Определяет критерии завершения для каждой из задач тестирования.
  7. Составляет расписание выполнения задач и назначает их членам команды.
- Разработчик оценивает все задачи разработки.

Важно! Если по предварительным расчетам на выполнение задачи разработки потребуется больше двух дней, ее следует разделить на меньшие задачи.

- Специалист по тестированию оценивает все задачи тестирования.

Во время итерации

- Менеджер проекта руководит итерацией.
- Разработчик пишет код для задачи и закрывает задачу, если выполнены критерии ее завершения.
- Тестировщик выполняет назначенные ему задачи тестирования и для каждой выявленной проблемы создает новый рабочий элемент-ошибку.

После итерации

- Менеджер проекта:
  1. Анализирует развитие проекта и заново расставляет приоритеты сценариев, которые не были завершены в ходе итерации.
  2. Предоставляет отчет о состоянии проекта заинтересованным лицам.
  3. На основании приоритетов решает, над какими сценариями будет вестись работа на следующей итерации.
- Менеджер продукта:
  1. Добавляет все вновь выявленные сценарии.
  2. Меняет приоритеты сценариев (при необходимости).
  3. Вместе с менеджером проекта формулирует требования QoS для сценария. Требования QoS привязываются к сценариям.

Пример 2: сторонние разработчики

В следующем примере показано, как процесс MSF Agile используется одним из сторонних разработчиков.

Новый проект с нулевой итерации

- Бизнес-аналитик:
  1. Пишет краткую (на одну страницу) декларацию проекта.
  2. Находит представителя заказчика, с которым можно будет обсудить входные данные и создать персоны.
  3. Совместно с заказчиком идентифицирует сценарии.
  4. Совместно с заказчиком определяет приоритеты сценариев.
  5. Пишет сценарии для предстоящей итерации.

- Менеджер проекта:
  1. Собирает совещание разработчиков, на котором они делятся своими оценками, пока приблизительными, по порядку величины.
  2. Проверяет, не изменились ли приоритеты после подсчета затрат.
  3. Составляет расписание сценариев для предстоящей итерации.
- Архитектор разделяет сценарии на задачи архитектуры.
- Разработчик:
  1. Разделяет сценарии на задачи разработки.
  2. Определяет соответствующую стратегию сборки (по возможности, непрерывную интеграцию).
- Специалист по тестированию разделяет сценарий на задачи тестирования.

Во время итерации

- Менеджер проекта:
  1. Руководит итерацией.
  2. Руководит проектом.
- Архитектор определяет архитектуру решения.
- Разработчик реализует задачу разработки.
- Тестировщик проводит тестирование сценария.

После нулевой итерации

На этом этапе задачи несколько меняются.

- Бизнес-аналитик:
  1. Обновляет персоны (при необходимости).
  2. Добавляет вновь обнаруженные сценарии.
  3. Изменяет приоритеты сценариев (при необходимости).
  4. Пишет сценарии для предстоящей итерации.
- Менеджер проекта:
  1. Оценивает новые сценарии.
  2. Составляет расписание сценариев для предстоящей итерации.
- Архитектор разделяет сценарии на задачи архитектуры.
- Разработчик:
  1. Разделяет сценарии на задачи разработки.
  2. Обновляет процесс сборки (по возможности, непрерывную интеграцию).
- Специалист по тестированию разделяет сценарий на задачи тестирования.

Настройка шаблона MSF Agile

Есть два способа отредактировать шаблон MSF Agile согласно потребностям вашей организации:

- **Ручная настройка XML-файлов** Ручная настройка чревата ошибками, но с другой стороны позволяет управлять тонкими деталями настройки. Подробнее — в статье «Customizing Process Templates» по адресу *http://msdn2.microsoft.com/en-us/library/ms243782(VS.80).aspx*.
- **Process Template Editor** В последнюю версию комплекта усовершенствований, инструментов и утилит командной строки Visual Studio 2005 Team Foundation Server Power Tool входит программа с графическим интерфейсом для просмотра и редактирования шаблонов процесса. Подключившись к TFS, вы можете воспользоваться этим редактором для настройки определений типов рабочих элементов и глобальных списков активного проекта. Подробнее — в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server».

Резюме

В шаблоне MSF Agile определен набор задач, которые должны выполняться различными ролями на протяжении цикла разработки ПО. Шаблон MSF Agile содержит определения рабочих элементов, групп и разрешений, управления исходным кодом, областей и итераций, отчетов и портала проекта.

Если стандартный шаблон процесса не отвечает потребностям вашей организации, настройте его, вручную отредактировав XML-файлы определения процесса или при помощи Process Editor Tool из комплекта TFS Power Tools.

Дополнительные ресурсы

- Шаблон процесса MSF Agile доступен для загрузки по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=EA75784E-3A3F-48FB-824E-828BF593C34D&displaylang=en*.
- Комплект Team Foundation Server Power Tools, включая Process Template Editor, доступен для загрузки по адресу *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.
- Подробнее о настройке шаблона процесса читайте в статье «Process Template Customization Overview» по адресу *http://msdn2.microsoft.com/en-us/library/ms194945(VS.80).aspx*.
- Подробнее о ручной настройке шаблона процесса читайте в статье «Customizing Process Templates» по адресу *http://msdn2.microsoft.com/en-us/library/ms243782(VS.80).aspx*.

ЧАСТЬ VII

Отчеты

В этой части

- Знакомство с отчетами.

ГЛАВА 15

Знакомство с отчетами

В этой главе

- Архитектура отчетов Microsoft® Visual Studio® Team Foundation Server.
- Компоненты системы подготовки отчетов TFS.
- Назначение имеющихся отчетов.
- Определение отчетов, содержащихся в каждом шаблоне процесса.
- Создание и настройка отчетов.

Обзор

В этой главе рассказывается об архитектуре системы подготовки отчетов TFS и об отчетах, чаще всего применяемых в командных проектах. Типовые сценарии сбора информации сопоставлены с отчетами, доступными в TFS. Описаны основные причины, по которым стоит изменять параметры существующих отчетов или создавать новые. Система отчетов TFS позволяет просматривать обобщенные данные по самым разным аспектам проекта. При помощи этой информации можно анализировать развитие проекта, его состояние, эффективность работы команд разработчиков и тестировщиков.

В системе отчетов TFS для создания, управления и запуска отчетов используются службы Microsoft SQL Server™ 2005 Reporting Services. В каждый шаблон процесса включен набор предопределенных отчетов, которые развертываются в папке отчетов проекта при его создании. С помощью Reporting Services вы можете также дополнять эти отчеты и создавать специфические отчеты для своего проекта. Затем вновь созданные отчеты можно добавить к шаблону процесса, чтобы они были доступны в других проектах.

Из этой главы вы узнаете, как работает система отчетов TFS и как с ее помощью оценить состояние проекта.

Сценарии и решения

Отчеты — основной инструмент получения информации о проекте для менеджеров проекта и руководителей групп. Когда вы создаете новый проект, одновременно с ним создается и набор отчетов, определяемый выбранным вами шаблоном. Эти отчеты доступны на сайте Microsoft Office SharePoint® портала проекта или в Visual Studio в узле отчетов обозревателя Team Explorer.

Далее перечислены основные вопросы, на которые должны отвечать отчеты TFS:

- Когда приложение будет готово к поставке?
- По плану ли идет работа?
- Соблюдается ли качество сборки?
- Насколько статус разработки соответствует определенным сценариям?
- Насколько быстро выполняется разработка?
- Все ли ошибки исправляются?
- Имеются ли случаи повторного возникновения ошибок?

Отчеты Team Foundation Server

Как шаблон MSF Agile, так и шаблон MSF CMMI содержат набор шаблонов по умолчанию.

Отчеты об ошибках

Отчеты об ошибках в шаблонах позволяют узнать, какие ошибки возникают, как они исправляются, на какие тенденции указывают. Доступны следующие отчеты об ошибках:

- **Bugs by Priority** Правильные ли выявляются ошибки? В этом отчете сравниваются темпы возникновения ошибок с большим и малым приоритетом. Отчет доступен в обоих стандартных шаблонах.
- **Bug Rates** Насколько эффективно выявляются, исправляются и закрываются ошибки? В этом отчете показаны тенденции возникновения новых ошибок, неразрешенные и разрешенные ошибки. Отчет доступен в обоих стандартных шаблонах.

Отчеты об управлении выпусками

Отчеты об управлении выпусками позволяют оценить, насколько создаваемое ПО близко к выпуску. Доступны следующие отчеты об управлении выпусками:

- **Actual Quality versus Planned Velocity** Сколько сценариев можно завершить, прежде чем качество станет неприемлемым? На каждой итера-

ции этот отчет представляет соотношение примерного объема проекта и общего качества. Отчет доступен в обоих стандартных шаблонах.

- **Builds** Каково качество сборки? В этом отчете содержится список имеющихся сборок, а также их качество и другая подробная информация. Отчет доступен в шаблоне MSF CMMI.
- **Quality Indicators** Каково качество ПО? В этом отчете собраны результаты тестов, ошибки, сведения о покрытии кода и его изменчивости. Отчет доступен в обоих стандартных шаблонах.
- **Velocity** Насколько быстро команда справляется с работой? Из этого отчета вы узнаете, насколько своевременно команда выполняет плановые задания и как темп ее работы меняется изо дня в день. Отчет доступен в обоих стандартных шаблонах.
- **Scenario Details** Для каких сценариев мы готовим приложение? В отчете содержатся сведения обо всех сценариях, включая информацию о завершенности, рисках и испытаниях. Отчет доступен в шаблоне MSF CMMI.

Отчеты о тестировании

Отчеты о тестировании позволяют следить за эффективностью испытаний. Доступны следующие отчеты о тестировании:

- **Regressions** Какие тесты ранее выполнялись, а теперь — нет? Их список содержится в этом отчете. Отчет доступен в шаблоне MSF CMMI.
- **Requirements Test History** Насколько хорошо протестированы сценарии и требования? В этом отчете показаны результаты испытаний определенных сценариев и требований. Отчет доступен в шаблоне MSF CMMI.
- **Test Failure Without Active Bug** Каждый ли из известных дефектов документирован как ошибка? В этом отчете показаны неудачные испытания, с которыми не связаны открытые ошибки. Отчет доступен в шаблоне MSF CMMI.
- **Test Passing With Open Bug** Своевременно ли обновляется список ошибок и согласуется ли он с качеством приложения? Отчет отображает список устаревших ошибок, тесты для которых теперь выполняются. Доступен в шаблоне MSF CMMI.
- **Load Test Summary** К каким выводам о производительности приложения привели испытания под нагрузкой? В отчете содержатся результаты испытания нагрузочного тестирования. Отчет доступен в шаблоне MSF Agile.

Отчеты о рабочих элементах

Отчеты о рабочих элементах позволяют оценивать текущее состояние проекта и его продвижение. Доступны следующие отчеты о рабочих элементах:

- **Open Issues and Blocked Work Items Trend** Сколько у вас осталось неразрешенных проблем? В отчете перечислены открытые проблемы и наметившиеся тенденции к их разрешению. Отчет доступен в шаблоне MSF CMMI.
- **Reactivations** Сколько рабочих элементов было повторно активировано? В отчете указаны рабочие элементы, которые были преждевременно закрыты или помечены как разрешенные. Отчет доступен в обоих стандартных шаблонах.
- **Related Work Items** Как одни рабочие элементы зависят от других рабочих элементов? В отчете отображается список рабочих элементов, которые связаны с другими рабочими элементами, что позволяет прослеживать зависимости между ними. Отчет доступен в шаблоне MSF CMMI.
- **Remaining Work** Сколько осталось выполнить работ и когда они будут завершены? В отчете отражена незавершенная работа, а также разрешенная и закрытая работа. Выявив тенденции, вы определите время, к которому код будет завершен. Отчет доступен в обоих стандартных шаблонах.
- **Triage** Какие рабочие элементы нуждаются в уточнении? В этом отчете показаны рабочие элементы, все еще имеющие статус предложения. Отчет доступен в шаблоне MSF CMMI.
- **Unplanned Work** Сколько выполняется внеплановых работ? В отчете полная работа сопоставляется с уже выполненной с разделением плановых и внеплановых задач. Отчет доступен в обоих стандартных шаблонах.
- **Work Items** Какие рабочие элементы активны? В отчете перечислены все активные рабочие элементы. Отчет доступен в шаблоне MSF CMMI.
- **Work Items by Owner** Сколько работы назначено каждому члену команды? В этом отчете рабочие элементы отсортированы по владельцам. Отчет доступен в шаблоне MSF CMMI.
- **Work Items by State** Сколько имеется активных, разрешенных и закрытых рабочих элементов? В этом отчете рабочие элементы отсортированы по состоянию. Отчет доступен в шаблоне MSF CMMI.

Настройка отчетов

Не исключено, что вам понадобится отчет, которого нет ни в одном шаблоне MSF. Есть три способа настройки отчета:

- **Фильтрование существующего отчета** Во многих отчетах имеются параметры, по которым содержание отчета можно отфильтровать. В частности, доступны фильтры по дате, области, итерации и приоритету, позволяющие просматривать подмножества данных отчета. Помните, что эти фильтры являются временными и прекращают действовать, когда вы выходите из просмотра отчета.

- **Настройка существующего отчета** Если нужный вам отчет похож на один из существующих отчетов, часто проще бывает сделать копию готового отчета и отредактировать ее. Например, у вас может возникнуть потребность нанести на график развитие рисков со временем, чтобы оценить, насколько хорошо команда с ними справляется.
- **Создание нового отчета** Конечно, можно создать новый отчет «с нуля».

Исправив существующий отчет или создав новый, опубликуйте его на сервере отчетов (Report Server), чтобы доступ к нему получили другие члены команды. Есть несколько способов редактирования существующего или создания нового отчета:

- Используйте Microsoft Office Excel® для создания сводной таблицы на основе данных из БД отчетов.
- Создайте в Visual Studio новый проект Report Server, а затем импортируйте в него существующий отчет или создайте новый.

Создание проекта Report Server в Visual Studio — наиболее мощный и гибкий метод работы с отчетами.

> **Примечание** С этой же целью можно использовать Report Builder, размещенный на сайте отчетов команды, однако этот инструмент не очень хорошо поддерживается в сценариях отчетов Visual Studio, поэтому его лучше не применять.

Физическая архитектура

Сервер Team Foundation Server построен на основе SQL Server 2005 и использует SQL Server Analysis Services для сбора данных и составления отчетов. Новые отчеты создаются при помощи Microsoft Excel или Visual Studio 2005 Report Designer, размещаются в SQL Server 2005 Reporting Services и доступны для просмотра на веб-сайте сервера отчетов, портале SharePoint проекта или в узле Reports обозревателя Team Explorer. Физическая архитектура системы подготовки отчетов показана на рис. 15-1.

Для каждого компонента TFS (рабочих элементов, управления исходным кодом, тестирования, ошибок и Team Build) ведется собственный набор баз данных транзакций. Эти данные собраны в реляционную базу данных. Затем эти данные помещаются в куб OLAP (Online Analytical Processing) и используются для составления отчетов с учетом тенденций и более детального анализа данных.

Реляционная БД TfsWarehouse представляет собой хранилище, предназначенное скорее для обслуживания запросов, чем для транзакций. Данные переносятся в это хранилище из различных БД TFS, оптимизированных для обработки транзакций. Хранилище не является основным источником данных для отчетов, но вполне может применяться для их составления. На эту

реляционную БД указывает источник данных TfsReportDS. OLAP-куб Team System Data Warehouse представляет собой базу данных OLAP, доступ к которой осуществляется при помощи SQL Server Analysis Services. Куб полезен при составлении отчетов, предполагающих анализ тенденций, например, «Насколько изменилось по сравнению с предыдущим месяцем количество закрытых ошибок?». На OLAP-куб Team System Data Warehouse в базе данных служб анализа указывает источник данных TfsOlapReportDS.

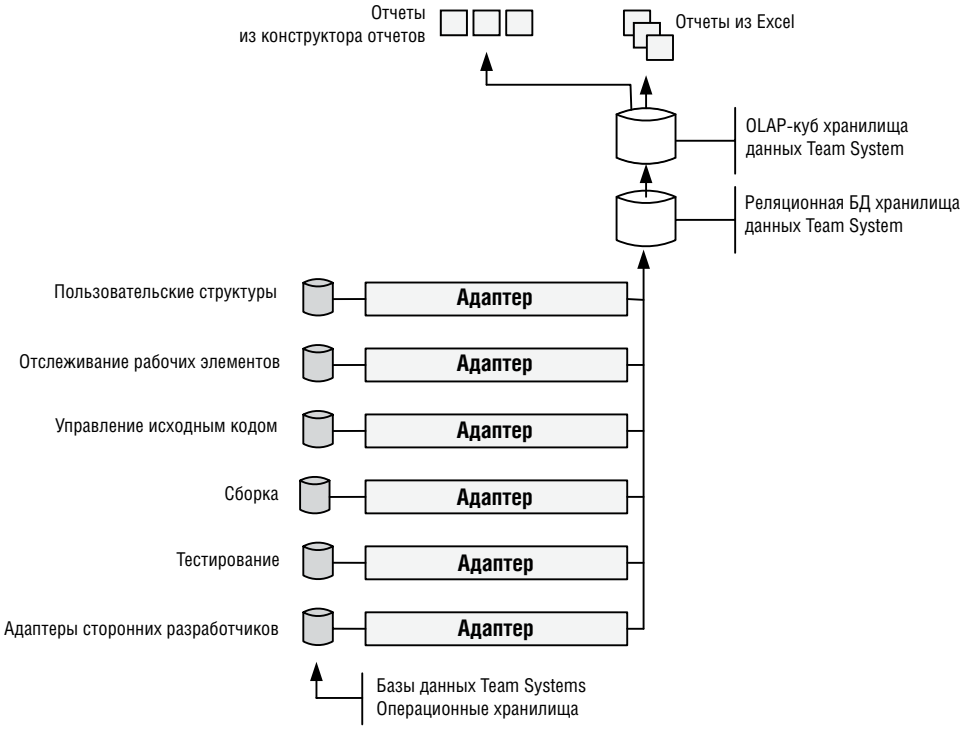


Рис. 15-1. *Физическая архитектура системы подготовки отчетов*

Компоненты системы подготовки отчетов

Система подготовки отчетов включает в себя компоненты как на стороне сервера, так и на стороне клиента.

Серверные компоненты

К числу серверных компонентов относятся:

- **Базы данных сервера отчетов** В этих БД содержатся определения отчетов, старые отчеты и данные о конфигурации.
- **Веб-служба сервера отчетов** Обеспечивает программируемый доступ к Report Server.

- **Веб-сайт диспетчера отчетов** Обеспечивает доступ пользователей к Report Server из веб-обозревателя.
- **Служба Windows** Обеспечивает составление расписания и доставку снимков отчетов.

Клиентские компоненты

К числу клиентских компонентов относятся:

- **Веб-обозреватель** Обеспечивает доступ к веб-сайту диспетчера отчетов.
- **Team Explorer** Обеспечивает доступ к отчетам из Visual Studio.

Средства разработки отчетов

К средствам разработки отчетов относятся:

- **Business Intelligence Designer Studio (BIDS)** Позволяет разработчикам конструировать и развертывать отчеты из Visual Studio 2005.
- **Excel** Применяется для построения сводных таблиц на основе сведений из хранилища отчетов.
- **Report Builder** Позволяет конечным пользователям конструировать собственные отчеты. В сценариях подготовки отчетов для Team Foundation поддерживается не очень хорошо и потому не рекомендован к использованию.

Резюме

В шаблонах MSF Agile и MSF CMMI содержится набор стандартных отчетов об ошибках, управлении выпуском, тестировании и отслеживании рабочих элементов:

- Отчеты об ошибках позволяют следить за обнаруживаемыми ошибками и своевременно выделять тенденции.
- По отчетам об управлении выпуском можно судить, готово ли приложение к выпуску.
- Отчеты о тестировании позволяют следить за тем, насколько эффективно проводятся испытания.
- Отчеты о рабочих элементах служат для оценки развития и текущего состояния проекта.

Чтобы изменить существующий отчет или создать новый, вы можете воспользоваться построителем отчетов Report Builder на сайте отчетов команды, создать при помощи Excel сводную таблицу на основе данных из отчетных БД или создать в Visual Studio новый проект Report Server.

Дополнительные ресурсы

- Подробную информацию о настройке существующего отчета вы найдете в разделе «Как настроить отчет в Visual Studio 2005 Team Foundation Server».
- Подробную информацию о создании отчетов вы найдете в разделе «Как создать собственный отчет в Visual Studio 2005 Team Foundation Server».
- Инструкцию по созданию отчета о рисках вы найдете в разделе «Как создать отчет о развитии рисков в Visual Studio 2005 Team Foundation Server».
- Подробнее об отчетах Team Foundation Server читайте по адресу *http://msdn2.microsoft.com/en-us/library/ms194922(VS.80).aspx*.

ЧАСТЬ VIII

Настройка и обслуживание командной среды

В этой части

- Развертывание Team Foundation Server.
- Интернет-доступ к Team Foundation Server.

ГЛАВА 16

Развертывание Team Foundation Server

В этой главе

- Преимущества и недостатки односерверного и раздельного развертывания.
- Выбор топологии развертывания, отвечающей требованиям организации.

Обзор

В этой главе описан общий подход к развертыванию Microsoft® Visual Studio® 2005 Team Foundation Server (TFS), а также основные решения, которые необходимо принимать при развертывании TFS в организации. В главе рассказывается о двух вариантах развертывания и объясняется, как выбрать между двумя этими вариантами.

Два этих варианта — односерверная и раздельная установка. В первом случае уровень данных и уровень приложений размещаются на одном сервере. Во втором случае эти уровни размещаются на разных серверах. Кроме того, вы вольны установить на отдельных компьютерах сервер сборки и прокси управления исходным кодом. Доступ к этим серверам требуется каждому клиенту, и потому на клиентской стороне необходимо установить соответствующие инструменты.

Архитектура TFS

Архитектура TFS показана на рис. 16-1.

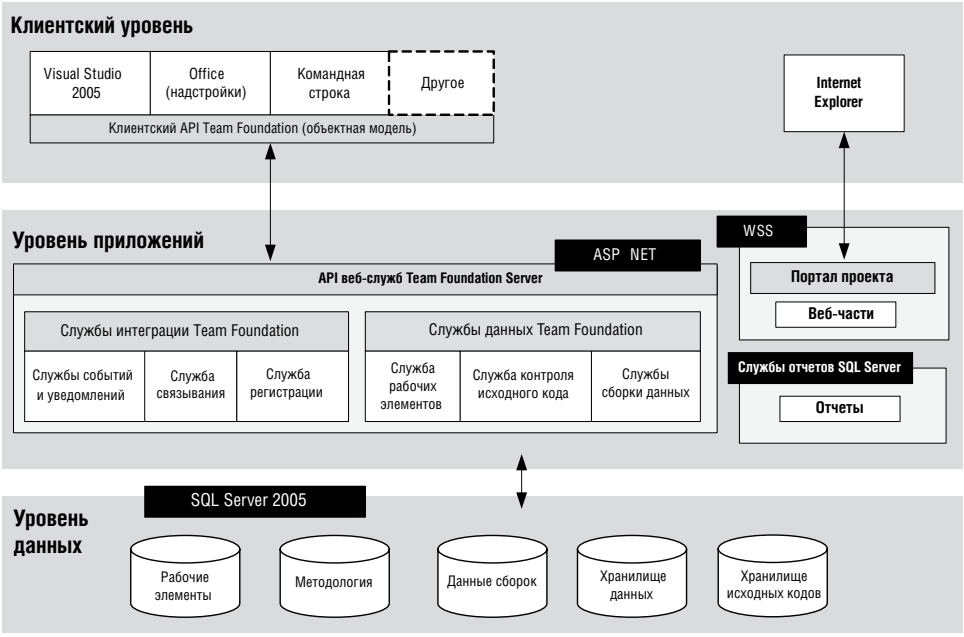


Рис. 16-1. *Архитектура TFS*

С точки зрения архитектуры TFS разделен на три уровня — уровень данных (data tier), уровень приложений (application tier) и клиентский уровень (client tier). Разделение это логическое, и все три уровня вполне можно установить на одном и том же компьютере.

На уровне данных Team Foundation находится Microsoft SQL Server™ 2005. С ним устанавливается ряд баз данных для хранения рабочих элементов, версий, результатов испытаний и любых отчетов.

На уровне приложений содержатся веб-интерфейс, встроенный в Internet Information Services (IIS), веб-службы Team Foundation и службы Microsoft Office SharePoint®. Также на уровне приложений находятся серверы сборки и прокси управления исходным кодом.

На клиентском уровне находятся приложения, осуществляющие доступ к TFS. Разработчики используют для подключения к Team Server обозреватель Team Explorer, установленный как самостоятельное приложение или как часть Visual Studio 2005. Менеджеры проекта пользуются Microsoft Office Excel® или Microsoft Office Project. Для подключения к серверу можно применять и инструменты сторонних разработчиков.

Подробнее — в главе 2.

Сценарии развертывания

Существуют следующие способы развертывания TFS:

- Развертывание с одним сервером;
  - в рабочей группе;
  - при помощи Microsoft Active Directory®.
- Развертывание с несколькими серверами.

Односерверное развертывание в рабочей группе

В этом варианте создается рабочая группа без использования контроллера домена Active Directory. Такой подход удобен в небольших коллективах. Для подключения к серверу каждому пользователю необходима локальная учетная запись на нем. При использовании рабочих групп раздельное развертывание не поддерживается.

Односерверное развертывание в Active Directory

Если вы используете Active Directory, вам доступны оба варианта развертывания. Вы вольны установить уровни данных и приложений как на одном, так и на нескольких серверах.

Какой тип развертывания выбрать?

Чтобы выбрать вариант развертывания, максимально отвечающий потребностям вашей организации, ответьте на следующие вопросы:

- **Сколько пользователей мне предстоит поддерживать?** Если в ваших планах значится более 400 пользователей, вам, вероятно, стоит подумать о раздельном развертывании.
- **Сколько проектов я буду поддерживать при помощи TFS?** Если проектов много, вам, скорее всего, лучше развернуть TFS на нескольких серверах. Каждый экземпляр TFS способен поддерживать до 5000 проектов. Если у вас более 5000 проектов, одним экземпляром Team Foundation Server вам не обойтись.
- **Могу ли я выделить для TFS специальный сервер?** При односерверном развертывании TFS компьютер не должен выполнять никакие иные функции, то есть, не должен быть почтовым сервером, файловым сервером или сервером баз данных для других приложений.

Преимущества односерверного развертывания

У односерверного развертывания есть следующие преимущества:

- **Простота**
  - Все аспекты TFS управляются на одном сервере.
  - На одном сервере настраиваются права и разрешения пользователей и групп.

- Только для одного сервера нужно настраивать обслуживание и резервное копирование.

- **Доступность** Поскольку как уровень приложений, так и уровень данных размещены на одном сервере, при планировании развертывания вам не нужно учитывать сетевые ограничения и задержки.

Преимущества раздельного развертывания

У раздельного развертывания есть следующие преимущества:

- **Масштабируемость** В варианте развертывания с одним сервером можно обслуживать не более 400 пользователей, тогда как разделение функций серверов позволяет увеличить количество пользователей до 2000.
- **Отказоустойчивость** На время ремонта или планового обслуживания можно перенаправить сервер уровня приложений на другой сервер уровня данных. А вот резервный сервер уровня приложений настроить нельзя.

Односерверное развертывание

Типичное развертывание с одним сервером показано на рис. 16-2. На сервере установлены уровни данных и приложений TFS, а также SharePoint Services и SQL Server 2005.

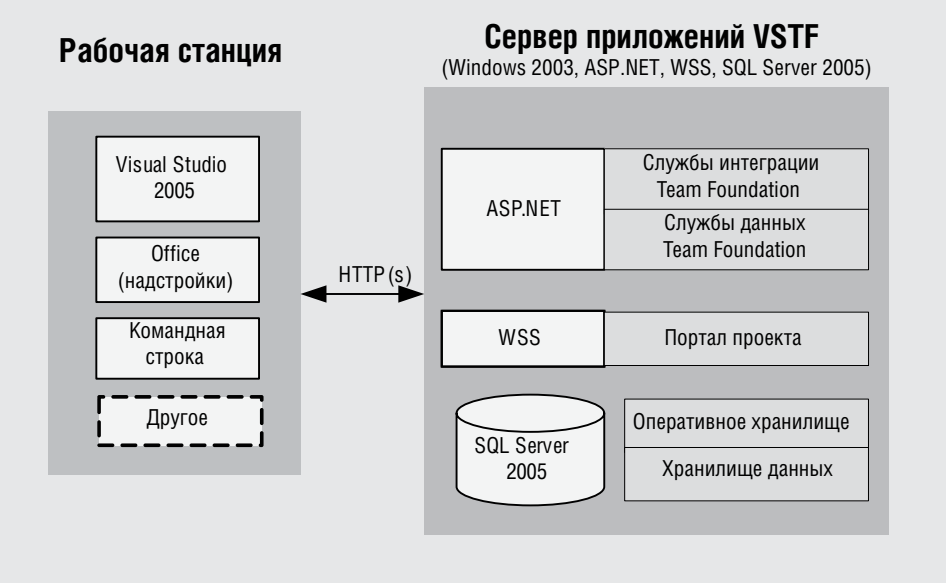


Рис. 16-2. *Типичное развертывание с одним сервером*

Развертывание на раздельных серверах

Типичный вариант развертывания с несколькими серверами показан на рис. 16-3. Уровень приложений TFS установлен совместно с SharePoint Services. На другом компьютере размещены уровень данных TFS и SQL Server 2005.

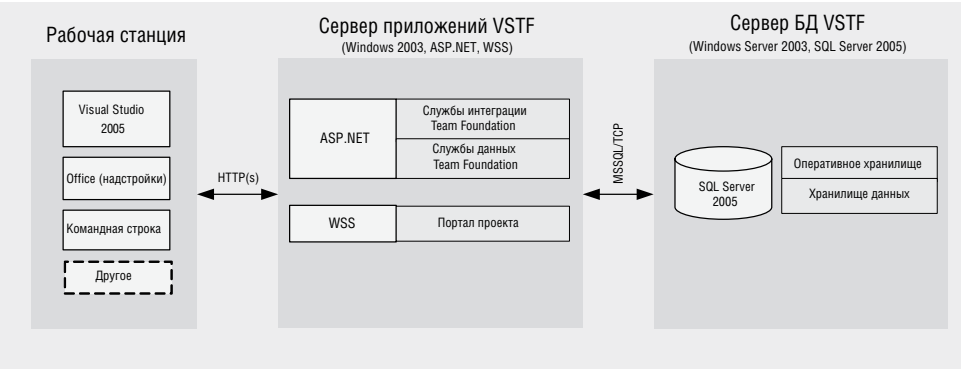


Рис. 16-3. *Типичное раздельное развертывание*

Другие серверы

Как в односерверном, так и в раздельном варианте вы вольны установить также сервер сборки и прокси-сервер. Их можно установить как на том же сервере, что и уровень приложений, так и на других серверах.

Установка сервера сборки

Чтобы повысить производительность сборки и снизить нагрузку на уровень приложений, разместите службы сборки на отдельном сервере. Например, это нужно сделать, если сборки планируется проводить достаточно часто.

Прокси-сервер Team Foundation

Прокси-сервер Team Foundation кеширует копии файлов, включенных в систему управления исходным кодом. Используйте прокси-сервер, если вы обращаетесь к серверу управления исходным кодом по сети и испытываете проблемы с ее быстродействием.

Топологии TFS

Выбрав вариант установки, вы должны затем выбрать одну из нескольких топологий. К вашим услугам как простые, так и сложные топологии — для команд самых различных размеров.

Простая топология

На рис. 16-4 показана самая простая топология TFS — уровни приложений и данных развернуты на одном и том же сервере. Прокси-сервер TFS развер-

нут на отдельном сервере. Доступ к серверу имеется с клиентских рабочих станций в том же домене.

Эта конфигурация подходит для команд разработчиков и для пилотных проектов с числом пользователей не более 400.

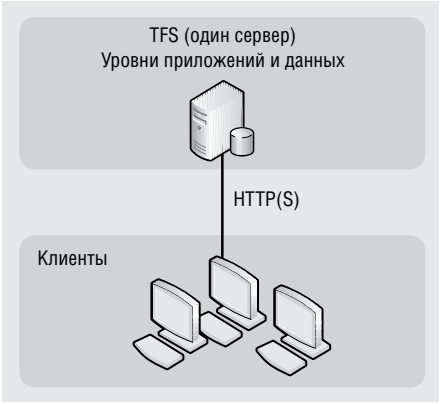


Рис. 16-4. *Простая топология TFS*

Топология умеренной сложности

На рис. 16-5 показан вариант с топологии с разделением уровней. Службы приложений развернуты на одном сервере, базы данных — на другом.

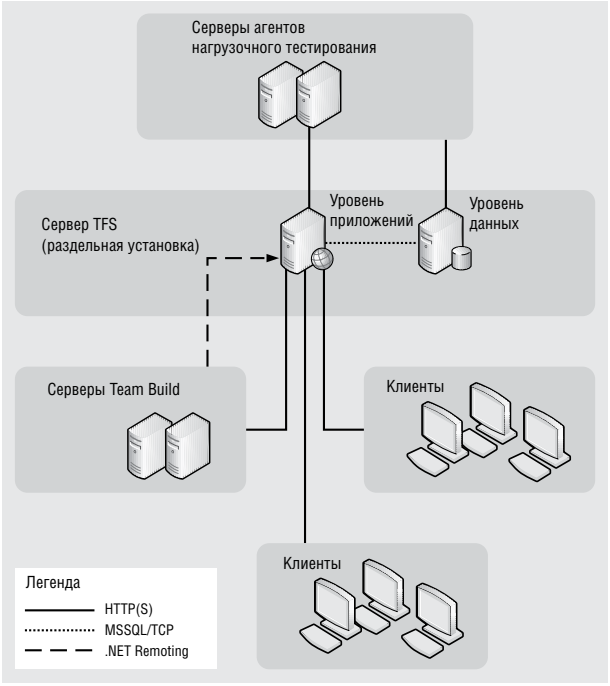


Рис. 16-5. *Топология TFS умеренной сложности*

На рис. 16-5 показано также испытательное оборудование и серверы сборки, развернутые на отдельных узлах. Клиентские узлы находятся либо в том же домене, что и серверы, либо в доменах, которые связаны с серверами отношениями доверия. Топологии этого уровня сложности уместны в больших командах разработки с количеством пользователей от 400 до 2000.

Сложная топология

Сложная топология, показанная на рис. 16-6, близка к предыдущему случаю. Однако теперь в нее добавлены компоненты отказоустойчивости — резервный сервер уровня приложений и уровень данных с использованием технологий кластеризации SQL.

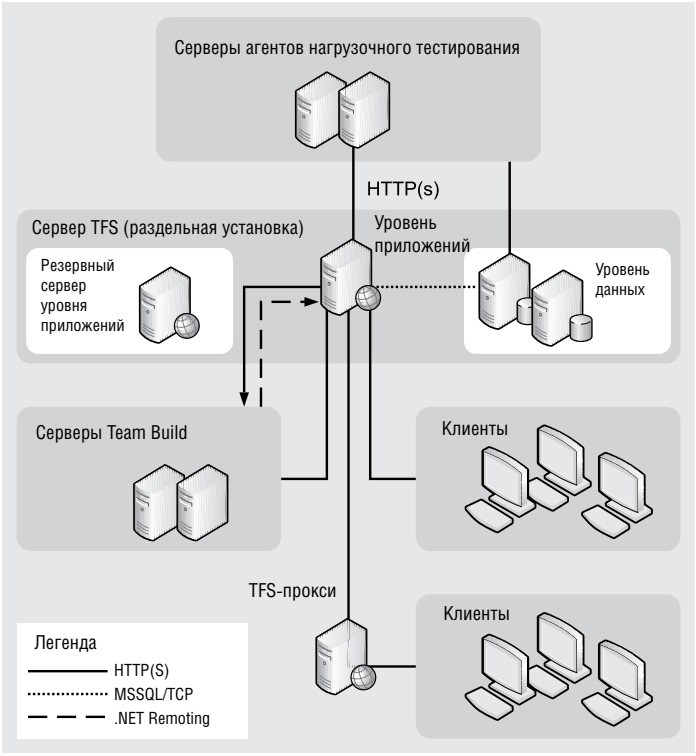


Рис. 16-6. *Сложная топология TFS*

Кроме того, на рис. 16-6 показан географически удаленный дочерний домен, связанный с основным доменом при помощи низкоскоростного соединения. Клиенты в этом домене для более эффективного доступа к системе управления исходным кодом используют прокси-сервер TFS.

Дополнительные соображения

При развертывании TFS учитывайте следующее:

- Если у вас уже есть настроенный сервер SharePoint, который вы хотите использовать для размещения портала Team Foundation Server, у вас имеется возможность переместить портал на этот сервер. Подробнее — по адресу *http://blogs.msdn.com/bharry/archive/2006/10/30/moving-your-tfs-sharepoint-site.aspx*.
- В крупных проектах выигрыша позволяет добиться перемещение механизма и куба на отдельный компьютер. Вы можете задать кластеризацию SQL на уровне данных и настроить конфигурацию «активный-активный» с SQL на одном узле и OLAP на другом, так что бы каждый из них был отказоустойчивым резервом для другого. Подробнее — по адресам *http://msdn2.microsoft.com/en-us/library/aa721760(vs.80).aspx* и *http://msdn2.microsoft.com/en-us/library/ms252505(VS.80).aspx*.

Стратегия масштабирования и архивации TFS

Планируя установку и развертывание Team Foundation Server, вы должны среди прочего решить, как будете управлять архивацией серверов и их восстановлением после сбоев. Выбор соответствующих стратегий определяется размером системы и доступными ресурсами. Поскольку уровень данных опирается на SQL Server 2005, подбор стратегии зависит от того, какой подход к архивации SQL Server вы в данный момент используете.

Если вы используете зеркалирование или кластеризацию SQL Server 2005, тот же самый подход можно применить и к уровню данных TFS. Вам также предстоит решить, как поступать в случае сбоя на сервере уровня приложений. Чтобы добиться отказоустойчивости, подготовьте резервный сервер уровня приложений и обеспечьте возможность быстрого переключения на него.

Выбор стратегии для компании

Выбирая стратегию установки архивации и восстановления TFS, учитывайте следующие соображения:

- размер команд;
- количество проектов;
- размер проектов;
- расположение команд;
- потребность в отказоустойчивости;
- потребность в архивации.

Рекомендуемое оборудование Team Foundation Server

Как правило, небольшие команды с незначительным числом проектов могут работать в односерверной среде, тогда как большим командам необходима раздельная установка и более быстрое оборудование. Выбор между односерверной и раздельной установкой влияет также на механизмы архивации и восстановления после сбоев.

Табл. 16-1 поможет вам решить, какой вариант установки стоит выбрать и какое оборудование использовать.

Табл. 16-1. Подбор оборудования для TFS

Конфигурация	Уровень	ЦП	Жесткий диск	Память
Один сервер, менее 20 пользователей	Сервер уровня приложений и данных	Одиночный процессор, 2,2 ГГц	8 Гб	1 Гб
Один сервер; от 20 до 100 пользователей	Сервер уровня приложений и данных	Двойной процессор, 2,2 ГГц	30 Гб	2 Гб
Несколько серверов; от 100 до 250 пользователей	Сервер уровня приложений	Одиночный процессор, 2,2 ГГц	20 Гб	1 Гб
	Сервер уровня данных	Двойной процессор, 2,2 ГГц	80 Гб	2 Гб
Несколько серверов; от 250 до 2000 пользователей	Сервер уровня приложений	Двойной процессор, 2,8 ГГц	40 Гб	4 Гб
	Сервер уровня данных	Счетверенный процессор, 2,7 ГГц	Накопитель прямого подключения, 14 000–15 000 RPM RAID 0	16 Гб

Стратегия архивации и восстановления

Выбирая стратегию архивации и восстановления, необходимо, конечно, учитывать, какой ущерб вашей команде будет нанесен в результате выхода из строя конкретного сервера.

Архивация

Планирование стратегии архивации является частью плана развертывания TFS. Учитывайте следующие соображения:

- частоту архивации;
- частоту полной и добавочной архивации;
- требования к хранению архивов, например, внутри организации или вне ее.

Вы вольны использовать ту же методику архивации, что используете для любой БД SQL Server 2005. Восстановление TFS из архивов проводится по одному из трех сценариев:

- восстановление только данных;
- полное восстановление односерверной установки;
- полное восстановление раздельной установки.

Восстановление только данных применяется в случае сбоя на уровне данных. С помощью архивных данных и журналов вы полностью восстановите БД. Восстановление серверов требуется при их выходе строя. В этом случае вы можете восстановить полную БД на втором компьютере.

Резервный сервер уровня приложения

На серверах уровня приложений нет никаких данных, которые нуждались бы в архивации, но это не делает их неуязвимыми. Чтобы минимизировать последствия сбоя, подумайте о подготовке «теплого» резервного сервера, на который можно было бы оперативно переключиться.

Восстановление после сбоя

Оценивая необходимость отказоустойчивого решения для TFS, принимайте во внимание стоимость оборудования для резервных серверов и сопоставляйте ее с возможными потерями от временной недоступности TFS. Обеспечение отказоустойчивости повышает сложность установки и накладные расходы на ее обслуживание. Не забудьте включить обслуживание в свои расчеты.

Особенно велики затраты на создание и обслуживание кластеров, поэтому кластеризацию рекомендуется использовать лишь в тех случаях, когда ресурсы для создания кластера в вашей организации уже имеются.

С зеркалированием также связаны расходы, но они несравнимы с расходами на кластеризацию. Причем в этом случае у вас появляется возможность временно отключить основной сервер, скажем, для мелких ремонтных работ. Подумайте о зеркалировании, если ваша организация может позволить себе установку и обслуживание второго сервера уровня данных.

Уровень данных

Кластеризация серверов уровня данных

Если у вашей организации есть необходимые ресурсы, создайте кластер с выделенными серверами. Кластер обеспечит непрерывный доступ к уровню данных. Правда, за это придется заплатить высокую цену с точки зрения как создания, так и обслуживания кластера.

Работая в кластере, TFS поддерживает конфигурацию с пассивным узлом, активным узлом и одним сервером кворума. Когда уровень данных после сбоя передается на пассивный узел, этот узел принимает на себя владение кворумом и уровнем данных.

Устанавливать TFS в кластере можно лишь после предварительной подготовки. Подробную информацию о кластеризации SQL Server 2005 вы найдете в статье «SQL Server 2005 Failover Clustering White Paper» по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=818234dc-a17b-4f09-b282-c6830fead499&displaylang=en*.

Зеркалирование серверов уровня данных

Зеркалирование сервера состоит в синхронизации данных на этом сервере с их копией на другом сервере. Сервер уровня данных является главным, а сервер с зеркалированными данными — резервным или зеркальным сервером. Если сервер уровня данных выходит из строя, переключиться на зеркальный сервер можно вручную.

Наличие зеркального сервера позволяет отключать главный сервер для обслуживания или ремонта, а также обеспечивает возможность быстрого восстановления в случае выхода из строя главного сервера уровня данных.

Зеркалирование может быть синхронным или асинхронным. Можно также менять главный и зеркальный серверы ролями. Когда производится *смена ролей* (role switching), зеркальный сервер становится главным, а бывший главный — зеркальным. В принципе, смену ролей можно производить неоднократно.

Автоматическое переключение TFS с главного на зеркальный сервер не поддерживается, это нужно делать вручную.

Чтобы настроить зеркалирование SQL для уровня данных, выполните следующие действия:

1. Выполните полную архивацию всех БД и журнала транзакций.
2. Создайте резервную копию ключа шифрования Reporting Services.
3. Установите на зеркальном сервере SQL Server 2005.
4. Восстановите данные с уровня данных на зеркальном сервере.
5. Для каждой БД на главном сервере уровня данных запустите мастер настройки зеркального сервера **Configure Database Mirroring Security Wizard**.
6. Начните зеркалирование.

Чтобы вручную переключиться на зеркальный сервер, выполните следующие действия:

1. На уровне приложений TFS:
   a. Перенастройте Report Service на использование нового сервера.
   б. Остановите веб-узел по умолчанию.

в. Остановите веб-узел SharePoint Web.
г. Остановите службу SharePoint Timer.
д. Остановите службу TfsServerScheduler.
е. Основите пул приложений ReportServer.
ж. Остановите пул приложений TFS App Pool.

2. Убедитесь, что на зеркальном сервере уровня данных созданы все необходимые учетные записи служб.
3. Переключите все БД главного сервера на зеркальный сервер.
4. Создайте на новом сервере хранилище данных.
5. Настройте сервер уровня приложений на использование зеркального сервера уровня данных:
   а. Введите в командной строке **TFSAdminUtil RenameDT MirrorDataTierServer**.
   б. Перезапустите IIS.
   в. Измените строку подключения Reporting Services, включив в нее ссылку на зеркальный сервер уровня данных.
   г. Задайте на сервере SharePoint использование зеркального сервера уровня данных.
   д. Запустите службу SharePoint Timer.
   е. Запустите службу TfsServerScheduler.
   ж. Запустите пул приложений ReportServer.
   з. Запустите пул приложений TFS App Pool.
   и. Запустите Reporting Services.
   к. Вызовите веб-службу StampWorkItemCache.

Уровень приложений

Отказоустойчивость уровня приложений

Настроив главный сервер уровня приложений, подготовьте также резервный компьютер для «теплой замены» на случай выхода главного сервера из строя.

Резервное оборудование и ПО

Резервный сервер необязательно должен быть идентичен главному серверу, но он должен удовлетворять аппаратным требованиям к серверу уровня приложений. На него необходимо установить ПО уровня приложений TFS.

Убедитесь, что у обоих серверов одинаковая конфигурация, включая одинаковые учетные записи пользователей, одинаковые разрешения и обновления ПО. Применяя на главном компьютере очередное обновление, не забудьте применить его и на резервном сервере.

Чтобы свести проблемы с обеспечением отказоустойчивости к минимуму, настройте сетевые адаптеры главного и резервного компьютеров на использование одного и того же имени хоста. Сделать это можно многими способами.

Переключение сервера уровня приложений

Переключение на резервный сервер уровня приложений производится вручную. Если главный сервер вышел из строя, выполните следующие действия:

1. Отключите главный сервер.
2. На резервном сервере:
   а. Зарегистрируйтесь как администратор.
   б. Запустите **TFSAdminUtil** с параметром **ActivateAT.**
   в. Запустите на резервном сервере веб-службы.

   Эта команда выполняет следующие действия:

- Регистрирует имя резервного сервера в интеграционной БД TFS.
- Подключает резервный сервер уровня приложений к активному серверу уровня данных.
- Проверяет, что в результате подключения корректный сервер уровня приложений соединен с корректным сервером уровня данных.

Подробнее об активации резервного сервера уровня приложений читайте в статье «How to: Activate a Fail-Over Application-Tier Server» по адресу *http://msdn2.microsoft.com/en-us/library/ms252501(VS.80).aspx*.

Резюме

Архитектура TFS состоит из трех уровней: уровня приложений, уровня данных и клиентского уровня. Устанавливая сервер, вы выбираете между установкой уровней данных и приложений на одном и том же или на разных серверах. Выбор конкретного варианта развертывания TFS определяется, главным образом, количеством пользователей, которых вам предстоит поддерживать. Выбрав топологию, отвечающую потребностям команды, решите также, какой способ архивации и восстановления вам подходит.

На уровне данных удобно использовать тот же механизм архивации, который принят в вашей организации для других архивов SQL Server 2005. Отказоустойчивость можно обеспечить при помощи зеркалирования или кластеризации.

Автоматическое восстановление уровня данных после сбоя не производится. Если вам нужно быстро вернуть систему в рабочее состояние, подготовьте резервный сервер для быстрого «теплого» переключения.

Дополнительные ресурсы

- Если вы не совсем разобрались в архитектуре TFS, перечитайте главу 2.
- Подробнее об установке TFS читайте в справочнике «Visual Studio 2005 Team Foundation Installation Guide» по адресу *http://go.microsoft.com/fwlink/?linkid=40042*.
- Дополнительную информацию о пределах масштабирования TFS вы найдете в статье «Team Foundation Server Capacity Planning» по адресу *http://blogs.msdn.com/bharry/archive/2006/01/04/509314.aspx*.
- Дополнительную информацию о том, как переместить куб OLAP и механизм анализа на отдельный сервер вы найдете в статье «How To: Move the Data Warehouse SQL Server Analysis Services Database to a Separate Server» по адресу *http://msdn2.microsoft.com/en-us/library/aa721760(vs.80).aspx*.
- Дополнительную информацию о кластеризации SQL Server 2005 вы найдете в документе «SQL Server 2005 Failover Clustering White Paper» по адресу *http://www.microsoft.com/downloads/details.aspx?familyid= 818234dc-a17b-4f09-b282-c6830fead499&displaylang=en*.
- Дополнительную информацию о создании отказоустойчивого кластера SQL Server вы найдете в статье «How To: Create a New SQL Server 2005 Failover Cluster (Setup)» по адресу *http://uat.technet.microsoft.com/en-us/library/ms179530(SQL.90).aspx*.
- Дополнительную информацию о том, как настроить кластер SQL Server для уровня данных вы найдете в статье «Clustering the Data-Tier Server» по адресу *http://msdn2.microsoft.com/en-us/library/ms252505(VS.80).aspx*.
- Дополнительную информацию о том, как переместить сайт TFS SharePoint на другой сервер вы найдете в статье «Moving your TFS SharePoint site» по адресу *http://blogs.msdn.com/bharry/archive/2006/10/ 30/moving-your-tfs-sharepoint-site.aspx*.
- Подробную информацию о масштабируемости Team Foundation Server вы найдете в блоге Брайана Харри (Brian Harry) по адресу *http://blogs.msdn.com/bharry/archive/2005/12/09/502190.aspx*.
- Дополнительную информацию о восстановлении после сбоев вы найдете в статье «Visual Studio Team System User Education» по адресу *http://www.microsoft.com/technet/itshowcase/content/vs05teamsystemnote.mspx*.
- Дополнительную информацию о архивации Team Foundation Server и его восстановлении после сбоев вы найдете в статье «Ensuring Team Foundation Server Availability» по адресу *http://msdn2.microsoft.com/en-gb/library/ms253159(VS.80).aspx*.
- Дополнительную информацию о кластеризации серверов уровня данных вы найдете в статье «Clustering the Data-Tier Server» по адресу *http://msdn2.microsoft.com/en-gb/library/ms252505(VS.80).aspx*.

- Дополнительную информацию о зеркалировании уровня данных Team Foundation Server вы найдете в статье «Mirroring the Team Foundation Data-Tier Server» по адресу *http://msdn2.microsoft.com/en-gb/library/aa980644(VS.80).aspx*.
- Дополнительную информацию о настройке зеркалирования SQL Server на уровне данных вы найдете в статье «How to: Configure SQL Server Mirroring for the Team Foundation Data-Tier Server» по адресу *http://msdn2.microsoft.com/en-us/library/aa980629(VS.80).aspx*.
- Дополнительную информацию о восстановлении уровня данных вы найдете в статье «How To: Fail Over to a Mirrored Data-Tier Server» по адресу *http://msdn2.microsoft.com/en-gb/library/aa980627(VS.80).aspx*.
- Дополнительную информацию о восстановлении уровня данных при недоступности главного сервера вы найдете в статье «How To: Fail Over to a Mirrored Data-Tier Server if the Principal Data-Tier Server is Unavailable» по адресу *http://msdn2.microsoft.com/en-gb/library/aa980528(VS.80).aspx*.
- Дополнительную информацию о том, как активировать резервный сервер уровня приложений, вы найдете в статье «How To: Activate a Fail-Over Application-Tier Server» по адресу *http://msdn2.microsoft.com/en-us/library/ms252501(VS.80).aspx*.
- Дополнительную информацию о том, как активировать резервный сервер уровня приложений, вы также найдете в статье «Activating a Fail-Over Application-Tier Server» по адресу *http://msdn2.microsoft.com/en-us/library/ms252486(VS.80).aspx*.

ГЛАВА 17

Интернет-доступ к Team Foundation Server

В этой главе

- Основные сценарии удаленного доступа и обстоятельства их применения.
- Удаленный доступ к Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) через Интернет.
- Повышение эффективности удаленного доступа при помощи Team Foundation Server Proxy.

Обзор

Из этой главы вы узнаете, как предоставить удаленный доступ к TFS через Интернет. Доступны три варианта удаленного доступа:

- посредством виртуальной частной сети (VPN);
- посредством обратного прокси, например, Microsoft Internet Security and Acceleration (ISA) Server;
- разместив сервер TFS в экстрасети.

В версиях TFS до Service Pack 1 (SP1) поддерживается только VPN-доступ. В версию TFS SP1 добавлена поддержка обычной (Basic) проверки подлинности, благодаря чему помимо VPN стали возможными решения с экстрасетью и обратным прокси.

В главе также подробно описаны процедуры установки и настройки сертификатов и SSL для использования с обычной или краткой (Digest) проверкой подлинности. Из раздела «Повышение эффективности удаленного доступа» вы узнаете, как сократить объем передаваемой по Интернету информации.

Основные стратегии

Ниже перечислены основные стратегические варианты предоставления удаленного доступа к TFS-серверу:

- **VPN-подключение** TFS находится в интрасети. Внешние пользователи подключаются к нему посредством VPN, внутренние — непосредственно.
- **Публикация TFS посредством обратного прокси** TFS находится в интрасети. Клиентские запросы из Интернета направляются к нему одним или несколькими обратными прокси, например, ISA Server.
- **Размещение TFS в экстрасети («сценарий с размещением»)** Если доступ к TFS нужен только внешним клиентам, его можно разместить в экстрасети, вне пределов действия брандмауэра.

Типичные сценарии использования удаленного доступа к TFS таковы:

- **Удаленный офис** В этом случае вы наверняка обеспечили удаленным сотрудникам вход в сеть посредством VPN, и потому вам удобно использовать вариант с VPN. Он прост в реализации, обеспечивает уверенную безопасность, открывает удаленный доступ ко всем компонентам TFS и позволяет использовать TFS-прокси для повышения производительности.
- **Разбросанная команда** Если вы не предоставили удаленным пользователям доступ VPN или доступ к домену, используйте вариант с обратным прокси. Это решение сложнее в настройке, но оно позволяет удаленным пользователям обращаться к TFS в интрасети без использования VPN.
- **Работа в сообществе** Если вы поддерживаете работу группы удаленных пользователей, специально для которых установили TFS, используйте вариант с экстрасетью. Он позволит максимально отделить удаленных пользователей от ресурсов внутренней сети.

VPN-подключение

Архитектура подключения к TFS посредством VPN показана на рис. 17-1.

Этот подход позволяет удаленным командам разработчиков использовать прямое VPN-подключение к TFS во внутренней сети. Если у вас установлен TFS без SP1 или если вам требуется встроенная проверка подлинности Windows, помимо VPN никаких других вариантов у вас и нет. Впрочем, TFS предназначен для работы в низкоскоростных сетях, подобных VPN, и потому в этом варианте обеспечит вполне приемлемую производительность.

Преимущества

- Работают все компоненты TFS, включая TFS Proxy.
- Поддерживается использование встроенной проверки подлинности Windows; возможно использование существующей инфраструктуры предприятия.
- Велика вероятность того, что VPN-доступ у вас уже и так настроен.

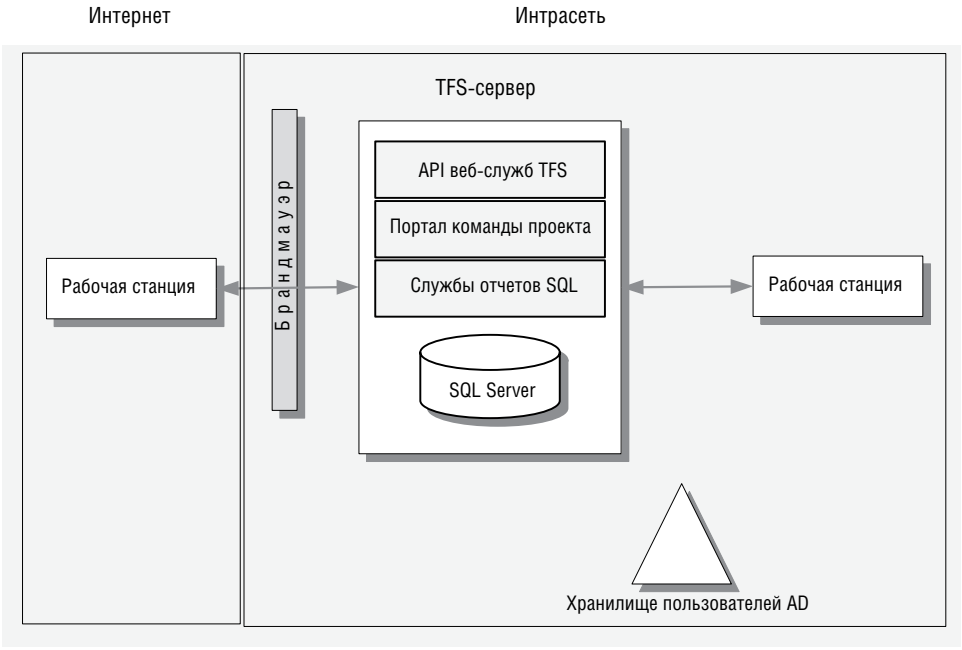


Рис. 17-1. *Архитектура доступа к TFS посредством VPN*

Недостатки

В вашей инфраструктуре решение VPN может отсутствовать или, по крайней мере, быть недоступным для удаленных пользователей. Более подробную информацию о настройке VPN вы найдете по адресу *http://support.microsoft.com/kb/324747*.

Публикация TFS посредством обратного прокси

В этом случае вы устанавливаете TFS-сервер во внутренней сети и используете возможность веб-публикации в ISA Server для предоставления доступа к нему из внешней сети. Удаленные пользователи обращаются к TFS посредством SSL и используют обычную проверку подлинности. Чтобы воспользоваться этим вариантом, вы должны установить TFS SP1.

Сети без контроллера домена в периметре

На рис. 17-2 показана архитектура предоставления доступа к TFS посредством ISA для случая, когда контроллер домена находится во внутренней сети.

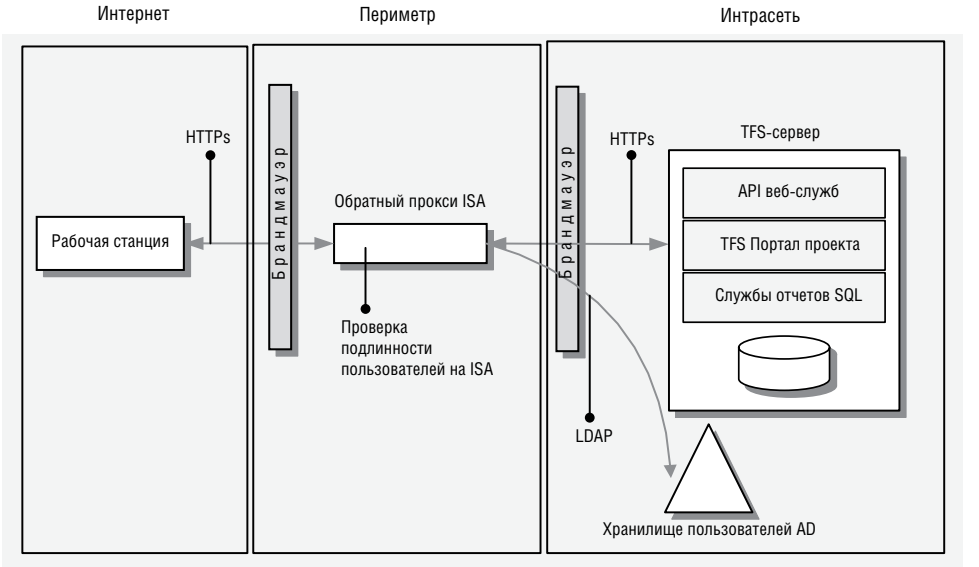


Рис. 17-2. *Доступ к TFS посредством ISA, контроллер домена расположен во внутренней сети*

Если в сети периметра нет контроллера домена, откройте на брандмауэре порт, через который внешние пользователи TFS могли посредством протокола LDAP (Lightweight Directory Access Protocol) подключаться к внутреннему контроллеру домена через ISA Server для проверки подлинности.

Сеть с контроллером домена в периметре

На рис. 17-3 показана архитектура предоставления доступа к TFS посредством ISA для случая, когда контроллер домена находится в сети периметра.

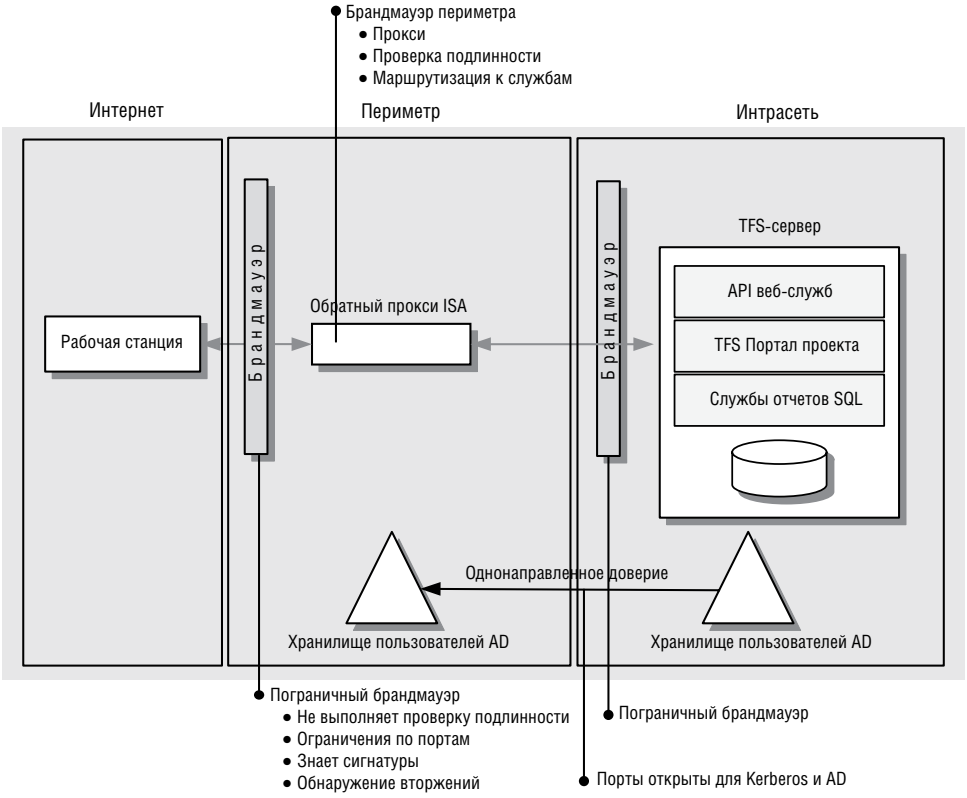


Рис. 17-3. *Доступ к TFS посредством ISA, контроллер домена расположен в сети периметра*

Если в сети периметра имеется контроллер домена, удаленные пользователи могут проходить проверку подлинности непосредственно на нем. Однонаправленное доверие между внутренним контроллером домена и контроллером домена в периметре позволяет внешним пользователям получать доступ к TFS-серверу. Внутренние пользователи обращаются к нему непосредственно, используя проверку подлинности Windows.

Преимущества

- Обратные прокси, наподобие ISA Server, проверяют не только подлинность пользователей, но и трафик.
- Удаленным пользователям не нужен доступ к домену.
- Удаленным пользователям не нужен доступ к VPN.

Недостатки

- В удаленных филиалах нельзя использовать TFS Proxy.

- Удаленным пользователям не разрешено добавлять пользователей в группы TFS.
- Удаленным пользователям не разрешено добавлять группы Microsoft Active Directory® в папки системы управления исходным кодом.
- У удаленных пользователей не будет возможности запускать сборку или управлять ею.
- Удаленным пользователям нельзя будет создавать новые командные проекты.
- Удаленным пользователям нельзя будет публиковать в TFS результаты тестирования.

> **Примечание** Всегда используйте обычную проверку подлинности в сочетании с SSL. Иначе учетные данные будут передаваться открытым текстом.

Размещение TFS в экстрасети

На рис. 17-4 показана архитектура размещения TFS в экстрасети.

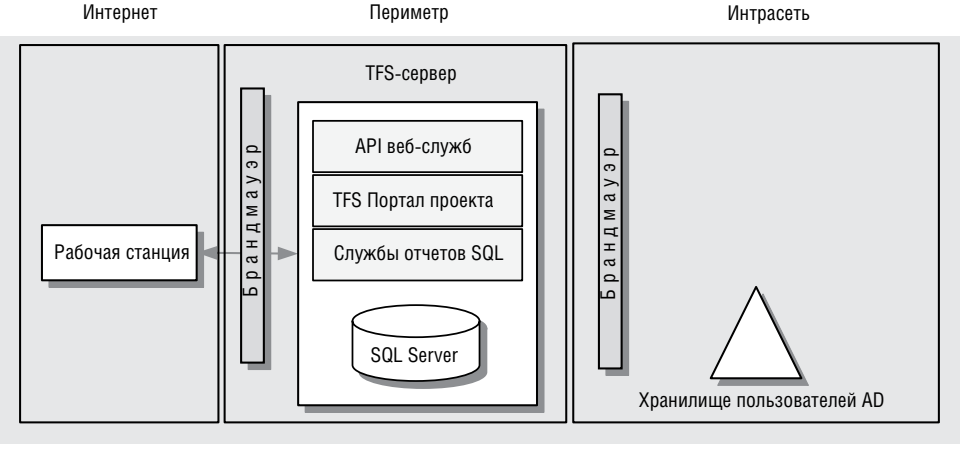


Рис. 17-4. *TFS размещен в экстрасети*

В этом варианте вся инфраструктура TFS, включая и уровень приложений, и уровень данных, располагается в сети периметра, за пределами внутренней сети. Все подключения к TFS — и от внешних, и от внутренних пользователей — производятся из Интернета. TFS способен работать как с контроллером домена (domain controller, DC), так и без него. Если из сети периметра доступа к DC нет, функции TFS, связанные с Active Directory, работать не будут. Например, в такой ситуации нельзя будет добавлять пользователей в группы TFS или добавлять группы Active Directory в папки системы управления исходным кодом.

Преимущества

- Пользователи TFS четко отделены от внутренней сети.
- Удаленным пользователям не нужен доступ к домену.

Недостатки

- В удаленных филиалах нельзя использовать TFS Proxy.
- У удаленных пользователей не будет возможности запускать сборку или управлять ею.
- Удаленным пользователям нельзя будет создавать новые командные проекты.
- Удаленным пользователям нельзя будет публиковать в TFS результаты тестирования.
- Внутренние пользователи вынуждены обращаться к TFS посредством SSL, подобно внешним пользователям.

> **Примечание** Всегда используйте обычную проверку подлинности в сочетании с SSL. Иначе учетные данные будут передаваться открытым текстом.

Обычная проверка подлинности и SSL

Если вы установили TFS SP1 и хотите воспользоваться сценарием развертывания TFS в экстрасети или с использованием обратного прокси, вам придется разрешить обычную проверку подлинности при помощи SSL, соответствующим образом настроив IIS на уровне приложений TFS. При использовании обычной проверки подлинности учетные данные передают по Интернету в незащищенной кодировке Base64. Чтобы защитить учетные данные, применяйте обычную проверку подлинности только по подключениям HTTPS, в которых используется SSL.

Настройте фильтр Internet Server API (ISAPI) так, чтобы удаленные клиенты подключались при помощи обычной проверки подлинности с SSL, а локальным клиентам разрешалась бы встроенная проверка подлинности Windows. Фильтр ISAPI выделяет клиентов, которых вы настроили как «external/Internet», и удаляет проверку подлинности NTLM по отклику 401, чтобы заставить эти клиенты использовать другие методы проверки подлинности, например, обычную.

Дополнительная информация

Подробнее о настройке TFS-сервера на использование обычной проверки подлинности и HTTPS при удаленных подключениях читайте в статье «Walkthrough: Setting up Team Foundation Server to Require HTTPS and Secure Sockets Layer (SSL)» по адресу *http://msdn2.microsoft.com/en-us/library/aa833873(VS.80).aspx*.

Подробнее о настройке ISAPI-фильтра читайте в статье «Walkthrough: Setting up Team Foundation Server with Secure Sockets Layer (SSL) and an ISAPI Filter» по адресу *http://msdn2.microsoft.com/en-us/library/aa833872(VS.80).aspx*.

Более подробную информацию об ISAPI-фильтрах для TFS вы найдете в публикации «The TFS extranet ISAPI filter mechanics» из блога Джеймса Мэннингса (James Manning) по адресу *http://blogs.msdn.com/jmanning/archive/2006/10/27/the-tfs-quot-extranet-quot-isapi-filter-mechanics.aspx*.

Team Foundation Server Proxy

На рис. 17-5 показана архитектура использования Team Foundation Server Proxy.

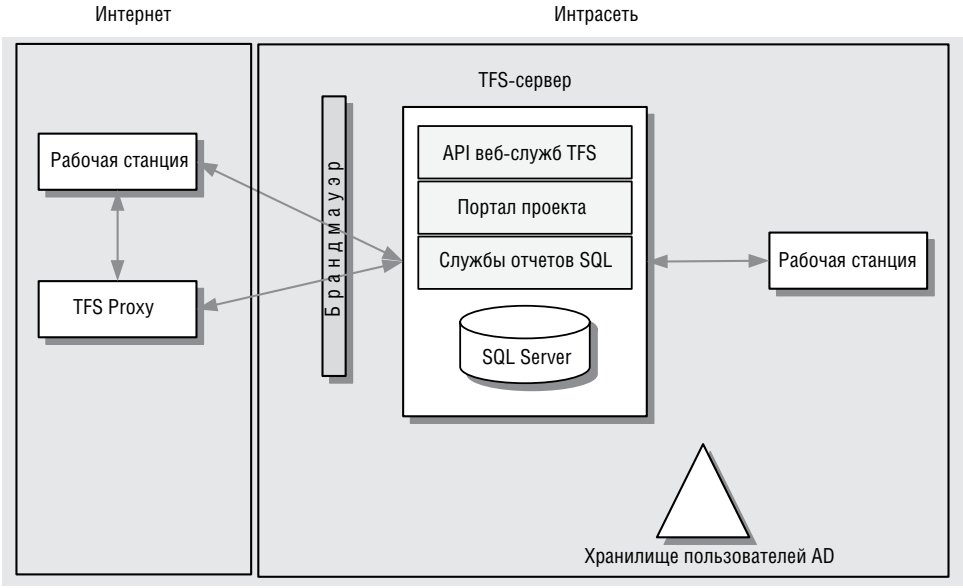


Рис. 17-5. *Архитектура TFS Proxy*

Чтобы повысить производительность удаленного филиала, подключающегося к TFS через VPN, установите там TFS Proxy. Вообще, для обеспечения удаленного доступа TFS Proxy не требуется, но удобен в качестве кеша для файлов системы управления исходным кодом. Когда удаленному клиенту понадобится файл исходного кода из хранилища, он сначала запросит его в TFS Proxy. Если в прокси имеется локальная версия, возвращена клиенту будет она. Если локальной версии в кеше нет, прокси запросит исходный код у удаленного TFS-сервера. Это позволяет сократить сетевой трафик и ускорить работу с исходным кодом в удаленном филиале.

Советы по повышению производительности прокси

Настраивая прокси, учитывайте следующие соображения:

- Убедитесь, что кеширование включено, и следите за производительностью кеша. Вообще, регулярно просматривайте счетчики производительности (устанавливаемые по умолчанию) и журналы событий (на предмет наличия ошибок и предупреждений) прокси-сервера. TFS Proxy сохраняет информацию о производительности кеша в XML-файле ProxyStatistics.xml, который находится в папке App_Data установочной папки прокси. Интервал сохранения информации можно изменить.
- Запускайте по расписанию задачу переноса на прокси-сервер последних версий файлов. Это гарантирует, что в кеше прокси всегда хранятся наиболее актуальные версии и что последующие запросы клиентов можно будет удовлетворить из кеша.
- Если вы заранее знаете о готовящейся передаче больших файлов по медленной сети (< 3 Мбит/с), присвойте соответствующее значение параметру **executionTimeout** в файле Web.config. Значение по умолчанию равно одному часу — <httpRuntime executionTimeout="3600"/>.
- Если вы собираете отключить прокси на длительный срок, отключите его использование на клиентах, чтобы избежать безрезультатных соединений. По умолчанию попытки подключения осуществляются каждые 5 минут.
- Чтобы предотвратить ненужную передачу файлов и запретить просмотр определенных рабочих областей, сделайте их скрытыми (cloaked). Это сокращает сетевой трафик и экономит локальное дисковое пространство, поскольку предотвращает копирование в локальную рабочую область файлов и папок, которые вам в данный момент не нужны. Скрыть можно и существующее сопоставление папок в рабочей области, однако лучше создать для сокрытия специальное сопоставление.

Более подробную информацию вы найдете в разделе «Распределенная и удаленная разработка» части «Руководство по управлению исходным кодом» этой книги.

Зеркальные учетные записи

Работа TFS Proxy в удаленных офисах поддерживается только при использовании подключения VPN. Однако в небольших удаленных группах это возможно и при развертывании TFS в экстрасети или при помощи обратного прокси. Для этого вам понадобятся зеркальные учетные записи (mirrored accounts).

Чтобы задействовать TFS Proxy, используйте на TFS, TFS Proxy и каждом из удаленных клиентских компьютеров учетные записи рабочей группы с точно совпадающими именами и паролями. Необходимость поддержки идентичных учетных записей в трех различных местах существенно увеличивает затраты времени на администрирование. Именно поэтому такое решение рекомендуется только для небольших удаленных групп.

Дополнительная информация

- Подробную информацию о TFS Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.
- Подробную информацию об управлении TFS Proxy вы найдете в статье «Managing Remote Connections to TFS Proxy» по адресу *http://msdn2.microsoft.com/en-us/library/ms253156(VS.80).aspx*.
- Подробную информацию о диагностике TFS Proxy вы найдете в статье «Troubleshooting TFS Server Proxy» по адресу *http://msdn2.microsoft.com/en-us/library/ms400681(VS.80).aspx*.

Удаленный сервер сборки

На рис. 17-6 показана архитектура использования удаленного сервера сборки.

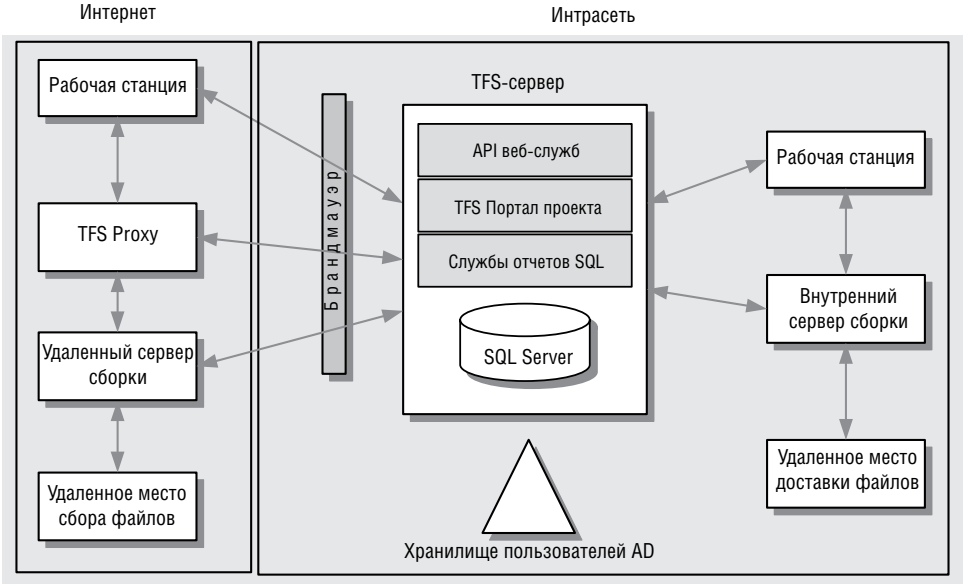


Рис. 17-6. *Архитектура удаленного сервера сборки*

Чтобы повысить эффективность работы удаленной команды, настройте в удаленном офисе собственный сервер сборки. Если там установлен TFS Proxy, сервер будет действовать подобно любому другому клиенту системы управления исходным кодом, перед каждой сборкой запрашивая исходный код у прокси. У удаленного сервера сборки есть следующие преимущества:

- Благодаря тому что удаленные сборки осуществляются «на месте», снижается нагрузка на внутренний сервер сборки.
- Благодаря удаленной сборке сокращается потребность в передаче по сети двоичных файлов.

Не используйте удаленный сервер сборки в качестве полной замены. Даже если удаленная сборка генерируется из того же набора исходных файлов, что и внутренняя, приложение может вести себя по-разному из-за разницы конфигураций серверов в удаленном и главном офисах. Как правило, все важные испытания должны проводиться на внутреннем сервере сборки, особенно, когда близок выпуск очередной версии продукта.

> **Примечание** Уровень приложений взаимодействует с сервером сборки по порту 9191. Если вы используете удаленный сервер сборки, настройте брандмауэр так, чтобы приложения могли подключаться через этот порт.

Резюме

Если вы используете TFS без SP1, для удаленного доступа вам доступно только VPN-подключение. При установленном TFS SP1 вы можете развернуть TFS в экстрасети или при помощи обратного прокси, используя обычную проверку подлинности с SSL.

Чтобы повысить эффективность работы удаленных групп, особенно, при использовании VPN-подключения, установите и настройте TFS Proxy для кеширования файлов системы управления исходным кодом.

Дополнительные ресурсы

- Подробную информацию о настройке удаленного доступа к TFS вы найдете в статье «Walkthrough: Setting up TFS for a Remote Development Location» по адресу *http://msdn2.microsoft.com/en-us/library/ms242919(VS.80).aspx*.
- Подробную информацию об использовании TFS с SSL вы найдете в статье «Walkthrough: Setting up Team Foundation Server with Secure Sockets Layer (SSL)» по адресу *http://msdn2.microsoft.com/en-us/library/ms242875(VS.80).aspx*.
- Подробную информацию об использовании обычной и краткой проверки подлинности в TFS вы найдете в статье «Team Foundation Server, Basic Authentication, and Digest Authentication» по адресу *http://msdn2.microsoft.com/en-us/library/aa833874(VS.80).aspx*.
- Подробную информацию о TFS Proxy вы найдете по адресу «Team Foundation Server Proxy and Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.
- Подробную информацию об управлении TFS Proxy вы найдете в статье «Managing Remote Connections to TFS Proxy» по адресу *http://msdn2.microsoft.com/en-us/library/ms253156(VS.80).aspx*.

- Подробную информацию о диагностике TFS Proxy вы найдете в статье «Troubleshooting TFS Server Proxy» по адресу *http://msdn2.microsoft.com/en-us/library/ms400681(VS.80).aspx*.
- Подробную информацию о проверке производительности TFS Proxy вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms252455(VS.80).aspx*.

ЧАСТЬ IX

Система Visual Studio 2008 Team Foundation Server

В этой части

- Что нового в Visual Studio 2008 Team Foundation Server.

ГЛАВА 18

Что нового в Visual Studio 2008 Team Foundation Server

В этой главе

- Новые возможности и функции Microsoft® Visual Studio® 2008 Team Foundation Server.

Обзор

В продукт Microsoft® Visual Studio® 2008 Team Foundation Server внесен целый ряд новых возможностей и функций. Области основных изменений таковы:

- **Администрирование, работа и установка** Система стала проще, что позволило сократить время установки. Кроме того, в ней поддерживается больше сценариев развертывания.
- **Сборка** В комплект изначально включены непрерывная сборка, сборка по расписанию и очереди сборки. Упростилось управление сборкой и ее расширяемость. Больше функций стало доступно из пользовательского интерфейса.
- **Управление версиями** Лучше поддерживается автономный режим работы, повышена производительность.
- **Отслеживание рабочих элементов** В новую версию включен усовершенствованный построитель запросов, улучшена поддержка вложений рабочих элементов.

Далее приводятся более подробные описания этих изменений. Их связь с инструкциями, приведенными в этой книге, обобщена в специальной табли-

це. Эта глава поможет вам при планировании обновления Microsoft Visual Studio Team Foundation Server.

Администрирование, работа и установка

- **Упрощенная установка** По сравнению с Visual Studio 2005 TFS установка существенно упростилась и выполняется быстрее. Это обусловлено отказом от отдельной установки уровня данных, а также от требования учетной записи домена. По возможности Team Foundation Server 2008 использует встроенные системные учетные записи (например, Network Service).
- **Поддержка SharePoint 2007** Включена поддержка SharePoint 2007 и Windows SharePoint Services 3.0. Продукт Team Foundation Server 2008 будет поддерживать SharePoint на сервере, отдельном от сервера уровня приложений TFS.
- **Поддержка Windows Server 2008** Поддерживаются Microsoft Windows Server 2008 и Internet Information Services (IIS) 7.0.
- **Поддержка клиентских сертификатов X.509** Для повышения безопасности проверки подлинности поддерживается использование клиентских сертификатов X.509.
- **Синхронизация в больших группах** Благодаря повышенной производительности и надежности появится возможность поддержки большего количества пользователей: приблизительно 30000 и более пользователей на один экземпляр TFS.
- **Поддержка именованных экземпляров SQL** Допускается совместное использование сервера SQL несколькими экземплярами TFS или другими приложениями. Это позволяет различным экземплярам TFS пользоваться одной системой SQL Server 2005.
- **Поддержка нестандартных портов** Позволяет гибче изменять конфигурацию, поддерживая альтернативные веб-сайты и порты.

Сборка

- **Непрерывная сборка** Поддерживается создание триггеров, которые позволяют точно указать, когда следует выполнять непрерывную сборку (Continuous Integration). В частности, можно настроить триггер так, что сборка будет начинаться при каждом возвращении файла после правки. Кроме того, вы можете задать периодическую сборку (rolling build), которая будет начинаться не чаще, чем каждые X минут.
- **Очереди сборок** Поддерживаются очереди сборок и управление ими. Это особенно полезно при использовании непрерывной сборки, поскольку несколько возвращенных после правки файлов способны поставить в очередь несколько сборок.

- **Сборка по расписанию** Поддерживаются плановые сборки, запуск которых можно настроить на определенное время.
- **Управление удалениями** Вы можете настраивать политики, определяющие, когда следует автоматически удалять сборки.
- **Параметры сборки** Можно задавать, какой источник и версию источника следует собирать, а также другие параметры. К вашим услугам — большое количество свойств, допускающих настройку. Кроме того, создавая очереди сборок, вы можете передавать параметры командной строки MSBuild.
- **Расширяемость объектов сборки** Улучшена расширяемость объектов сборки. Теперь у вас есть возможность запускать целевые объекты до и после сборки проекта или решения Visual Studio.
- **Управление сборкой** Позволяет останавливать и удалять сборки в Visual Studio.
- **Конфигурирование сборки** Упрощено указание тестов, которые следует запускать в составе сборки.
- **Гибкость расположения файла сборки проекта** Имеется возможность хранить файл проекта MSBuild (и связанный с ним файл .rsp) в любом месте иерархии управления версиями, а не только в папке **TeamBuildTypes**.
- **Поддержка тестирования пользовательского интерфейса** Допускается запуск в составе сборки графического интерфейса.
- **Политика возврата после правки** Поддерживается новая политика возврата после правки, запрещающая возврат при остановке непрерывной сборки.
- **Управление сервером сборки** Улучшена возможность управления несколькими серверами сборки.
- **Сопоставление рабочих областей** Определение сборки можно сопоставить с «реальной» рабочей областью. Благодаря этому появляется возможность извлекать код из нескольких командных проектов, задавать клиентские сопоставления и пр. Управлять сопоставлениями рабочих папок можно не только в файле workspacemapping.xml, но и посредством графического интерфейса.

Управление версиями

- **Аннотирование** Благодаря аннотированию разработчики могут построчно разобраться, кто изменял разделы кода последним.
- **Folder Diff** Поддерживает сравнение папок для выявления различающихся файлов. Программа Folder Diff способна сравнивать локальные папки между собой, локальные папки с папками на сервере и папки на сервере между собой.

- **Destroy** Поддерживается функция Destroy, удаляющая файлы и папки из системы управления исходным кодом. Удаленные файлы и папки восстановить нельзя.
- **Get Latest On Checkout** Включает возможность загрузки последней версии файла.
- **Сопоставления рабочей области при помощи символов подстановки** Допускает сопоставление папки или файла, находящихся в скрытой папке, а также сопоставление при помощи символов подстановки, позволяющее сопоставлять все файлы в папке, но не сопоставлять вложенные папки.
- **Повышение производительности** Произведены различные улучшения, связанные с производительностью системы управления версиями. Выигрыш для малых серверов или проектов (<10000 файлов) невелик, однако в крупных проектах (где число файлов измеряется в сотнях тысяч) рост эффективности будет значительным.
- **Справка командной строки Team Foundation Server 2008** Поддерживает вывод в командной строке справки по инструменту *tf*. Чтобы получить справку, введите **tf help**. Помощь по отдельным командам вы получите, запустив команду **tf команда /help**.
- **Улучшения автономного режима** Усовершенствован автономный режим работы. Для возвращения в оперативный режим в среду разработки Visual Studio встроена функция *tfpt*.
- **Сбор информации о перезаписи при возвращении файлов после правки** Поддерживает добавление в хранилище информации о перезаписи файлов.

Отслеживание рабочих элементов

- **Улучшения работы с вложениями** Поддерживается добавление вложения способом перетаскивания; допускается выделение нескольких файлов для вложения.
- **Построитель запросов** Усовершенствованы некоторые аспекты построителя запросов:
  - раскрывающиеся фильтры, основанные на текущем проекте;
  - улучшенные списки последних файлов;
  - перетаскивание столбцов;
  - сортировка по нескольким столбцам при помощи мыши и клавиши Shift.

Совместимость с Visual Studio 2005 Team System

В целом, клиент Visual Studio 2008 Team Foundation Server способен работать с сервером Visual Studio 2005 Team Foundation Server, а клиент Visual Studio 2005 способен работать с сервером Visual Studio 2008 Team Foundation Server. Однако есть и некоторые проблемы совместимости:

- **Надстройки Visual Studio** Потребуется повторная компиляция (или изменение политик) клиентских надстроек Visual Studio, поскольку версии файлов сборок объектной модели Team Foundation Server (TFSOM) изменятся, и надстройки придется привязать к другим файлам сборки.
- **Командные сборки** Большинство действий по сборке — например, составление списка определений сборки, запуск и остановка сборки и просмотр отчетов — будут работать при сочетании клиентов и серверов Visual Studio 2005 TFS и Visual Studio 2008. Далее перечислены известные проблемы:

1. Экземпляр Visual Studio 2008 Team Foundation Server будет работать только с сервером сборки Visual Studio 2008 Team Foundation Server.
2. Чтобы клиент Visual Studio 2005 начал сборку на экземпляре сервера Visual Studio 2008 Team Foundation Server, определение сборки должно храниться в папке $/<Проект>/TeamBuildTypes/<имя>.
3. Изменения свойств, внесенные в файл tfsbuild.proj, которые в Team Foundation Server 2008 находятся в БД, перестанут обновляться и синхронизироваться.
4. При работе с непрерывной сборкой в Team Foundation Server 2008 клиент Visual Studio 2005 сможет начать сборку, но не сможет поставить сборку в очередь, просматривать список сборок в очереди, список агентов сборки и пр.
5. На сервере Visual Studio 2005 TFS нельзя создать новый тип сборки с помощью клиента Visual Studio 2008 Team Foundation Server.
6. Параметры в диалоговом окне запуска сборки на Visual Studio 2005 Team Foundation Server невозможно изменить при использовании клиента Visual Studio 2008 Team Foundation Server.

Что меняется по содержанию книги

Руководство по Visual Studio 2005 Team Foundation Server	Руководство по Visual Studio 2008 Team Foundation Server
При раздельном развертывании поддерживаются до 2000 пользователей	При раздельном развертывании поддерживаются до 30000 пользователей

Руководство по Visual Studio 2005 Team Foundation Server	Руководство по Visual Studio 2008 Team Foundation Server
Пользователям нужны корректные доменные учетные записи	Доменные учетные записи более не требуются. Вместо них можно использовать встроенные системные учетные записи, например, Network Service
Для поддержки непрерывной сборки используется нестандартное решение	Для создания и настройки непрерывной и периодической сборки используются триггеры Visual Studio
В составе сборки для определения ее качества могут использоваться автоматизированные тесты	Проще создавать списки тестов для сборки и указывать, какие тесты запускаются на этапе сборки. Есть также возможность проводить тестирование интерфейса в составе автоматизированных тестов сборки
Типы сборок должны находиться в специальной папке, где их сможет найти Team Build	Файлы определений сборки проекта (tfsbuild.proj) могут храниться в любом месте иерархии системы управления версиями
Для настройки сборок по расписанию используются нестандартные решения.	Можно настраивать запланированные сборки Visual Studio, не прибегая к нестандартным решениям
Существует несколько стандартных политик возврата после правки	Существует новая политика возврата после правки для прерванной непрерывной сборки. Она запрещает возврат, пока непрерывная сборка остановлена
Для перехода с VSS на Team Foundation Server используется инструмент converter.exe	Решения переноса и зеркализирования для Team Foundation Server и других систем управления исходным кодом, включая VSS, создаются при помощи инструментария Visual Studio
Для определения набора файлов на локальном компьютере, которые нужно синхронизировать, используется сопоставление рабочей области	Система Team Foundation Server 2008 позволяет сопоставлять папки и файлы, расположенные в скрытой папке, а также с помощью символов подстановки сопоставлять все файлы в папке, при этом не сопоставляя вложенные папки.
Для изменения сопоставления рабочей области используется файл workspacemapping.xml	Для управления сопоставлением рабочей области в Team Foundation Server 2008 используется графический интерфейс. Файл workspacemapping.xml не применяется
Для автономной работы используется инструмент TFS Power Tool	Для автономной работы используется среда разработки Visual Studio

Руководство по Visual Studio 2005 Team Foundation Server	Руководство по Visual Studio 2008 Team Foundation Server
В системе управления исходным кодом получение последней версии файла и загрузка файла для редактирования являются двумя различными операциями	Предусмотрена возможность автоматического получения последней версии файла во время его загрузки для редактирования
При необходимости ссылки на файлы сборки из другого командного проекта нужно настроить предсборочные шаги	Шаблон рабочей области определения сборки управляется с помощью интерфейса VS и обладает той же гибкостью, что и обычная рабочая область, включая сопоставление путей из нескольких командных проектов
Для удаления сборок используется инструмент командной строки TFSBuild	Для остановки и удаления сборок используется среда разработки Visual Studio

Дополнительные ресурсы

Дополнительную информацию о Visual Studio 2008 Team Foundation Server вы найдете в статье «An Overview of Microsoft Visual Studio Code Name “Orcas” White Paper» по адресу *http://go.microsoft.com/?linkid=6625887*.

Руководства

В этой части

- Руководство по Team Build.
- Руководство по управлению проектом.
- Руководство по отчетам.
- Руководство по управлению исходным кодом.

Руководство по Team Build

В этом разделе

Стратегия

- Используйте расписание для регулярного выполнения сборок.
- Используйте непрерывную интеграцию, чтобы обеспечить быструю реакцию на возврат после правки.
- Используйте скользящую сборку, если непрерывная интеграция негативно отражается на производительности сервера сборок.
- Используйте ветвление, чтобы сократить число сбоев при сборке.
- Используйте политики возврата после правки для улучшения качества кода.
- Используйте уведомления, чтобы получать сообщения об окончании сборки.

Ветвление

- Используйте новые типы командной сборки при создании неполной ветви.
- Изменяйте пути к решениям в файлах TFSBuild.proj при создании полной ветви.

Политики возврата после правки

- Используйте политики возврата после правки для улучшения качества кода.
- Используйте политики возврата после правки, чтобы связать рабочие элементы со сборкой.

Непрерывная интеграция

- Используйте непрерывную интеграцию, чтобы обеспечить быструю реакцию на возврат после правки.
- Используйте скользящую сборку, если непрерывная интеграция негативно отражается на производительности сервера сборок.

- Убедитесь, что периодичность сборок значительно превышает продолжительность одной сборки.

Настройка

- Используйте послесборочный шаг для создания проекта установщика.
- Используйте MS Build Toolkit Extras для сборки приложений Microsoft .NET 1.1.
- Используйте TFSBuild.proj для изменения параметров сборки.
- Используйте досборочный шаг для сборки проекта, зависящего от другого проекта.

Развертывание

- В больших командах устанавливайте службы сборки на отдельный сервер.

Производительность

- Повышайте производительность при помощи инкрементных сборок.
- Избегайте синхронизации излишних папок при сборке.
- Используйте рабочие области, чтобы запретить извлечение для правки нежелательных файлов и проектов в Team Build.
- Повышайте производительность за счет использования нескольких компьютеров.

Проекты

- Избегайте перекрестных зависимостей между командными проектами.
- Используйте ссылки на проекты вместо ссылок на файлы.
- Используйте Web Deployment Project для веб-приложений.
- Используйте стратегию единого решения при работе над небольшим командным проектом.
- При работе над большим командным проектом с несколькими независимыми подпроектами разделяйте решение на части.
- Используйте несколько решений при работе над очень большим командным проектом с десятками независимых подпроектов.

Сборки по расписанию

- Используйте расписание для регулярного выполнения сборок.

Тестирование как основа разработки

- Выполняйте анализ кода при каждой сборке.
- Выполняйте автоматизированные тесты при каждой сборке.

- Считайте сборку неудачной, если автоматизированные тесты завершаются с ошибкой.

Рабочие элементы

- Используйте рабочие элементы для отслеживания сбоев сборки.

Стратегия

- Используйте расписание для регулярного выполнения сборок.
- Используйте непрерывную интеграцию, чтобы обеспечить быструю реакцию на возврат после правки.
- Используйте скользящую сборку, если непрерывная интеграция негативно отражается на производительности сервера сборок.
- Используйте ветвление, чтобы сократить число сбоев при сборке.
- Используйте политики возврата после правки для улучшения качества кода.
- Используйте уведомления, чтобы получать сообщения об окончании сборки.

Используйте расписание для регулярного выполнения сборок

Используйте расписание для выполнения сборок на регулярной основе с предсказуемыми интервалами.

Как правило, сборка, выполняемая для группы тестирования и других членов команды, должна производиться безотказно и с фиксированной частотой по времени, чтобы реакция на результаты сборки была своевременной.

Компонент Team Build из комплекта Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) не поддерживает создание сборок по расписанию из пользовательского интерфейса. Однако, чтобы начать сборку в заданное время, вы вольны использовать Microsoft Windows® Task Scheduler для запуска утилиты командной строки TFSBuild.

Создание сборки по расписанию

1. Создайте командную строку TFSBuild:

```
TfsBuild start <<имя сервера сборки>> <<командный проект>> <<тип сборки>>
```

2. Поместите командную строку в пакетный файл.
3. Создайте задачу планировщика Windows, которая будет запускать пакетный файл с нужным интервалом.

Дополнительные ресурсы

- Дополнительную информацию о настройке сборок по расписанию в Team Build вы найдете в главе 9 этой книги.
- Дополнительную информацию о настройке сборок по расписанию в TFS вы найдете в разделе «Как настроить плановую сборку в Visual Studio Team Foundation Server» этой книги.

Используйте непрерывную интеграцию, чтобы обеспечить быструю реакцию на возврат после правки

Используйте непрерывные сборки, чтобы обеспечить оперативное информирование группы разработки о качестве сборки и возможных негативных изменениях после каждого возврата после правки. Это позволит группе разработки быстро устранить проблемы сборки и повысит качество вашего кода.

Хотя Team Foundation Server 2005 не содержит встроенного средства для непрерывной интеграции, в нем есть все необходимое для реализации собственного решения непрерывной сборки.

Дополнительную информацию о настройке непрерывных сборок в TFS вы найдете в разделе «Как настроить непрерывную сборку в Visual Studio Team Foundation Server», где используется решение, представленное группой разработки Microsoft Visual Studio Team System (VSTS). Решение устанавливает веб-службу, которая работает от имени учетной записи, имеющей доступ к серверу TFS. Team Foundation Server позволяет при возникновении определенного события отправлять сообщение электронной почты или вызвать веб-службу. Механизм событий используется решением непрерывной интеграции для регистрации веб-службы, связанной с событием **CheckinEvent**. Всякий раз, когда происходит возврат после правки, веб-служба инициирует запуск Team Build.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 8 этой книги.
- Дополнительную информацию о настройке сборок по расписанию в TFS вы найдете в разделе «Как настроить плановую сборку в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию об использовании решения непрерывной интеграции VSTS вы найдете в статье «Continuous Integration Using Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx*.
- Чтобы загрузить MSI-пакет решения непрерывной интеграции VSTS, перейдите по адресу *http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi*.
- Дополнительную информацию по гибкой разработке ПО и непрерывной интеграции в TFS вы найдете в статье «Extend Team Foundation Server To

Enable Continuous Integration» по адресу *http://msdn.microsoft.com/msdnmag/issues/06/03/TeamSystem/default.aspx*.

Используйте скользящую сборку, если непрерывная интеграция негативно отражается на производительности сервера сборок

Выполнение сборки после каждого возврата — самая простая стратегия непрерывной интеграции, которая обычно обеспечивает самую быструю реакцию. Однако, если возврат после правки производится достаточно часто, она приводит к перегрузке сервера сборок. В этом случае вам следует использовать другой подход, при котором сборка производится после определенного числа возвратов после правки или по истечении определенного времени. Чтобы решить, нужно ли вам использовать скользящую сборку, выясните следующее:

- продолжительность сборки Team Build в минутах;
- средний интервал между последовательными возвратами после правки в минутах;
- промежуток времени, в течение которого возвраты после правки производятся наиболее часто.

Если длительность сборки больше среднего интервала между возвратами после правки, сборки будут выполняться непрерывно: одна сборка еще не будет завершена, когда уже произойдет следующий возврат после правки, инициирующий следующую сборку. Если возвраты после правки производятся до завершения предшествующей сборки, это негативно отражается на производительности сервера сборок и блокирует запуск других сборок (например, сборок по расписанию). Определите промежуток времени, в течение которого возвраты после правки производятся особенно часто, и выясните, будет ли непрерывная интеграция оказывать влияние на сборки по расписанию и другие важные командные сборки.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 8 этой книги.

Используйте ветвление, чтобы сократить число сбоев при сборке

Чтобы сократить количество сбоев при сборке, используйте ветвь **Development** для активной разработки и ветвь **Main** для интеграции.

Ниже приведен пример структуры ветвей после создания ветви **Development**:

- **Development** — ветвь для разработки.
  - **Source**.

- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.

При работе с ветвлением принимайте во внимание следующие рекомендации:

- **Когда следует создавать ветви** Если вы производите ежедневные сборки и у вас возникают проблемы устойчивости и интеграции, создайте основную ветвь и ветвь для разработки, чтобы обеспечить большую предсказуемость ежедневных сборок. Подумайте также о более строгих критериях политик возврата после правки.
- **Когда не следует создавать ветви** Если вы производите только непрерывную интеграцию или у вас нет проблем с устойчивостью ежедневных сборок, дополнительные расходы на создание ветвей от вас, вероятно, не потребуются.
- **Права доступа к ветвям**
  - Права на чтение и запись в ветви **Main** должны предоставляться разработчикам, ответственным за слияние и интеграцию. Всем остальным достаточно доступа только для чтения.
  - Ветвь **Development** должна быть доступна всем как для чтения, так и для записи.
- **Периодичность сборки в ветвях**
  - Ежедневная сборка в ветви **Main**.
  - Непрерывная интеграция в ветви **Development**.
- **Виды тестирования в ветвях**
  - Выполняйте тестирование интеграции, производительности и безопасности в ветви **Main**.
  - Выполняйте тестирование компонентов и оценку качества в ветви **Development**.

Используйте ветвь **Main** для интеграции изменений, произведенных в ветви разработки. Всю активную разработку ведите в ветви **Development**. В ветвь **Main** переносите только проверенные изменения.

Дополнительные ресурсы

- Дополнительную информацию по определению стратегии ветвления и слияния вы найдете в главе 5 этой книги.
- С общими сведениями о ветвлении и слиянии вас познакомит статья «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.

- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о ветвлении и слиянии в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Используйте политики возврата после правки для улучшения качества кода

Чтобы повысить качество возвратов после правки, применяйте сочетание анализа кода и политики тестирования. Например, воспользуйтесь готовой политикой тестирования, чтобы исходный код возвращался в систему управления исходными кодами лишь при условии успешного выполнения определенных тестов. Также вы вольны настроить политику анализа кода, чтобы обеспечить соответствие кода определенным стандартам, и выполнение требований к безопасности, производительности, переносимости, удобству сопровождения и надежности.

Применяя политику возврата после правки в сочетании с политиками, устанавливающими стандарты и правила кодирования, вы протестируете код на предмет наличия определенных проблем.

Чтобы настроить политику анализа кода при возврате после правки для командного проекта, щелкните проект правой кнопкой мыши в **Team Explorer**, выберите **Team Project Settings**, затем щелкните **Source Control**. Перейдите на вкладку **Check-in Policy**, щелкните **Add**, затем выберите и настройте соответствующую политику.

Дополнительные ресурсы

- Дополнительную информацию о создании и применении пользовательских политик возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Чтобы узнать о настройке политики возврата после правки, читайте статью «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- Чтобы посмотреть пример кода, который запрещает возврат правок с определенными вариантами кодирования, читайте статью «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.

- Чтобы посмотреть пример кода, который обязывает вводить комментарии при возврате после правки, читайте статью «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- Чтобы узнать, как зарегистрировать новую политику возврата после правки, читайте статью «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Используйте уведомления, чтобы получать сообщения об окончании сборки

Чтобы отслеживать процесс сборки, создайте уведомление, которое при завершении сборки будет отправлять вам или кому-нибудь еще сообщение электронной почты.

Это важно, поскольку обеспечивает максимальную оперативность рабочего цикла группы разработки. Получив сообщение о завершении сборки, члены группы тестирования смогут приступить к тестам без дополнительных указаний.

Дополнительные ресурсы

- Дополнительную информацию о создании уведомлений вы найдете в статьях «How to: Receive Build Notification E-Mail» по адресу *http://msdn2.microsoft.com/en-us/library/ms181725(VS.80).aspx* и «How to: Add or Edit Alerts» по адресу *http://msdn2.microsoft.com/en-us/library/ms181335(VS.80).aspx*.

Ветвление

- Используйте новые типы командной сборки при создании неполной ветви.
- Изменяйте пути к решениям в файлах TFSBuild.proj при создании полной ветви.

Используйте новые типы командной сборки при создании неполной ветви

Создание ветви с подмножеством решений командного проекта может повлечь за собой создание новых типов сборок.

Существуют неполные ветви двух типов:

- **Ветвь, не включающая ветви любых типов сборки** При этом создается ветвь для решения и исходных файлов, но в новых папках не создаются ветви для каких-либо папок TeamBuildTypes.
- **Ветвь, включающая в себя ветви типов сборки** В дополнение к папкам, содержащим файлы решения и исходных кодов, создаются ветви

подпапок TeamBuildTypes (типов сборки).

Если вы создаете неполную ветвь, не включающую в себя типы командной сборки, все существующие командные сборки сохраняют работоспособностьми, но вам придется создать новые типы командной сборки, чтобы осуществить сборку в ветви. Создавайте новые типы сборки с помощью мастера Team Build Wizard. В новом типе сборки будет указание на новое положение ветви, а также на родительское положение для любых решений, которые должны быть включены в сборку, но при этом не включены в ветвь.

Если вы создадите неполную ветвь, включающую типы командной сборки, скопированные с ветвью типы сборки будут указывать на положение исходной родительской ветви и не позволят собрать новую ветвь. Измените ответвленные типы сборки, чтобы они указывали на новое положение ответвленного кода.

Дополнительные ресурсы

- Дополнительную информацию об обновлении типа сборки вы найдете в статье «How to: Update Build Types on Branched Team Projects» по адресу *http://msdn2.microsoft.com/en-us/library/ms252500(VS.80).aspx*.

Изменяйте пути к решениям в файлах TFSBuild.proj при создании полной ветви

Когда вы создаете новую ветвь, включающую типы командной сборки, пути к типам сборки будут по-прежнему указывать на предыдущее положение. Чтобы сборка работала в новой ветви, вам следует обновить пути к файлам типов сборки проекта: они должны ссылаться на новые пути, созданные после ветвления.

Если создается полная ветвь, при этом также осуществляется ветвление типов сборки. Типы сборки содержат ссылки на папки из первичного дерева управления исходными кодами. Чтобы они ссылались на папки ответвления, отредактируйте ссылки. Извлеките типы сборки из ветви, которую хотите изменить, внесите изменения, а затем верните их в ветвь.

Дополнительные ресурсы

- Дополнительную информацию об обновлении типа сборки вы найдете в статье «How to: Update Build Types on Branched Team Projects» по адресу *http://msdn2.microsoft.com/en-us/library/ms252500(VS.80).aspx*.

Политики возврата после правки

- Используйте политики возврата после правки для улучшения качества кода.
- Используйте политики возврата после правки, чтобы связать рабочие элементы со сборкой.

Используйте политики возврата после правки для улучшения качества кода

Чтобы повысить качество возвратов после правки, применяйте сочетание анализа кода и политики тестирования. Например, воспользуйтесь готовой политикой тестирования, чтобы исходный код возвращался в систему управления исходными кодами лишь при условии успешного выполнения определенных тестов. Также вы вольны настроить политику анализа кода, чтобы обеспечить соответствие кода определенным стандартам, и выполнение требований к безопасности, производительности, переносимости, удобству сопровождения и надежности.

Применяя политику возврата после правки в сочетании с политиками, устанавливающими стандарты и правила кодирования, вы протестируете код на предмет наличия определенных проблем.

Дополнительные ресурсы

- Дополнительную информацию о создании и применении пользовательских политик возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Чтобы узнать о настройке политики возврата после правки, читайте статью «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- Чтобы посмотреть пример кода, который запрещает возврат правок с определенными вариантами кодирования, читайте статью «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- Чтобы посмотреть пример кода, который обязывает вводить комментарии при возврате после правки, читайте статью «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- Чтобы узнать, как зарегистрировать новую политику возврата после правки, читайте статью «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Используйте политики возврата после правки, чтобы связать рабочие элементы со сборкой

Существует политика возврата после правки, которая вынуждает разработчиков связывать возврат после правки с рабочим элементом.

Если при сборке произошел сбой, важно знать, какие наборы изменений связаны с данной сборкой и какие рабочие элементы связаны с этими наборами. Имея такие сведения, вы определите разработчика, ответственного за внесение измененного кода, и область проекта, в которой он работает.

Чтобы сборку можно было связать с набором завершенных рабочих элементов, каждый возврат после правки должен быть связан с рабочим элементом. Возвраты после правки представляют собой наборы изменений, связанных со сборкой, благодаря чему возможно перейти от сборки к набору изменений и соответствующему рабочему элементу.

Дополнительные ресурсы

- Дополнительную информацию о создании и применении пользовательских политик возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Чтобы узнать о настройке политики возврата после правки, читайте статью «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.

Непрерывная интеграция

- Используйте непрерывную интеграцию, чтобы обеспечить быструю реакцию на возврат после правки.
- Используйте скользящую сборку, если непрерывная интеграция негативно отражается на производительности сервера сборок.
- Убедитесь, что периодичность сборок значительно превышает продолжительность одной сборки.

Используйте непрерывную интеграцию, чтобы обеспечить быструю реакцию на возврат после правки

Используйте непрерывные сборки, чтобы обеспечить оперативное информирование группы разработки о качестве сборки и возможных негативных изменениях после каждого возврата после правки. Это позволит группе разработки быстро устранить проблемы сборки и повысит качество вашего кода.

Хотя Team Foundation Server 2005 не содержит встроенного средства для непрерывной интеграции, в нем есть все необходимое для реализации собственного решения непрерывной сборки.

Дополнительную информацию о настройке непрерывных сборок в TFS вы найдете в разделе «Как настроить непрерывную сборку в Visual Studio Team Foundation Server», где используется решение, представленное группой разработки Microsoft Visual Studio Team System (VSTS). Решение устанавливает веб-службу, которая работает от имени учетной записи, имеющей доступ к серверу TFS. Team Foundation Server позволяет при возникновении определенного события отправлять сообщение электронной почты или вызвать веб-службу. Механизм событий используется решением непрерывной интеграции для регистрации веб-службы, связанной с событием **Checkin-**

Event. Всякий раз, когда происходит возврат после правки, веб-служба инициирует запуск Team Build.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 8 этой книги.
- Дополнительную информацию о настройке сборок по расписанию в TFS вы найдете в разделе «Как настроить плановую сборку в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию об использовании решения непрерывной интеграции VSTS вы найдете в статье «Continuous Integration Using Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx*.
- Чтобы загрузить MSI-пакет решения непрерывной интеграции VSTS, перейдите по адресу *http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi*.
- Дополнительную информацию по гибкой разработке ПО и непрерывной интеграции в TFS вы найдете в статье «Extend Team Foundation Server To Enable Continuous Integration» по адресу *http://msdn.microsoft.com/msdnmag/issues/06/03/TeamSystem/default.aspx*.

Используйте скользящую сборку, если непрерывная интеграция негативно отражается на производительности сервера сборок

Выполнение сборки после каждого возврата — самая простая стратегия непрерывной интеграции, которая обычно обеспечивает самую быструю реакцию. Однако, если возврат после правки производится достаточно часто, она приводит к перегрузке сервера сборок. В этом случае вам следует использовать другой подход, при котором сборка производится после определенного числа возвратов после правки или по истечении определенного времени. Чтобы решить, нужно ли вам использовать скользящую сборку, выясните следующее:

- продолжительность сборки Team Build в минутах;
- средний интервал между последовательными возвратами после правки в минутах;
- промежуток времени, в течение которого возвраты после правки производятся наиболее часто.

Если длительность сборки больше среднего интервала между возвратами после правки, сборки будут выполняться непрерывно: одна сборка еще не будет завершена, когда уже произойдет следующий возврат после правки, инициирующий следующую сборку. Если возвраты после правки производятся до завершения предшествующей сборки, это негативно отражается на производительности сервера сборок и заблокирует запуск других сборок

(например, сборок по расписанию). Определите промежуток времени, в течение которого возвраты после правки производятся особенно часто, и выясните, будет ли непрерывная интеграция оказывать влияние на сборки по расписанию и другие важные командные сборки.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 8 этой книги.

Убедитесь, что периодичность сборок значительно превышает продолжительность одной сборки

Важно определить интервал скользящей сборки, чтобы обеспечить эффективность процесса. Если интервал между сборками превышает время, за которое выполняется одна сборка, в промежутках между скользящими сборками сервер будет доступен для сборок других типов.

Чтобы определить идеальный интервал времени скользящей сборки, разделите среднюю частоту возвратов после правки на длительность сборки. Например, если сборка занимает 10 минут, а возвраты после правки происходят в среднем раз в 5 минут, можно задать выполнение сборки после двух возвратов после правки, а для времени ожидания выбрать значение 10 минут. Это гарантирует завершение одной сборки до начала следующей. Если нагрузка на сервер сборки возросла, увеличьте эти значения.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 8 этой книги.

Настройка

- Используйте послесборочный шаг для создания проекта установщика.
- Используйте MS Build Toolkit Extras для сборки приложений Microsoft .NET 1.1.
- Используйте TFSBuild.proj для изменения параметров сборки.
- Используйте досборочный шаг для сборки проекта, зависящего от другого проекта.

Используйте послесборочный шаг для создания проекта установщика

Поскольку Team Build по умолчанию не поддерживает проекты установки, для компиляции проекта установщика и копирования исполняемых файлов в общее место накопления вам следует использовать пользовательский послесборочный шаг.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Walkthrough: Configuring Team Foundation Build to Build a Visual Studio Setup Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms404859(VS.80).aspx*.

Используйте MS Build Toolkit Extras для сборки приложений Microsoft .NET 1.1

Team Build по умолчанию не поддерживает приложения .NET 1.1. Производить сборки .NET 1.1 позволяет пакет MSBuild Extras – Toolkit for .NET 1.1 (MSBee), но при этом потребуется обновление проектов и решений до Visual Studio 2005. Если это невозможно, используйте для компиляции приложений .NET 1.1 пользовательский послесборочный шаг.

Дополнительные ресурсы

- Чтобы загрузить MSBee, обратитесь по адресу *http://www.codeplex.com/MSBee*.
- Дополнительную информацию о создании послесборочных действий для сборки приложений .NET 1.1 вы найдете в блоге по адресу *http://blogs.msdn.com/nagarajp/archive/2005/10/26/485368.aspx*.

Используйте TFSBuild.proj для изменения параметров сборки

Чтобы изменить параметры сборки, например, сервер сборки, место накопления или папку сборки, отредактируйте файл TFSBuild.proj.

В файле TFSBuild.proj содержится множество сведений, необходимых для работы Team Build. Здесь, в частности, задается, должны ли при сборке выполняться статический анализ кода и модульные тесты. Чтобы изменить параметры сборки, отредактируйте файл TFSBuild.proj.

Редактирование файла TFSBuild.proj

1. Извлеките файл для правки.
2. Отредактируйте параметры сборки в файле.
3. Возвратите файл в систему.

При следующем выполнении сборки будет использован файл с внесенными изменениями.

Дополнительные ресурсы

- Дополнительную информацию о настройке Team Foundation Build вы найдете в статье «Customizing Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx*.

Используйте досборочный шаг для сборки проекта, зависящего от другого проекта

Team Build не поддерживает сборку решений, охватывающих несколько командных проектов. Чтобы выполнить такую сборку, вам следует настроить файл TFSBuild.proj на извлечение для правки кода из других проектов, от которого зависит ваша сборка.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.
- Дополнительную информацию о пользовательской настройке Team Foundation Build вы найдете в статье «Customizing Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx*.

Развертывание

- В больших командах устанавливайте службы сборки на отдельный сервер.

В больших командах устанавливайте службы сборки на отдельный сервер

Большие командные сборки могут длиться продолжительное время и занимать значительные ресурсы сервера. Если вы выполняете сборки на командном сервере TFS, это отражается на его надежности, производительности и масштабируемости. Для повышения производительности сборки и снижения нагрузки на уровень приложений рекомендуется выполнять сборки на выделенном сервере.

Дополнительные ресурсы

- Чтобы загрузить Team Foundation Installation Guide или получить дополнительные сведения по установке TFS и Team Build, обращайтесь по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyId=E54BF6FF-026B-43A4-ADE4-A690388F310E&displaylang=en*.

Производительность

- Повышайте производительность при помощи инкрементных сборок.
- Избегайте синхронизации излишних папок при сборке.
- Используйте рабочие области, чтобы запретить извлечение для правки нежелательных файлов и проектов в Team Build.
- Повышайте производительность за счет использования нескольких компьютеров.

Повышайте производительность при помощи инкрементных сборок

По умолчанию перед извлечением дерева исходного кода, необходимого для сборки, Team Build очищает папку, используемую для выполнения сборки. Кроме того, Team Build удаляет и повторно инициализирует рабочую область, используемую для извлечения исходного кода. Если объем исходного кода, необходимого для сборки, достаточно велик и сервер сборки установлен отдельно от TFS-сервера, извлечение исходного кода может занять много времени. Чтобы повысить производительность, настройте Team Build так, чтобы извлекался только тот исходный код, который изменился с момента последней сборки. Для этого вам потребуется установить несколько значений в файле TFSBuild.proj:

- остановите очистку компонентом Team Build локальной папки сборки и папки исходных кодов;
- остановите воссоздание компонентом Team Build рабочей области, используемой при сборке;
- настройте Team Build на получение из системы управления исходным кодом только изменившихся исходных кодов.

Выполнение инкрементной сборки

1. Создайте новый тип сборки, который будет представлять инкрементную сборку.
2. Извлеките для правки файл TFSBuild.proj, связанный со вновь созданным типом инкрементной сборки.
3. Добавьте в TFSBuild.proj следующий раздел перед закрывающим элементом </project>:

```
<PropertyGroup>
   <SkipClean>true</SkipClean>
   <SkipInitializeWorkspace>true</SkipInitializeWorkspace>
   <ForceGet>false</ForceGet>
</ PropertyGroup>
```

Эти настройки приводят к следующим результатам:

- **SkipClean** Значение **true** этого параметра гарантирует, что локальные папки сборки и исходных кодов не будут очищаться при сборке.
- **SkipInitializeWorkspace** Значение **true** этого параметра означает, что сборка на затронет текущую рабочую область для компьютера сборки.
- **ForceGet** Присвоение значения **false** этому параметру приводит к тому, что при сборке будут извлекаться не все исходные коды рабочей области, а только измененные коды.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в разделе «Практические рекомендации: Team Build» этой книги.
- Дополнительную информацию о конфигурировании инкрементной сборки вы найдете в статье «How to: Configure Team Foundation Build for an Incremental Build" по адресу *http://msdn2.microsoft.com/en-us/library/aa833876(VS.80).aspx*.

Избегайте синхронизации излишних папок при сборке

Вам следует сократить масштаб сопоставления рабочей области или скрыть папки, которые не нужны при сборке.

Когда вы запускаете Team Build, сервер получает все необходимые ему файлы из системы управления исходным кодом. Список получаемых файлов определяется рабочей областью, которая используется для создания типа командной сборки. Но некоторые файлы, сопоставленные в рабочей области, на самом деле могут быть не нужны. Измените определение рабочей области, чтобы сократить число включаемых папок или скрыть ненужные файлы, чтобы они не извлекались как часть сборки.

Например, по умолчанию новый проект сопоставляется с папкой $/TeamProject. Если все файлы исходных кодов находятся в папке $/TeamProject/foo/bar/foobar/sources, вам следует сопоставлять только эту папку.

Чтобы скрыть файлы под сопоставлением рабочей области, отредактируйте файл WorkspaceMapping.xml, формируемый при создании типа командной сборки и применяемый для определения папок, которые извлекаются при выполнении сборки. Скрывать можно как файлы, так и папки, но предпочтительнее скрывать папки, поскольку сокрытие отдельных файлов может привести к излишним накладным расходам.

Сокрытие папки

1. Извлеките для правки файл WorkspaceMapping.xml.
2. Добавьте в файл записи о сокрытии.
3. Верните файл WorkspaceMapping.xml в систему управления исходным кодом.

В следующем примере показано, как отменить извлечение из системы управления исходным кодом папки с документацией:

```
<Mappings>
   <InternalMapping ServerItem="$/MyTeamProject" LocalItem="c:\projects\
teamproject" Type="Map" />
   <InternalMapping ServerItem="$/MyTeamProject/documentation" Type="Cloak" />
</Mappings>
```

Дополнительные ресурсы

- Дополнительную информацию о сокрытии папок в рабочей области вы найдете в статье «How to: Cloak and Decloak Folders in a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181378(VS.80).aspx*.
- Дополнительную информацию об исключении папок при командной сборке вы найдете в статье «How to make 'Team Build' skip getting certain folders?» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/10/13/480584.aspx*.
- Дополнительную информацию о схеме WorkspaceMapping вы найдете в статье «Schema for the WorkspaceMapping.xml file» по адресу *http://blogs.msdn.com/buckh/archive/2007/02/28/schema-for-the-workspacemapping-xml-file.aspx*.

Используйте рабочие области, чтобы запретить извлечение для правки нежелательных файлов и проектов в Team Build

Выполняя сборку, Team Build извлекает исходные коды из системы управления. Если дерево исходных кодов проекта велико, извлечение исходных кодов может оказаться длительным процессом. Если вы собираете только часть проекта, убедитесь, что извлекаются только необходимые файлы.

Большой командный проект, как правило, содержит несколько решений Visual Studio, каждое из которых используется для сборки отдельных частей проекта. Создавая тип командной сборки, вы указываете решение, которое будет использоваться при сборке. Если вы зададите файл решения без указания рабочей области, перед выполнением сборки Team Build извлечет все исходные коды командного проекта.

Чтобы извлечь только необходимые коды, сначала определите рабочую область и создайте в ней сопоставление только для того решения, которое хотите собрать. Затем определите тип командной сборки, выберите определенную вами рабочую область и укажите решение. При таком подходе извлечены будут только исходные коды, определенные в этой рабочей области.

Дополнительные ресурсы

- Дополнительную информацию о создании типа командной сборки вы найдете в статье «Walkthrough: Creating a Build Type in Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms181286(VS.80).aspx*.
- Подробнее о том, почему Team Build извлекает для правки весь исходный код рабочей области, читайте в статье «Why does Team Build sync all sources in spite of my selecting only a subset of solutions?» по адресу *http://blogs.msdn.com/anutthara/archive/2005/12/07/500923.aspx*.

Повышайте производительность за счет использования нескольких компьютеров

Если сборки несколько типов выполняются на единственном сервере, он может оказаться перегружен. В такой ситуации вам следует подумать о выполнении различных типов сборки на различных серверах.

Сборка может занимать продолжительное время, особенно в больших проектах. Если вы используете непрерывную интеграцию или частые сборки по расписанию, сервер может не справиться с объемом производимых работ.

Для распределения нагрузки установите несколько серверов сборки на различные компьютеры. Назначьте для каждого сервера различные типы сборки, чтобы сбалансировать нагрузку.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 7 этой книги.

Проекты

- Избегайте перекрестных зависимостей между командными проектами.
- Используйте ссылки на проекты вместо ссылок на файлы.
- Используйте Web Deployment Project для веб-приложений.
- Используйте стратегию единого решения при работе над небольшим командным проектом.
- При работе над большим командным проектом с несколькими независимыми подпроектами разделяйте решение на части.
- Используйте несколько решений при работе над очень большим командным проектом с десятками независимых подпроектов.

Избегайте перекрестных зависимостей между командными проектами

Вообще говоря, следует избегать перекрестных зависимостей между командными проектами. Старайтесь сохранять все связанные и зависимые решения и проекты в одном командном проекте. Это сократит необходимость настройки сценария сборки. Если у вас есть зависимость, используйте ссылки проекта для ее определения или создайте ветвь из общего проекта в ваш проект. Избегайте файловых ссылок, поскольку ими намного труднее управлять.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.

- Дополнительную информацию об изменении рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.

Используйте ссылки на проекты вместо ссылок на файлы

Чтобы сослаться на другой файл сборки .NET в том же решении Visual Studio, используйте ссылку на проект Visual Studio. Используя ссылки на проект, вы даете Visual Studio возможность выполнять некоторые вещи автоматически, например, синхронизировать конфигурацию сборки (debug или release), отслеживать версии, при необходимости повторно собирать компоненты в случае изменения версии файла сборки .NET.

Дополнительные ресурсы

- Дополнительную информацию о ссылках на проекты вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.

Используйте Web Deployment Project для веб-приложений

Проекты веб-развертывания связываются с проектами Visual Studio Web Site или Web Application. Они позволяют управлять настройками сборки, а также множеством других настроек, общих для веб-приложений ASP.NET. Например, проекты развертывания предоставляют удобный доступ к файлу Web.config, строкам соединения, виртуальным папкам, а также позволяют с легкостью развертывать скомпилированные веб-приложения на сервере хостинга.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Visual Studio 2005 Web Deployment Projects» по адресу *http://msdn2.microsoft.com/en-us/asp.net/aa336619.aspx*.

Используйте стратегию единого решения при работе над небольшим командным проектом

Если вы работаете в небольшой группе, вам стоит использовать стратегию единого решения Visual Studio, содержащего все проекты. Эта структура упрощает разработку, поскольку при открытии решения сразу доступен весь код. При такой стратегии также легко устанавливать ссылки, поскольку все

они связывают проекты одного решения. Вы вольны использовать и файловые ссылки, чтобы ссылаться на сборки сторонних производителей, например, приобретенные компоненты, которые находятся вне вашего решения.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 3 этой книги.

При работе над большим командным проектом с несколькими независимыми подпроектами разделяйте решение на части

Если вы работаете в большой группе, вам выгодно использовать несколько решений, каждое из которых представляет определенную подсистему приложения. Разработчики будут применять эти решения для работы над мелкими частями системы без необходимости загрузки всего кода по всем проектам. Проектируйте структуру решения так, чтобы любые проекты, связанные зависимостями, группировались вместе. Это позволит вам использовать ссылки на проекты вместо ссылок на файлы. Подумайте о создании главного решения, которое содержало бы все проекты, для сборки всего приложения.

> **Примечание** Если вы осуществляете сборку при помощи Team Build (на основе MSBuild), можете создавать решения, не включающие в себя все связанные проекты. При условии что сначала вы сбираете решение целиком, генерируя двоичный файл для каждого решения, MSBuild сможет проследовать по ссылкам в проекты вне решения и произвести успешную сборку. Решения, создаваемые таким образом, не будут собираться при помощи команды сборки из Visual Studio. Они будут работать только с Team Build и MSBuild.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 3 этой книги.

Используйте несколько решений при работе над очень большим командным проектом с десятками независимых подпроектов

Работая над очень большим решением, для которого требуется несколько десятков проектов, вы рискуете столкнуться с ограничениями масштабируемости. Если это случилось, разбейте приложение на несколько решений, но не создавайте главное решение для всего приложения. Все ссылки внутри каждого решения будут ссылками на проекты. Ссылки на проекты вне решения (например, на библиотеки сторонних разработчиков в другом решении) будут ссылками на файлы. Это означает, что главного решения быть не может. Вместо этого нужно использовать сценарий, в котором учтен порядок

сборки решений. Одной из задач обслуживания структуры из нескольких решений является контроль за тем, чтобы разработчики по невнимательности не создали кольцевых ссылок между решениями. Такая структура требует создания сложного сценария сборки и явного сопоставления отношений зависимости. Собрать приложение во всей его полноте из Visual Studio невозможно. Вместо этого вам придется использовать TFS Team Build или непосредственно MSBuild.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 3 этой книги.

Сборки по расписанию

- Используйте расписание для регулярного выполнения сборок.

Используйте расписание для регулярного выполнения сборок

Используйте расписание для выполнения сборок на регулярной основе с предсказуемыми интервалами.

Как правило, сборка, выполняемая для группы тестирования и других членов команды, должна производиться безотказно и с фиксированной частотой по времени, чтобы реакция на результаты сборки была своевременной.

Компонент Team Build из комплекта Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) не поддерживает создание сборок по расписанию из пользовательского интерфейса. Однако, чтобы начать сборку в заданное время, вы вольны использовать Microsoft Windows® Task Scheduler для запуска утилиты командной строки TFSBuild.

Создание сборки по расписанию

1. Создайте командную строку TFSBuild:

```
TfsBuild start <<имя сервера сборки>> <<командный проект>> <<тип сборки>>
```

2. Поместите командную строку в пакетный файл.
3. Создайте задачу планировщика Windows, которая будет запускать пакетный файл с нужным интервалом.

Дополнительные ресурсы

- Дополнительную информацию о настройке сборок по расписанию в Team Build вы найдете в главе 9 этой книги.
- Дополнительную информацию о настройке сборок по расписанию в TFS вы найдете в разделе «Как настроить плановую сборку в Visual Studio Team Foundation Server» этой книги.

Тестирование как основа разработки

- Выполняйте анализ кода при каждой сборке.
- Выполняйте автоматизированные тесты при каждой сборке.
- Считайте сборку неудачной, если автоматизированные тесты завершаются с ошибкой.

Выполняйте анализ кода при каждой сборке

Для повышения качества сборки используйте анализ кода. Настройте политику анализа кода, чтобы гарантировать его соответствие определенным стандартам и правилам безопасности, производительности, переносимости, удобства сопровождения и надежности.

Чтобы включить анализ кода для типа сборки, установите флажок **code analysis** в мастере Team Build Type при создании нового типа командной сборки или отредактируйте файл TFSBuild.proj для существующего типа командной сборки.

Включение анализа кода в файле TFSBuild.proj

- Чтобы анализ кода выполнялся для всех проектов независимо от настроек проекта, измените значение тега **<RunCodeAnalysis>** на **Always**.
- Чтобы анализ кода выполнялся лишь при условии соответствующей настройки параметров проекта, измените значение тега **<RunCodeAnalysis>** на **Default**.

Дополнительные ресурсы

- Дополнительную информацию об автоматическом анализе кода в ходе сборки вы найдете в разделе «Как автоматически выполнять анализ кода при помощи Team Build» этой книги.
- Дополнительную информацию об инструментах анализа кода вы найдете в статье «Guidelines for Using Code Analysis Tools» по адресу *http://msdn2.microsoft.com/en-us/library/ms182023(VS.80).aspx*.

Выполняйте автоматизированные тесты при каждой сборке

Выполняйте автоматизированные тесты после каждой сборки, чтобы оперативно получать сведения о ее качестве. Чтобы создать список тестов, связанных со сборкой, необходимо установить Visual Studio Test Edition или Visual Studio Team Suite. Чтобы выполнять автоматизированные тесты на сервере сборки, на нем нужно установить Visual Studio Developer Edition, Visual Studio Test Edition или Visual Studio Team Suite.

Автоматизированные тесты как часть сборки Team Build

1. Создайте один или несколько автоматизированных тестов, которые хотите выполнять при сборке.
2. При помощи Test Manager создайте список тестов.
3. В Test Manager заполните новый список тестов, перетаскивая в него тесты из Test View.
4. Создайте новый тип командной сборки.
5. Установите флажок запуска автоматизированных тестов.
6. Выделите тестовый проект, в котором созданы тесты и список тестов.
7. Выберите список тестов, которые хотите выполнить.

Дополнительные ресурсы

- Дополнительную информацию об автоматическом выполнении тестов сборки вы найдете в статье «How to: Configure and Run Build Verification Tests (BVTs)» по адресу *http://msdn2.microsoft.com/en-us/library/ms182465(VS.80).aspx*.
- Дополнительную информацию о выполнении автоматизированных тестов без Visual Studio Test Edition или файла VSMDI вы найдете в статье «How to run tests in a build without test metadata files and test lists (.vsmdi files)» по адресу *http://blogs.msdn.com/buckh/archive/2006/11/04/how-to-run-tests-without-test-metadata-files-and-test-lists-vsmdi-files.aspx*.

Считайте сборку неудачной, если автоматизированные тесты завершаются с ошибкой

Когда сборка завершается неудачно из-за ошибок компиляции, для отслеживания ошибки создается рабочий элемент, а сама сборка помечается как неудачная. Однако сборка не считается неудачной, если не удается пройти автоматизированные тесты. Ошибка теста преобразуется в предупреждение, а сборка продолжается.

Иногда нужно, чтобы сборка считалась неудачной, если автоматизированные тесты завершаются с ошибкой. Также может понадобиться, чтобы для отслеживания неудачного выполнения теста автоматически создавался рабочий элемент.

Настройка сбоя сборки при завершении теста с ошибкой

1. Откройте файл Microsoft.TeamFoundation.Build.targets из папки Program Files\MSBuild\Microsoft\VisualStudio\v8.0\TeamBuild.
2. Извлеките для правки файл TFSBuild.proj для типа командной сборки, который хотите считать неудачным при завершении теста с ошибкой.
3. Скопируйте объект **RunTestWithConfiguration** из файла Microsoft.TeamFoundation.Build.targets в конец файла TFSBuild.proj перед закрывающим тегом **</Project>**.

4. Измените атрибут **ContinueOnError** с **true** на **false.**

> **Примечание** Вам доступны две задачи тестирования. Измените задачу серверной сборки, чтобы изменить поведение сборок только на сервере сборки. Задача клиентской сборки используется при выполнении сборки на компьютере разработчика.

5. Если вы хотите создать рабочий элемент при неудачном завершении сборки, измените **RunTestWithConfiguration,** добавив элемент **OnError** перед закрывающим тегом **</Target>** элемента **OnError**:

```
<OnError ExecuteTargets="CreateWorkItem;"/>
```

Если вы хотите, чтобы сборка всегда считалась неудачной при завершении теста с ошибкой, измените непосредственно Microsoft.TeamFoundation.Build.targets. Это повлияет на поведение всех типов командной сборки.

Рекомендованное выше решение легко реализуется, но нет гарантии, что оно будет работать в будущих версиях Visual Studio. Чтобы реализовать решение, которое обязательно будет работать после обновления, читайте статью «Determining Whether Tests Passed in Team Build» в блоге Аарона Холлберга (Aaron Hallberg) по адресу *http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether-tests-passed-in-team-build.aspx*.

Дополнительные ресурсы

- Дополнительную информацию о настройке сборки для создания рабочих элементов при завершении теста с ошибкой вы найдете в статье «Create Workitems for Test Failures in TeamBuild» по адресу *http://blogs.msdn.com/nagarajp/archive/2005/10/14/481290.aspx*.
- Дополнительную информацию о решении, которое будет работать после обновления Visual Studio, читайте статью «Determining Whether Tests Passed in Team Build» в блоге Аарона Халлберга (Aaron Hallberg) по адресу *http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether-tests-passed-in-team-build.aspx*.

Рабочие элементы

- Используйте рабочие элементы для отслеживания сбоев сборки.

Используйте рабочие элементы для отслеживания сбоев сборки

Если работа Team Build завершается с ошибкой, для отслеживания этой ошибки автоматически создается рабочий элемент. По умолчанию ему назначен статус «Active», а заголовок информирует о том, что произошла ошибка сборки. Вы должны назначить этот рабочий элемент ответственно-

му разработчику или руководителю сборки, чтобы исправить ошибку и разрешить проблему.

Задача сборки в TFSBuild.proj, определяющая этот рабочий элемент, выглядит следующим образом:

```
<!-Создание рабочего элемента для сбоя сборки -->
   <CreateNewWorkItem
   BuildId="$(BuildNumber)"
   Description="$(WorkItemDescription)"
   TeamProject="$(TeamProject)"
   TeamFoundationServerUrl="$(TeamFoundationServerUrl)"
   Title="$(WorkItemTitle)"
   WorkItemFieldValues="$(WorkItemFieldValues)"
   WorkItemType="$(WorkItemType)"
   ContinueOnError="true" />
```

Чтобы настроить созданный рабочий элемент по себя (например, назначить определенного разработчика, установить степень серьезности или приоритет), измените поле WorkItemFieldValues.

Дополнительные ресурсы

- Дополнительную информацию о настройке рабочих элементов ошибки сборки вы найдете в статье «Team Foundation Build Tasks» по адресу *http://msdn2.microsoft.com/en-us/library/ms243778(vs.80).aspx*.

Дополнительные ресурсы по сборке

- Дополнительные сведения о командных сборках вы найдете в статье «Overview of Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms181710(VS.80).aspx*.

Руководство по управлению проектом

В этом разделе

Области и итерации

- Используйте области для улучшения контроля за проектом.
- Представляйте вехи проекта с помощью итераций.
- Создавайте отдельные итерации для неназначенных сценариев и задач.
- Определите продолжительность цикла итерации.

Политики возврата после правки

- Повышайте качество кода при помощи политики возврата после правки.
- Обязывайте разработчиков связывать рабочие элементы с возвратом после правки при помощи политики.
- Используйте политики возврата для соблюдения стандартов программирования.
- Настраивайте уведомления о перекрытии политик возврата после правки.

Шаблоны процесса

- Используйте шаблон процесса MSF Agile в несложных проектах, допускающих неформальный подход.
- Используйте шаблон процесса MSF CMMI в проектах, требующих более формального подхода или соответствия стандартам CMMI.
- По возможности используйте минимальный шаблон процесса.
- Отредактируйте существующий шаблон в соответствии с потребностями вашей команды.

Группы безопасности и разрешения

- Создавайте группы безопасности для назначения определенного набора разрешений.

- Распределяйте членов команды по соответствующим группам.

Командные проекты

- Создавайте один командный проект на приложение, чтобы переносить рабочие элементы и другие ресурсы из версии в версию.
- Создавайте один командный проект на версию, если хотите создавать новые рабочие элементы и прочие ресурсы для каждой версии.
- Не увлекайтесь предоставлением разрешений на доступ к ресурсам проекта.
- Структурируйте дерево исходного кода с учетом ветвления.

Рабочие элементы

- Создавайте сценарии в начале работы над проектом.
- Правильно определяйте требования QoS.
- Разделяйте сценарии на управляемые задачи.
- Разрабатывайте критерии приемки для каждой задачи.
- Связывайте требования и задачи со сценариями.
- Используйте Microsoft Excel для массового редактирования рабочих элементов.

Области и итерации

- Используйте области для улучшения контроля за проектом.
- Представляйте вехи проекта с помощью итераций.
- Создавайте отдельные итерации для неназначенных сценариев и задач.
- Определите продолжительность цикла итерации.

Используйте области для улучшения контроля за проектом

Используйте области (area) командного проекта для упорядочения задач, ошибок, требований и других рабочих элементов. Разрешения для областей позволяют ограничить доступ к различным частям командного проекта.

Области применяются для представления логических или физических компонентов, а подобласти (sub-area) представляют отдельные функции. Такая структура помогает упорядочить рабочие элементы и упрощает отслеживание работ по компоненту или функции.

Создание области проекта

1. Щелкните проект в окне Team Explorer.
2. В меню **Team** раскройте подменю **Team Project Settings** и выберите команду **Areas and Iterations**.

3. В диалоговом окне **Areas and Iterations** перейдите на вкладку **Area**.
4. Щелкните кнопку **Add a child node** на панели инструментов.
5. Щелкните правой кнопкой новый узел, выберите команду **Rename** и введите нужное имя.
6. Щелкните узел **Area**.
7. Повторяйте шаги 2, 3 и 4 для создания дополнительных областей и иерархической структуры проекта.

Остерегайтесь создания слишком сложной структуры. Области позволяют назначать разные разрешения на доступ к рабочим элементам, однако управление этими разрешениями в сложных деревьях связано с дополнительными расходами. Кроме того, сложную структуру с разветвленными разрешениями сложнее копировать в другие командные проекты.

Дополнительные ресурсы

- Дополнительную информацию об использовании областей вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.

Представляйте вехи проекта с помощью итераций

Используйте итерации, чтобы определить, сколько раз в ходе разработки приложения команда будет повторять определенный набор масштабных действий (планирование, выполнение или тестирование). Этот набор должен представлять веху проекта, имеющую четкий критерий завершения, например, готовность функции или компонента.

Создание итерации

1. Щелкните проект в окне Team Explorer.
2. В меню **Team** раскройте подменю **Team Project Settings** и выберите команду **Areas and Iterations**.
3. В диалоговом окне **Areas and Iterations** перейдите на вкладку **Iteration**.
4. Щелкните кнопку **Add a child node** на панели инструментов.
5. Щелкните правой кнопкой новый узел, выберите команду **Rename** и введите имя.
6. Щелкните узел **Iteration**.
7. Повторите шаги 2, 3 и 4 для создания других итераций.
8. Щелкните **Close**.

> **Примечание** В шаблон процесса MS Agile включено три предопределенных итерации. Вы можете удалить эти итерации, переименовать их или оставить неизменными.

Дополнительные ресурсы

- Дополнительную информацию об использовании итераций вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.

Создавайте отдельные итерации для неназначенных сценариев и задач

Создайте отдельную итерацию для сценариев и задач, которые до этих пор не были назначены ни одной итерации. Это позволит при планировании итераций легко распознать незавершенные сценарии и задачи.

Создание отдельной итерации

1. Щелкните командный проект в окне Team Explorer.
2. В меню **Team** раскройте подменю **Team Project Settings** и выберите команду **Areas and Iterations**.
3. В диалоговом окне **Areas and Iterations** перейдите на вкладку **Iteration**.
4. Щелкните кнопку **Add a child node** на панели инструментов.
5. Щелкните правой кнопкой новый узел, выберите команду **Rename** и введите нужное имя.
6. Щелкните **Close**.

Дополнительные ресурсы

- Дополнительную информацию о создании итераций вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.

Определите продолжительность цикла итерации

В процессе настройки командного проекта определите продолжительность цикла итерации, исходя из размера и сложности проекта. Учитывайте следующие ключевые моменты:

- Цикл итерации должен быть достаточно продолжительным, чтобы позволить членам команды завершить основную часть работы, и охватывать несколько различных сценариев.
- Цикл итерации должен быть достаточно коротким, чтобы позволить гибко реагировать на изменения в ситуации и приоритетах.

На практике в большинстве командных проектов работает двухнедельный цикл итерации.

Дополнительные ресурсы

- Дополнительную информацию о продолжительности цикла итераций вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.
- Подробнее о продолжительности цикла итерации — в главе 11.

Политики возврата после правки

- Повышайте качество кода при помощи политики возврата после правки.
- Обязывайте разработчиков связывать рабочие элементы с возвратом после правки при помощи политики.
- Используйте политики возврата для соблюдения стандартов программирования.
- Настраивайте уведомления о перекрытии политик возврата после правки.

Повышайте качество кода при помощи политики возврата после правки

Сочетая политики анализа и тестирования, вы обеспечите соблюдение стандартов качества кода. Например, встроенная в VSTS политика тестирования обеспечит обязательное проведение специальных тестов перед возвратом кода в системе управления исходным кодом Microsoft® Visual Studio® 2005 Team Foundation Server (TFS). Также можно настроить политику анализа кода, обеспечив с ее помощью соответствие кода определенным нормам безопасности, производительности, переносимости, удобства сопровождения и надежности.

Применяя этот тип политики возврата после правки в дополнение к политикам, направленным на укрепление стандартов и нормативов программирования, вы гарантируете соответствие кода специфическим критериям качества.

Применение анализа кода в командном проекте

1. В окне Team Explorer щелкните правой кнопкой нужный командный проект, раскройте подменю **Team Project Settings** и выберите команду **Source Control**.
2. Перейдите на вкладку **Check-in Policy**, щелкните кнопку **Add**, а затем выберите и настройте соответствующую политику.

Дополнительные ресурсы

- Дополнительную информацию о создании и использовании пользовательской политики возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.

Обязывайте разработчиков связывать рабочие элементы с возвратом после правки при помощи политики

Используйте новую стандартную политику возврата после правки **Work Item**, чтобы заставить разработчиков связывать с возвратом рабочие элементы. В случае ошибки сборки важно уметь определить связь между сборкой, набором изменений, внесенными в исходный код, и рабочими элементами, входящими в этот набор. Это позволяет установить разработчика, ответственного за редактирование данного кода и область проекта, в которой он работает.

Настройка политики Work Items

1. В окне Team Explorer правой кнопкой мыши щелкните нужный командный проект, раскройте подменю **Team Project Settings** и выберите команду **Source Control**.
2. Перейдите на вкладку **Check-in Policy**.
3. Щелкните кнопку **Add**, а затем выделите и настройте политику **Work Item**.

Дополнительные ресурсы

- Дополнительную информацию о возврате после правки вы найдете в статье «How to: Check In Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181411(VS.80).aspx*.
- Дополнительную информацию о рабочих элементах и незавершенных изменениях вы найдете в статье «How to: Associate Work Items with Changesets» по адресу *http://msdn2.microsoft.com/en-us/library/ms181410(VS.80).aspx*.

Используйте политики возврата для соблюдения стандартов программирования

Проект, над которым вы работаете, может требовать соблюдения определенных стандартов программирования, не охватываемых статическим анализом кода и существующими политиками возврата после правки. Например, в проекте может требоваться полное отсутствие в коде символов табуляции или обязательное добавление комментария при возврате кода. В подобных случаях создавайте новые политики возврата.

Применение политики для соблюдения стандартов программирования

1. В окне Team Explorer правой кнопкой мыши щелкните нужный командный проект, раскройте подменю **Team Project Settings** и щелкните команду **Source Control**.

2. Перейдите на вкладку **Check-in Policy** и щелкните кнопку **Add**.
3. В диалоговом окне **Add Check-in Policy** установите параметр **Code Analysis** и щелкните **OK**.
4. В редакторе **Code Analysis Policy Editor** задайте параметр **Enforce C/C++ Code Analysis (/analyze)**, **Enforce Code Analysis For Managed Code** или оба, если ваш проект содержит сочетание управляемого и неуправляемого кода.
5. Если вы выбрали анализ управляемого кода, задайте необходимые правила, руководствуясь собственными стандартами программирования.

Кроме того, вы вольны создать пользовательскую политику возврата после правки для выполнения проверок, не выполняющихся по умолчанию. Можно запретить определенные виды кода, например, вызовы запрещенных функций API, или написать политику для соблюдения стиля программирования, принятого в вашей команде, задав, например, как расставлять фигурные скобки в исходном коде.

Дополнительные ресурсы

- Дополнительную информацию о создании и использовании пользовательской политики возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.

Настраивайте уведомления о перекрытии политик возврата после правки

Система управления версиями Team Foundation Server Version Control не запрещает перекрывать политики возврата после правки. Однако вы можете выполнить следующие действия, чтобы при помощи службы Team Foundation Server Eventing Service из Team Foundation Core Services API выявить факт перекрытия политики: напишите метод **Notify**, проводящий разбор свойств набора изменений и реагирующий на факт перекрытия. Можно также обнаружить перекрытие политики, просмотрев историю набора изменений вручную.

Дополнительная информация

- Дополнительную информацию о перекрытии политики возврата вы найдете в статье «How to: Override a Check-in Policy» по адресу *http://msdn2.microsoft.com/en-us/library/ms245460(VS.80).aspx*.

Шаблоны процесса

- Используйте шаблон процесса MSF Agile в несложных проектах, допускающих неформальный подход.

- Используйте шаблон процесса MSF CMMI в проектах, требующих более формального подхода или соответствия стандартам CMMI.
- По возможности используйте минимальный шаблон процесса.
- Отредактируйте существующий шаблон в соответствии с потребностями вашей команды.

Используйте шаблон процесса MSF Agile в несложных проектах, допускающих неформальный подход

Используя методику разработки через тестирование (Test-Driven Development, TDD) или другие гибкие методики, работайте с шаблоном процесса MSF for Agile Software Development (MSF Agile). Он представляет собой облегченный подход к гибким проектам разработки ПО. Его следует использовать, если вы не слишком нуждаетесь в дополнительных функциях усовершенствования процесса из шаблона MSF CMMI.

Шаблон процесса MSF Agile легко редактировать, изменяя его в соответствии с требованиями вашего процесса.

Дополнительные ресурсы

- Дополнительную информацию об управлении проектами вы найдете в главе 11 этой книги.
- Дополнительную информацию о шаблоне процесса MSF Agile вы найдете в главе 13 этой книги.
- Дополнительную информацию о настройке шаблона процесса вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.

Используйте шаблон процесса MSF CMMI в проектах, требующих более формального подхода или соответствия стандартам CMMI

При использовании более формального подхода к разработке ПО, направленного на усовершенствование существующего процесса, применяйте шаблон процесса MSF для CMMI Software Development. Его также можно изменить в соответствии с требованиями вашего процесса.

Дополнительные ресурсы

- Дополнительную информацию об управлении проектами вы найдете в главе 11 этой книги.
- Дополнительную информацию о настройке шаблона процесса вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.

По возможности используйте минимальный шаблон процесса

Многим командам не требуется поддержка всех разделов стандартного командного проекта. Например, многие команды довольствуются системой управления исходным кодом и не собираются использовать портал Microsoft Office SharePoint®. В подобных случаях шаблоны командных проектов можно изменять, удаляя ненужные разделы. В шаблоне всегда должны присутствовать разделы Group Permissions и Classifications, от остальных же можно смело избавляться.

Чтобы создать минимальный шаблон, при помощи диспетчера Process Template Manager загрузите шаблон на локальный компьютер, отредактируйте его, удалив ненужные разделы, а затем выгрузите шаблон обратно на сервер.

Дополнительные ресурсы

- Дополнительную информацию о шаблонах процесса вы найдете в главе 13 этой книги.
- Дополнительную информацию о настройке шаблона процесса вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о настройке шаблона процесса вы найдете в статье «Customizing Process Templates» по адресу *http://msdn2.microsoft.com/en-us/library/ms243782(VS.80).aspx*.
- Дополнительную информацию об использовании минимального шаблона вы найдете в статье «How to use TFS for source control only» по адресу *http://blogs.msdn.com/richardb/archive/2007/05/10/how-to-use-tfs-for-source-control-only.aspx*.

Отредактируйте существующий шаблон в соответствии с потребностями вашей команды

Проект, над которым вы работаете, может не соответствовать стандартным шаблонам процесса, поставляющимся с Microsoft Visual Studio Team System (VSTS). Вам может понадобиться другой тип рабочего элемента или иная методология процесса. В таком случае вам следует отредактировать существующий шаблон процесса. Выберите шаблон, максимально близкий к вашим требованиям, и измените его в соответствии с этими требованиями.

Как правило, настройке подлежат следующие области шаблона:

- группы и разрешения;
- типы рабочих элементов;
- заметки и политики возврата после правки;
- области и итерации;

- отчеты;
- портал проекта;
- руководство по процессу.

Дополнительные ресурсы

- Дополнительную информацию о шаблонах процесса вы найдете в главе 13 этой книги.
- Дополнительную информацию о настройке шаблона процесса вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.

Группы безопасности и разрешения

- Создавайте группы безопасности для назначения определенного набора разрешений.
- Распределяйте членов команды по соответствующим группам.

Создавайте группы безопасности для назначения определенного набора разрешений

Когда вы создаете новый проект в Team Foundation Server, независимо от выбранного шаблона в нем создается четыре стандартные группы. По умолчанию каждая из этих групп обладает набором определенных разрешений, которые ограничивают полномочия членов этих групп. Четыре стандартные группы таковы:

- Project Administrator — администратор проекта;
- Contributor — участник проекта;
- Reader — читатель;
- Build Services — службы сборки.

Чтобы обеспечить выполнение специфических требований вашей организации, создайте собственные группы безопасности. Это эффективный способ предоставления группе пользователей конкретного набора разрешений. Убедитесь, что вы предоставляете группе лишь необходимый минимум разрешений, добавляйте в нее только тех пользователей или их группы, которым нужны эти разрешения.

Руководствуйтесь следующими соображениями:

- не изменяйте разрешения стандартных групп (или сделайте это и во всех остальных проектах);
- используйте группы AD только на уровне сервера;
- для назначения разрешений используйте группы TFS, а не группы AD;

- никогда и ничего не запрещайте без очень веских причин (запрет чаще всего означает, что используемое вами распределение участников по группам далеко от идеала).

Дополнительные ресурсы

- Дополнительную информацию о создании групп безопасности вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.
- Дополнительные сведения о разрешениях TFS вы найдете в статье «Team Foundation Server Permissions» по адресу *http://msdn2.microsoft.com/en-us/library/ms252587(VS.80).aspx*.

Распределяйте членов команды по соответствующим группам

Определите членов команды, которые будут заняты в проекте, и их роли. Распределите членов команды по группам TFS, группам уровня сервера и группам безопасности. Помните, что в группу безопасности следует добавлять только тех участников, которые действительно нуждаются в разрешениях, предоставляемых данной группой.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.

Командные проекты

- Создавайте один командный проект на приложение, чтобы переносить рабочие элементы и другие ресурсы из версии в версию.
- Создавайте один командный проект на версию, если хотите создавать новые рабочие элементы и прочие ресурсы для каждой версии.
- Не увлекайтесь предоставлением разрешений на доступ к ресурсам проекта.
- Структурируйте дерево исходного кода с учетом ветвления.

Создавайте один командный проект на приложение, чтобы переносить рабочие элементы и другие ресурсы из версии в версию

Если вы намерены переносить из версии в версию не только исходный код, но и рабочие элементы, а также другие объекты TFS, используйте один командный проект на все приложение. Все папки TFS при этом будут автоматически переноситься на следующий выпуск. Когда все будет готово к выпуску новой версии, вы сможете создать для ее изоляции ветвь внутри проекта.

Используя один проект на все приложение, помните о следующих ключевых моментах:

- В параллельных выпусках будут использоваться одни и те же схемы рабочих элементов, политики возврата после правки и руководства по процессам.
- Будет затруднено составление отчетов. Поскольку отчеты составляются для всего проекта, вам придется использовать фильтрацию по выпуску.
- Если количество разрабатываемых приложений исчисляется сотнями, наличие собственного проекта для каждого из них нанесет ущерб производительности TFS. К тому же, вы рискуете выйти за пределы масштабируемости.
- После нескольких выпусков в проекте набирается огромный «багаж». Самый простой способ избавиться от него — создать новый проект с переносом в него только нужного кода.

Дополнительные ресурсы

- Дополнительную информацию об использовании командных проектов вы найдете в статье «When to use Team Projects» по адресу *http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx*.

Создавайте один командный проект на версию, если хотите создавать новые рабочие элементы и прочие ресурсы для каждой версии

Если вы готовы с каждым выпуском начинать все заново, не перенося рабочие элементы и другие папки TFS, создавайте отдельный проект на каждый выпуск. Это позволит вам изменять схемы рабочих элементов, последовательность операций, политики возврата после правки и прочие элементы, не затрагивая при этом предыдущий выпуск. Это особенно полезно, если он будет сопровождаться отдельной командой, например, группой поддержки, порядок работы которой может отличаться от основной команды разработчиков.

Используя новый проект для каждого выпуска, помните о следующих ключевых моментах:

- Переместить код из одного проекта в другой достаточно просто, а вот перемещать рабочие элементы и прочие папки TFS из одного проекта в другой нелегко. Рабочие элементы можно копировать в другой проект только по одному. Если вы хотите копировать наборы рабочих элементов, вам придется написать собственную утилиту.
- Если количество приложений и выпусков исчисляется сотнями и каждый из них находится в собственном проекте, вы столкнетесь с проблемой производительности системы TFS и выйдете за пределы масштабируемости.

- Выбирая структуру, думайте о дне завтрашнем. Реструктуризация существующих командных проектов — трудное занятие.
- Организовать общий доступ к исходному коду из нескольких командных проектов можно несколькими способами:
  - ветвлением исходного кода из одного проекта в другой;
  - сопоставлением исходного кода из другого проекта в вашей рабочей области.
- Система Team Foundation Server способна вместить около 500 проектов на основе шаблона процесса MSF Agile или до 250 проектов на основе шаблона MSF CMMI. Если вы создаете собственный процесс или настраиваете существующий, помните, на масштабируемость сервера наибольшее влияние оказывает схема рабочих элементов. Чем сложнее схема, там меньше проектов сможет поддерживать сервер.
- Вам придется перенести из исходного проекта все области и, возможно, изменить все разрешения в системе управления исходным кодом.

Дополнительные ресурсы

- Дополнительную информацию об использовании командных проектов вы найдете в статье «When to use Team Projects» по адресу *http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx*.

Не увлекайтесь предоставлением разрешений на доступ к ресурсам проекта

Создавая командные проекты, проанализируйте стандартные группы безопасности, созданные процессом, и при необходимости создайте собственные группы с соответствующими разрешениями. Добавьте участников проекта в соответствующие группы, внимательно следя за тем, чтобы каждый из них получал разрешения на доступ только к тем ресурсам, которые ему нужны.

Дополнительные ресурсы

- Дополнительную информацию о предоставлении разрешений вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.

Структурируйте дерево исходного кода с учетом ветвления

Создавая структуру дерева исходного кода, убедитесь, что она поддерживает ветвление. Создайте отдельные папки для кода и для других ресурсов проекта. Если в будущем потребуется изолированная разработка, вы с легкостью выполните ветвление папки исходного кода. Убедитесь в наличии отдельных папок для каждого компонента внутри папки исходного кода, что позволит при необходимости выполнять частичное ветвление.

Разделяйте с помощью папок прочие категории объектов, например, модульные тесты, зависимости библиотек и т.п. Это позволит при ветвлении включать или исключать их по мере надобности.

Ниже приведен пример дерева исходного кода с поддержкой ветвления:
Main — контейнер, содержащий все объекты, необходимые для отправки проекта заказчику.

- **Source** — контейнер, содержащий все объекты, необходимые для выполнения сборки.
  - **Code** — контейнер для исходного кода.
  - **Shared Code** — контейнер для исходного кода, используемого совместно с другими проектами.
  - **Unit Tests** — контейнер для модульных тестов.
  - **Lib** — контейнер для двоичных зависимостей.
- **Docs** — контейнер для документации, поставляющейся с проектом.
- **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
- **Builds** — контейнер для сценариев Team Build.
- **Tests** — контейнер с результатами тестов, проводимых тестовой командой.

Дополнительные ресурсы

- Дополнительную информацию о структуре дерева исходного кода вы найдете в главе 5.

Рабочие элементы

- Создавайте сценарии в начале работы над проектом.
- Правильно определяйте требования QoS.
- Разделяйте сценарии на управляемые задачи.
- Разрабатывайте критерии приемки для каждой задачи.
- Связывайте требования и задачи со сценариями.
- Используйте Microsoft Excel для массового редактирования рабочих элементов.

Создавайте сценарии в начале работы над проектом

Создавайте и записывайте сценарии проекта в начале работы над ним. Это позволит вам составить полную картину проекта и в дальнейшем поможет отслеживать продвижение к намеченной цели. В ходе разработки вы вольны модифицировать существующие сценарии или добавлять новые согласно накопленной информации.

Создание сценариев в начале проекта

1. Изучите журнал заявок проекта (project back log, PBL), который содержит требования, выдвинутые различными заинтересованными сторонами (заказчиками, бизнес-аналитиками, конечными пользователями и руководителями производства) и определите область действия сценариев вашего проекта.
2. В Team Explorer разверните узел проекта, щелкните правой кнопкой папку Work Items, раскройте подменю **Add Work Item** и выберите команду **Scenario**.
3. На странице **New Scenario** введите описание сценария. Убедитесь, что для параметра **Iteration** задано значение **999**.
4. Сохраните новый сценарий.
5. Повторите шаги для всех сценариев, заданных для проекта.

Дополнительные ресурсы

- Дополнительную информацию о предоставлении разрешений вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.

Правильно определяйте требования QoS

Определите требования QoS для каждого сценария, над которым будет вестись работа в цикле итерации. Это поможет определить критерий приемки сценария. Требования к QoS основываются на целях и требованиях проекта, а также на документах спецификации, если таковые имеются.

Определение требований QoS

1. Правой кнопкой мыши щелкните папку **Work Items** вашего проекта, раскройте подменю **Add Work Item** и выберите команду **Quality of Service Requirements**.
2. На New Quality of Service Requirements введите следующие сведения:
   a. Задайте значение параметра **Type** — производительность, масштабируемость, нагрузка или безопасность.
   б. В параметре **Iteration** укажите текущий цикл итерации.
   в. На вкладке **Links** свяжите QoS с отдельным сценарием, чтобы облегчить отслеживание.
3. Сохраните новое требование QoS.
4. Создайте по одному требованию QoS для каждого уровня или типа требования. Помните, что у каждого сценария может быть несколько требований QoS.

5. Убедитесь, что создали требования QoS для всех сценариев, вызывающихся во время отдельного цикла итерации.

> **Важно!** Позже вы сможете разбить требования QoS на тестовые задачи.

Дополнительные ресурсы

- Дополнительную информацию о предоставлении разрешений вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о рабочих элементах вы найдете в статье «Managing Team Foundation Work Items» по адресу *http://msdn2.microsoft.com/en-us/library/ms181314(VS.80).aspx*.

Разделяйте сценарии на управляемые задачи

Планируя итерации, разделяйте сценарии на блоки, а блоки — на задачи. Убедитесь, что создаваемые задачи ограничены и управляемы. Задача не должна длиться более одного-двух дней. Если продолжительность задачи превышает два дня, ее следует разбить на меньшие подзадачи. Это положительно сказывается на гибкости расписания и управляемости проекта.

Разделение сценариев на управляемые задачи

1. Разделите выбранные сценарии на блоки.
2. Разделите блоки на задачи.
3. Введите задачи разработчиков в TFS в качестве рабочих элементов, выполнив следующие действия:
   а. В окне Team Explorer щелкните правой кнопкой папку **Work Items** в узле вашего проекта, раскройте подменю **Add Work Item** и выберите команду **Task**.
   б. На странице **New Task** укажите следующие сведения:
      - Параметру **Discipline** присвойте значение **Development**.
      - Параметру **Iteration** присвойте номер текущего цикла итерации.
      - На вкладке **Links** свяжите задачу с конкретным сценарием для облегчения отслеживания. Здесь же можно ввести критерий завершения задачи.
      - В поле **Assigned to** укажите разработчика, работающего над задачей.
   в. Сохраните задачу.
   г. Повторите эти шаги для всех вновь определенных задач.
4. Повторите описанные выше шаги для всех определенных сценариев данной итерации.

Дополнительные ресурсы

- Дополнительную информацию о предоставлении разрешений вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о рабочих элементах вы найдете в статье «Managing Team Foundation Work Items» по адресу *http://msdn2.microsoft.com/en-us/library/ms181314(VS.80).aspx*.

Разрабатывайте критерии приемки для каждой задачи

Определив задачи разработки, включите в них критерии приемки, позволяющие разработчику принять решение о завершении задачи. В зависимости от используемого шаблона процесса критерий можно реализовать двумя различными способами:

- **MSF Agile** При использовании MSF Agile без формального требования типа рабочего элемента, проще всего включить критерий приемки в сам рабочий элемент в виде текста. Создайте маркированный список и по мере надобности добавляйте в него новые сведения.
- **MSF CMMI** Этот шаблон позволяет задействовать для определения критериев приемки задачи формальные требования. Первым шагом является определение требований. Далее создается задача для их реализации. Между задачей и требованиями устанавливается связь, которая повышает возможности отслеживания и позволяет разработчику проверять результаты работы на соответствие требованиям.

Критерий приемки чаще всего определяется как требование к интерфейсу в виде мини-сценария или требования QoS. После успешного прохождения приемки разработчик помечает задачу как завершенную и переходит к следующей.

Дополнительные ресурсы

- Дополнительную информацию о рабочих элементах вы найдете в статье «Managing Team Foundation Work Items» по адресу *http://msdn2.microsoft.com/en-us/library/ms181314(VS.80).aspx*.

Связывайте требования и задачи со сценариями

Создавая новые рабочие элементы (задачи, ошибки, проблемы или требования QoS), не забывайте связывать их со сценариями, которые привели к их появлению. Это гарантирует, что в основе каждого рабочего элемента лежит конкретный сценарий, и помогает контролировать работу над сценарием в ходе итераций.

Связывание задач, ошибок, проблем и требований QoS со сценариями

1. На странице **New work item** перейдите на вкладку **Links** и щелкните кнопку **Add**.

2. В диалоговом окне **Add Link** в разделе **Link Type** выберите вариант **Scenario**.
3. Щелкните кнопку **Browse**, чтобы найти сценарии в командном проекте.
4. Выберите сценарий, на который хотите создать ссылку, и щелкните **OK**.
5. В поле **Comment** введите комментарий, поясняющий связь сценария с рабочим элементом. Поле **Description** заполняется автоматически.
6. Щелкните **OK**.

Дополнительные ресурсы

- Дополнительную информацию о предоставлении разрешений вы найдете в разделе «Как управлять проектами в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о рабочих элементах вы найдете в статье «Managing Team Foundation Work Items» по адресу *http://msdn2.microsoft.com/en-us/library/ms181314(VS.80).aspx*.

Используйте Microsoft Excel для массового редактирования рабочих элементов

Система Team Foundation Server не поддерживает массового редактирования рабочих элементов. Вам приходится редактировать каждый элемент индивидуально. Если вам все-таки нужно отредактировать много рабочих элементов за короткое время, например, за время совещания, вам поможет Microsoft Office Excel®. Экспортируйте рабочие элементы из TFS в Excel, модифицируйте, а затем снова импортируйте в TFS для сохранения правок.

Создание списка рабочих элементов в Excel с последующим редактированием

1. Запустите Microsoft Office Excel и выберите в меню **Team** команду **New List**.
2. В поле **Connect to a Team Foundation Server** укажите сервер, к которому следует выполнить подключение, или щелкните **Servers** и введите информацию о сервере.
3. В разделе **Team Projects** выберите на сервере Team Foundation Server командный проект, с которым хотите работать. Документ будет связан с этим командным проектом.
4. Щелкните **OK**.
5. Выберите тип списка. Чтобы создать список запросов, выберите вариант **Query List**, а затем укажите запрос в раскрывающемся списке **Select a Query**.
6. Выберите столбцы, которые должны присутствовать в новом списке рабочих элементов.

7. Импортируйте нужные рабочие элементы. Дополнительную информацию вы найдете в статье «How to: Import Work Items in Microsoft Excel or Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms181676(VS.80).aspx*.
8. Редактируйте рабочие элементы и публикуйте их обновленные версии в БД рабочих элементов, выбрав команду **Publish** в меню **Team**.

Дополнительные ресурсы

- Дополнительную информацию об использовании Microsoft Office Excel для управления проектом вы найдете в статье «Working with Work Item Lists in Microsoft Excel» по адресу *http://msdn2.microsoft.com/en-us/library/ms181694(VS.80).aspx*.

Дополнительные ресурсы по управлению проектами

- Дополнительную информацию о шаблонах процесса MSF вы найдете в статье «Process Templates» по адресу *http://msdn2.microsoft.com/en-us/teamsystem/aa718801.aspx*.

Руководство по отчетам

В этом разделе

Администрирование

- Убедитесь, что пользователи включены в правильные группы безопасности.
- Создайте единую панель отчетов для просмотра состояния проекта и его показателей.

Создание и настройка

- Убедитесь в правильности имени сервера при развертывании отчетов.
- Создавайте снимки отчетов по расписанию.
- Изменяйте существующие отчеты, чтобы получить доступ к дополнительным данным.

Просмотр

- Чтобы получить актуальные данные, убедитесь в работе веб-службы хранилища.

Администрирование

- Убедитесь, что пользователи включены в правильные группы безопасности.
- Создайте единую панель отчетов для просмотра состояния проекта и его показателей.

Убедитесь, что пользователи включены в правильные группы безопасности

Если вы хотите, чтобы у пользователя была возможность разворачивать отчеты, убедитесь, что он включен в роль Content Manager сервера отчетов. Эта роль предопределена в Report Services и предназначена для пользователей, осуществляющих развертывание и управление отчетами и подключе-

ниями к источникам данных на веб-сервере. Чтобы пользователь мог разворачивать отчеты на сервере отчетов системы Microsoft® Visual Studio® 2005 Team Foundation Server (TFS), он должен быть членом этой роли.

Добавление пользователя в роль Content Manager

1. Откройте сайт отчетов проекта. В Team Explorer щелкните правой кнопкой папку Reports вашего командного проекта и выберите **Show Report Site**.
2. Перейдите на вкладку **Properties**.
3. В левой части окна щелкните **Security**.
4. Щелкните **New Role Assignment**.
5. В поле **Group or user name** введите имя пользователя или группы, которую хотите добавить в роль Content Manager.
6. Установите флажок **Content Manager**.
7. Щелкните **OK**.

Дополнительные ресурсы

- Дополнительную информацию о роли Content Manager вы найдете в статье «Content Manager Role» по адресу *http://technet.microsoft.com/en-us/library/ms159693(SQL.90).aspx*.
- Дополнительную информацию о ролях безопасности на уровне данных вы найдете в статье «Securing Access Through Analysis Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms174839.aspx*.
- Дополнительную информацию о ролях безопасности на уровне приложений вы найдете в статье «Securing Reporting Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms157198.aspx*.

Создайте единую панель отчетов для просмотра в состояния проекта и его показателей

Панель отчетов позволит вам и вашей команде получить на одной странице доступ ко всей важной информации о проекте. Стандартная страница портала Microsoft Office SharePoint® шаблона Microsoft Solution Framework (MSF) для проектов MSF Agile содержит один отчет и ссылки на остальные. Чтобы создать единое хранилище для информации по проекту, измените страницу портала проектов MS Agile или MSF CMMI, чтобы она включала столько отчетов, сколько вам нужно.

Функциональная панель отчетов должна, вероятно, включать следующие отчеты:

- об оставшейся работе;
- о показателях качества;

- о частоте появления ошибок;
- о темпе продвижения проекта.

Вы вольны добавлять новые отчеты на страницу портала SharePoint. Для этого в каждый отчет, который вы хотите отобразить на странице, нужно добавить компонент Report Viewer Web Part.

Изменение портала командного проекта и создание панели отчетов

1. Установите компонент Report Viewer Web Part на сервер отчетов. Для этого используются инструмент stsadm.exe и файл RSWebParts.cab, которые входят в дистрибутив Microsoft Office SharePoint и Report Services, например:

   ```
   STSADM.EXE -o addwppack -filename "C:\ Program Files\Microsoft SQL
   Server\90\Tools\Reporting Services\SharePoint\RSWebParts.cab" -
   globalinstall
   ```

   - Инструмент STSADM.EXE находится в папке C:\Program Files\Common Files\Microsoft Shared\web server extensions\60\BIN.
   - Файл RSWebParts.Cab находится в папке C:\ Program Files\Microsoft SQL Server\90\Tools\Reporting Services\SharePoint.
2. В Team Explorer щелкните правой кнопкой ваш проект.
3. Выберите команду **Show Project Portal**.
4. Щелкните **Modify Shared Page**.
5. Наведите указатель на **Browse** и щелкните **Add Web Parts**.
6. Щелкните **Virtual Server Gallery**.
7. В списке **Web Part List** выберите вариант **Report Viewer**.
8. Щелкните кнопку **Add**.
9. Введите имя диспетчера отчетов, например, http://<*сервер отчетов*>/reports.
10. Введите путь отчета, который хотите отобразить, например: <*мой проект*>/Quality Indicators.

Дополнительные ресурсы

- Дополнительную информацию о добавлении компонента Report Viewer Web Part вы найдете в статье «Viewing Reports with SharePoint 2.0 Web Parts» по адресу *http://msdn2.microsoft.com/en-us/library/ms159772(SQL.90).aspx*.
- Дополнительную информацию о портале командного проекта вы найдете в статье «Using the Team Project Portal» по адресу *http://msdn2.microsoft.com/en-us/library/ms242883(VS.80).aspx*.

Создание и настройка

- Убедитесь в правильности имени сервера при развертывании отчетов.
- Создавайте снимки отчетов по расписанию.
- Изменяйте существующие отчеты, чтобы получить доступ к дополнительным данным.

Убедитесь в правильности имени сервера при развертывании отчетов

Если URL сервера отчетов или имя целевой папки указаны неверно, отчет не будет развернут на сервере отчетов. Выполняя развертывание отчета из Visual Studio 2005, укажите URL сервера, на который следует выполнить развертывание, и имя командного проекта, частью которого является данный отчет. Адрес URL сервера отчетов, на котором будет произведено развертывание, выглядит так: http://TeamServerName/**СерверОтчетов**, где **СерверОтчетов** — конечная точка веб-службы Report Server.

Укажите имя командного проекта в поле TargetReportFolder диалогового окна свойств развертывания с учетом регистра. При ошибке в регистре отчет будет развернут, однако не будет отображаться в списке отчетов командного проекта в Team Explorer.

Дополнительные ресурсы

- Дополнительную информацию о настройке развертывания вы найдете в статье «How to: Set Deployment Properties (Report Designer)» по адресу *http://technet.microsoft.com/en-us/library/ms155802(SQL.90).aspx*.

Создавайте снимки отчетов по расписанию

Чтобы создавать снимки данных проекта через регулярные промежутки времени, используйте историю отчета. Эти снимки можно просматривать через какое-то время, чтобы увидеть тенденции развития проекта. Кроме того, они позволяют сохранить важные информационные точки проекта.

Создание запланированного снимка отчета

1. Откройте отчет из портала отчетов.
2. Перейдите на вкладку **Properties**.
3. Перейдите по ссылке **History**.
4. Установите расписание для запуска снимка.

Создав расписание, вы сможете просматривать отчеты на вкладке **History** этого отчета. Там же можно создать снимок вручную.

Изменяйте существующие отчеты, чтобы получить доступ к дополнительным данным

Редактирование отчетов осуществляется при помощи инструмента Microsoft SQL Server™ 2005 Reporting Services Designer, входящего в Visual Studio (Business Intelligence Development Studio) из клиентского комплекта SQL Server 2005.

Настройка отчета позволяет расширить его функциональность, не создавая при этом нового отчета. Если нужный отчет похож на уже существующий, выполните настройку существующего отчета — это сэкономит ваше время. Для настройки существующего отчета вам придется экспортировать его с сервера отчетов, добавить в существующий проект отчета Visual Studio, а затем после внесения изменений выполнить повторное развертывание на портале отчетов.

> **Примечание** Вы, конечно, можете использовать построитель отчетов (Report Builder), который имеется на сайте отчетов команды, но этот инструмент не очень хорошо поддерживается сценариями отчетов Visual Studio, поэтому работать с ним не рекомендуется.

Дополнительные ресурсы

- Более подробную информацию вы найдете в разделе «Как настроить отчет в Visual Studio 2005 Team Foundation Server» этой книги.
- Дополнительную информацию об отчетах вы найдете в главе 15 этой книги.
- Учебные курсы по работе с проектами отчетов вы найдете в статье «Reporting Services Tutorials» по адресу *http://msdn2.microsoft.com/en-us/library/ms170246.aspx*.
- Подробнее о редактировании отчетов читайте в статье «How to: Edit Reports in Report Designer» по адресу *http://msdn2.microsoft.com/en-us/library/ms244655(VS.80).aspx*.

Просмотр

- Чтобы получить актуальные данные, убедитесь в работе веб-службы хранилища.

Чтобы получить актуальные данные, убедитесь в работе веб-службы хранилища

По умолчанию веб-служба хранилища запускается для генерирования данных отчета ежечасно. Готовя отчет, запускайте службу вручную, чтобы в отчете с гарантией содержались актуальные данные.

Ручной запуск службы хранилища

1. Откройте Internet Information Services (IIS) Manager.
2. Выберите веб-сайт **Team Foundation Server**.
3. Внутри веб-сайта откройте папку **Warehouse\v1.0**. Отобразится страница со списком действий, доступных по отношению к хранилищу.
4. Щелкните правой кнопкой файл **warehousecontroller.asmx** и выберите команду **Browse**.
5. Щелкните **Run**, затем **Invoke**. Откроется второе окно обозревателя, отображающее состояние запроса выполнения. В нем должно быть отображено значение **true**.
6. Вернитесь в предыдущее окно обозревателя и снова перейдите на страницу действий.
7. Выберите **GetwareHouseStatus** и щелкните **Invoke**.

   Отобразится текущее состояние веб-службы хранилища. Значение **idle** указывает, что запуск службы был выполнен.

Дополнительные ресурсы

- Дополнительную информацию о поиске неисправностей хранилища вы найдете в статье «Troubleshooting the Data Warehouse» по адресу *http://msdn2.microsoft.com/en-us/library/ms244674(vs.80).aspx*.

Дополнительные ресурсы по отчетам Team Foundation

- Дополнительную информацию об отчетах вы найдете в статье «Team Foundation Server Reporting» по адресу *http://msdn2.microsoft.com/en-us/library/ms194922(VS.80).aspx*.

Руководство по управлению исходным кодом

В этом разделе

Доступ к системе управления версиями

- Инструменты командной строки.
- Team Foundation Power Tools: восстановление отложенных изменений.
- Team Foundation Power Tools: откат изменения.
- Team Foundation Power Tools: автономная работа.
- Team Foundation Power Tools: получение набора изменений.
- Team Foundation Power Tools: удаление невозвращенных правок.

Администрирование

- Отключайте наследуемые разрешения в ветвях сопровождения.
- Запрещайте возврат после правки разработчикам, которым пока не доверяете.

Ветвление, метки, слияние

- Используйте метки для выделения сборок, к которым можете вернуться впоследствии.
- Используйте ветвление для изоляции выпусков.
- Планируйте структуру ветвей в соответствии с путями слияний.
- Выполняйте ветвление на высоком уровне, включая файлы конфигурации и исходного кода.
- Избегайте глубокого ветвления.
- Избегайте ветвления без необходимости.
- Избегайте слияния без основы.
- Предпочитайте полное слияние выборочному.
- Чаще выполняйте слияния.

- Всегда создавайте папку верхнего уровня для нового командного проекта.
- Используйте перед слиянием параметры Candidate или Preview.
- Если в составе слияния выполняется переименование, обращайте внимание на предлагаемый системой путь.
- Соблюдайте осторожность при разрешении конфликтов слияния.
- Не возвращайте результаты двух и более слияний одновременно.
- Проводите тесты после слияния, но перед возвратом.

Возврат после правки и политики возврата

- Возвращайте код, только убедившись в его надежности.
- Используйте наборы отложенных правок для архивации или совместного использования незавершенных изменений.
- Разрешайте только один рабочий элемент за один возврат.
- Применяйте политики возврата для соблюдения стандартов программирования.
- Применяйте политики возврата для проверки качества кода.
- Следите за перекрытием политики.
- Избегайте конфликтов при помощи планирования.

Получение и блокировка кода

- Перед внесением изменений извлеките последнюю версию исходного кода.
- Осмотрительно пользуйтесь командой lock.
- Предупредите коллег о блокировке файла.

Зависимости

- Старайтесь использовать ссылки на проект.
- Без необходимости не используйте файловые ссылки.
- Для ссылок на проект и файл используйте параметр copy local = true.
- В ссылках на веб-службы используйте динамические URL.

Распределенная и удаленная разработка

- Убедитесь, что объема диска достаточно для реализации прокси.
- Создайте задание для получения последних файлов по расписанию.
- Регулярно проверяйте счетчики производительности прокси и журнал событий.
- Настраивайте executionTimeout соответственно размеру файла и ширине канала.

- Отключайте прокси, если он не будет работать продолжительное время.
- Скрывайте рабочие области, чтобы избежать ненужного перемещения файлов.

Переход из других версий

- Переходите на Team Foundation Server Source Control при помощи VSS Converter.
- Переходите на Team Foundation Server Source Control из других систем управления исходным кодом.

Управление проектами и рабочими областями

- Для изоляции одного разработчика используйте рабочие области, а не ветвление.
- Удаляйте и переименовывайте файлы в системе управления исходным кодом, а не в проводнике Windows.
- Удаляйте и переименовывайте файлы только при открытом решении.
- Создавайте один командный проект на приложение, если хотите переносить ресурсы от версии к версии.
- Создавайте командный проект для каждой версии, если хотите с каждой новой версией начинать все заново.
- Используйте ветвление для обеспечения совместного доступа к исходному коду и двоичным файлам.
- Не используйте сопоставление для поддержки зависимостей между различными проектами.
- Сопоставляйте рабочие области на корневом уровне командного проекта.
- Используйте уникальный путь к локальной папке на общих компьютерах.
- Сопоставляйте только части дерева исходного кода.
- Структурируйте дерево исходного кода с учетом ветвления.

Отложенные правки

- Используйте отложенные правки для предоставления общего доступа к незавершенным изменениям.
- Используйте отложенные правки для архивации незавершенных изменений на сервере.
- Используйте отложенные правки, если вас прервали для выполнения более важного задания.

Доступ к системе управления версиями

- Инструменты командной строки.
- Team Foundation Power Tools: восстановление отложенных изменений.
- Team Foundation Power Tools: откат изменения.
- Team Foundation Power Tools: автономная работа.
- Team Foundation Power Tools: получение набора изменений.
- Team Foundation Power Tools: удаление невозвращенных правок.

Инструменты командной строки

Для операций, не предусмотренных пользовательским интерфейсом Visual Studio, а также для задач, выполняемых по расписанию, удобны инструменты командной строки, например, Team Foundation Power Tools (Tfpt.exe), которые включены в комплект Team Foundation Server (TFS), но загружаются отдельно. С их помощью вы можете настраивать выполнение задач по расписанию, используя планировщик заданий Windows (Task Scheduler).

Чтобы с гарантией задать нужные пути и другие переменные окружения, запускайте Tf.exe из окна командной строки Visual Studio или выполните пакетный файл Vsvars32, который обычно располагается в папке *Диск*:\Program Files\Microsoft Visual Studio 8\Common7\Tools. Инструмент Tf.exe поддерживает большинство команд системы управления исходным кодом, включая **Checkin, Checkout, Get, History, Shelve, Branch, Merge, Label, Status, Undelete** и **Undo**.

Ниже указаны наиболее распространенные операции, выполняемые из командной строки с помощью Tf.exe:

- Синхронизация файлов локального компьютера с файлами сервера: **tf get**.
- Добавление файлов на сервер: **tf add**.
- Извлечение файла для редактирования: **tf checkout**.
- Возврат правки: **tf checkin**.
- Извлечение определенного набора изменений с сервера: **tf get /version**.

Некоторые операции можно выполнить только из командной строки:

- Удаление рабочей области другого пользователя: **tf workspace /delete**.
- Удаление правок другого пользователя: **tf undo**.
- Снятие блокировки, установленной другим пользователем: **tf lock**.
- Определение области действия меток: **tf label**.
- Слияние без основы: **tf merge**.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Walkthrough: Working with Team Foundation Source Control from Command Line» на сайте

Microsoft MSDN® по адресу *http://msdn2.microsoft.com/en-us/library/zthc5x3f.aspx*.

- Подробнее о командах, доступных только из командной строки, читайте в статье «Operations Available Only From the Command-Line (Team Foundation Source Control)» по адресу *http://msdn2.microsoft.com/en-us/library/ms194957(VS.80).aspx*.

Team Foundation Power Tools: восстановление отложенных изменений

Инструментарий Team Foundation Power Tools (TFPT) обладает функциональными возможностями, недоступными в Visual Studio. Например, он поддерживает работу в автономном режиме, позволяет отменять возвращенные правки из набора изменений, а также восстановить отложенные изменения.

Операция возврата отложенных изменений (**unshelve**), поддерживаемая TFS, не допускает слияния *отложенных изменений* (shelved change) и *локальных изменений* (local change). Если в элемент локальной рабочей области внесено незафиксированное изменение-правка и кроме того с ним связано отложенное изменение-правка, TFPT позволяет выполнить трехстороннее слияние изменений.

Эта команда запускается из командной строки при помощи Tfpt.exe.

Дополнительные ресурсы

- Загрузить Team Foundation Power Tools можно по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Форум Team Foundation Power Tools расположен по адресу *http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1*.

Team Foundation Power Tools: откат изменения

Вообще, система TFS напрямую не позволяет отменить возврат набора изменений, но вы можете попытаться отменить любые изменения, сделанные в конкретном наборе, при помощи команды **rollback**. Отменить удастся не все изменения, но в большинстве сценариев команда rollback работает.

Эта команда запускается из командной строки при помощи Tfpt.exe.

Дополнительные ресурсы

- Загрузить Team Foundation Power Tools можно по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Форум Team Foundation Power Tools расположен по адресу *http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1*.

Team Foundation Power Tools: автономная работа

В целом, автономный режим в TFS не поддерживается. Чтобы работать автономно, выполните описанные ниже действия в строго заданной последовательности:

1. Вручную снимите флаги «только для чтения».
2. Отредактируйте файлы.
3. Добавьте или удалите файлы.
4. Запустите команду TFPT online.
   Далее приводится подробное описание каждого из этих шагов.

Важно! Во время автономной работы нельзя переименовывать файлы.

---

1. Вручную снимите флаги «только для чтения».

   По умолчанию, все файлы, извлеченные для правки, доступны только для чтения. При отсутствии подключения к серверу вы должны вручную снять флажки «только для чтения» с файлов, прежде чем редактировать или удалять их. Щелкните файл правой кнопкой в окне проводника Windows, выберите команду **Свойства (Properties)**, сбросьте флажок **Только чтение (Read-only)** и щелкните **OK**. То же действие можно выполнить с помощью команды DOS **attrib -r**.

2. Отредактируйте файлы.

   Теперь вы можете редактировать любые файлы, с которых сняли метку «только для чтения».

3. Добавьте или удалите файлы.

   Вы можете добавлять или удалять файлы, с которых сняли метку «только для чтения». Не переименовывайте файлы, поскольку инструмент TFTP **online** не в состоянии отличить переименование от удаления старого файла и добавления нового.

Примечание Выявление удаленных файлов — процедура, требующая довольно много времени, поэтому на необходимость ее выполнения в команде TFPT **online** указывает специальный параметр, который нужно задавать вручную.

---

4. Запустите команду TFPT online.

   Вернувшись в оперативный режим, введите в командной строке **TFTP online**. Эта команда проверит рабочую область на предмет наличия записываемых файлов и установит, какие изменения следует отправить на сервер. Если вы удалили какие-либо файлы, задайте в команде параметр **/delete**, чтобы команда зафиксировала их. Затем инструмент отобразит

окно, в котором можно выбрать изменения для переноса в рабочую область.

Дополнительные ресурсы

- Загрузить Team Foundation Power Tools можно по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Форум Team Foundation Power Tools расположен по адресу *http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1*.

Team Foundation Power Tools: получение набора изменений

Команда TFPT **GetCS** позволяет получить список всех элементов, содержащихся в наборе изменений на момент его создания. Это полезно, например, если вы хотите получить изменение, возвращенное вашим коллегой, но не можете обновить до последней версии всю рабочую область.

Эта команда запускается из командной строки при помощи Tfpt.exe.

Дополнительные ресурсы

- Загрузить Team Foundation Power Tools можно по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Форум Team Foundation Power Tools расположен по адресу *http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1*.

Team Foundation Power Tools: удаление невозвращенных правок

Используйте инструмент TFPT, чтобы удалить из файлов невозвращенные правки. Команда TFPT Undo Unchanged удаляет невозвращенные правки из файлов, редактирование которых не было выполнено. Это полезно, когда вы извлекаете для редактирования большое количество файлов, но реально изменяете лишь некоторые из них. Вы можете отменить правки в неизменных файлах, запустив инструмент TFPT UU. Он сравнивает хеши файлов в локальной рабочей области с хешами на сервере, что позволяет установить, был ли файл на самом деле изменен.

Эта команда запускается из командной строки при помощи Tfpt.exe.

Дополнительные ресурсы

- Загрузить Team Foundation Power Tools можно по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.

- Форум Team Foundation Power Tools расположен по адресу *http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1*.

Администрирование

- Отключайте наследуемые разрешения в ветвях сопровождения.
- Запрещайте возврат после правки разработчикам, которым пока не доверяете.

Отключайте наследуемые разрешения в ветвях сопровождения

Если ветвь предназначена для сопровождения, например, после выхода в свет очередной версии ПО, отключите наследование разрешений, чтобы изолировать дерево. После этого можно предоставить отдельным пользователям разрешения PendChange и Checkin для внесения исправлений.

Дополнительные ресурсы

Подробнее об удалении разрешений читайте в статье «How to: Remove Access to Source Control Files» по адресу *http://msdn2.microsoft.com/en-us/library/ms400718(VS.80).aspx*.

Запрещайте возврат после правки разработчикам, которым пока не доверяете

Вы можете запретить возврат после правки в дерево исходного кода разработчикам, которым еще не достаточно доверяете, например, новичкам или стажерам. Прежде чем отключить наследование, убедитесь, что задали все нужные разрешения (включая разрешения для собственной учетной записи). Эти разработчики не смогут возвращать изменения напрямую, но им можно будет делать отложенные изменения и откладывать их при помощи команды shelve. Более опытный разработчик затем просмотрит эти изменения и при необходимости передаст в исходный код.

Дополнительные ресурсы

Подробнее об удалении разрешений читайте в статье «How to: Remove Access to Source Control Files» по адресу *http://msdn2.microsoft.com/en-us/library/ms400718(VS.80).aspx*.

Ветвление, метки, слияние

- Используйте метки для выделения сборок, к которым можете вернуться впоследствии.
- Используйте ветвление для изоляции выпусков.

- Планируйте структуру ветвей в соответствии с путями слияний.
- Выполняйте ветвление на высоком уровне, включая файлы конфигурации и исходного кода.
- Избегайте глубокого ветвления.
- Избегайте ветвления без необходимости.
- Избегайте слияния без основы.
- Предпочитайте полное слияние выборочному.
- Чаще выполняйте слияния.
- Всегда создавайте папку верхнего уровня для нового командного проекта.
- Используйте перед слиянием параметры Candidate или Preview.
- Если в составе слияния выполняется переименование, обращайте внимание на предлагаемый системой путь.
- Соблюдайте осторожность при разрешении конфликтов слияния.
- Не возвращайте результаты двух и более слияний одновременно.
- Проводите тесты после слияния, но перед возвратом.

Используйте метки для выделения сборок, к которым можете вернуться впоследствии

Присваивайте метки повседневным сборкам, чтобы можно было быстро просмотреть и извлечь набор файлов, использованных в той или иной сборке. Система TeamBuild автоматически присваивает метку набору файлов, связанному с каждой создаваемой ей сборкой. Формат меток TeamBuild таков: «ТипВыпуска_НомерСборки», например, «Releasex86_20070226.1».

Хотя каждая сборка помечается автоматически, вам, возможно, потребуется создать собственные метки для сборок, к которым вы можете вернуться в дальнейшем, в частности:

- для внутренних этапных сборок, например, для альфа-версии;
- для сборки, созданной после интеграции ветви или внешней зависимости.

Чтобы облегчить поиск сборки в будущем и без проблем разобраться, что она собой представляет, соблюдайте соглашение об именах. Метки должны четко отражать содержание. Например, метка «AlphaBuild_200701122 создана на основе правила «ИмяСборки_Дата».

Выпуская сборку, которую вам придется сопровождать, используйте вместо меток ветвление, чтобы изолировать последующие работы по сопровождению. Позже вы сможете выполнить слияние этой ветви и всех ее исправлений с главным деревом исходного кода.

Чтобы найти существующую метку, откройте меню **File**, разверните подменю **Source Control, Label** и щелкните **Find Label**. Найдя метку, вы можете изменить или удалить ее в диалоговом окне **Find Label**.

Дополнительные ресурсы

- Информацию о метках вы найдете в статье «Working with labels» по адресу *http://msdn2.microsoft.com/en-us/library/ms181439(VS.80).aspx*.
- Подробнее о метках читайте в статье «How to: Apply Labels» по адресу *http://msdn2.microsoft.com/en-us/library/ms181440(VS.80).aspx*.

Используйте ветвление для изоляции выпусков

Создайте ветвь с исходным кодом, использованным при создании выпуска, поддержка которого осуществляется в настоящее время. Это позволит применять исправления и обновления к этой версии ПО, не влияя на продолжающуюся разработку базового исходного кода в главной ветви. При этом возможно выполнение стабилизационных слияний из ветви сопровождения в главную ветвь до выхода очередного выпуска продукта.

Далее приведен пример структуры ветвей после создания ветви Maintenance для поддержки вышедших версий ПО:

- **Main** — ветвь интеграции
  - **Source**
  - **Другие папки ресурсов**
- **Releases** — контейнер для ветвей сопровождения
  - **Release 1** — ветвь сопровождения
    - **Source**

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Планируйте структуру ветвей в соответствии с путями слияний

Слияние в обозревателе Source Control возможно только по существующим путям ветвления. Из командной строки вы можете выполнять слияние без ос-

новы по другим путям, однако такой тип слияния менее гибок, что приводит к большему количеству конфликтов в текущем слиянии и будущих слияниях.

Выполняя слияния, имейте в виду следующее. Слияние вдоль иерархии, от дочерней папки к родительской и наоборот, приводит к меньшему количеству конфликтов, чем слияние поперек иерархии. Кроме того, иерархия ветвей, основанная на родительских и дочерних ветвях, может отличаться от физической структуры исходного кода на диске. Например:

- **Физическая структура**
  - **Development** — ветвь разработки
  - **Main** — ветвь интеграции
  - **Releases** — контейнер для ветвей выпусков
    - **Release 1** — ветвь выпуска
- **Логическая структура**
  - **Main**
    - **Development**
    - **Release 1**

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Выполняйте ветвление на высоком уровне, включая файлы конфигурации и исходного кода

Выполняйте ветвление на уровне, положение которого обеспечит компилируемость создаваемой ветви. Пример ветвления в дереве исходного кода:

- **Main** — контейнер для всех ресурсов, требующихся при создании продукта.
- **Source** — контейнер, содержащий все, что нужно для сборки.
  - **Code** — контейнер для исходного кода.
  - **Shared Code** — контейнер для исходного кода, общего с другими проектами.

  - **Unit Tests** — контейнер для модульных тестов.
  - **Lib** — контейнер для зависимостей двоичных файлов.
- **Docs** — контейнер для документации, поставляемой с продуктом.
- **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
- **Tests** — контейнер для результатов тестирования.

Ветвление следует выполнять на уровне папки Source. Благодаря этому новая ветвь будет содержать все файлы исходного кода и конфигурации.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Избегайте глубокого ветвления

Не увлекайтесь глубоким ветвлением. Это приводит к увеличению времени, требующегося на перенос изменений из дочерней ветви в родительскую ветвь. Пример ветвления в дереве исходного кода:

- **Development** — контейнер для ветвей разработки.
  - **Ветвь разработки**.
    - **Вложенная ветвь первого уровня**.

      Вложенная ветвь второго уровня.
- **Main** — ветвь интеграции.
  - **Source**.
  - **Другие папки ресурсов**.

При слиянии вдоль иерархии ветвей происходит меньше конфликтов. Поэтому при переносе изменений из вложенной ветви второго уровня в ветвь **Main** сначала выполняется перенос во вложенную ветвь первого уровня и ветвь разработки и лишь после этого — перенос в **Main**. Каждый из переносов требует времени на завершение, разрешение конфликтов, сборку и

тестирование. Умножьте это время на количество уровней ветвей, имеющихся в созданной вами структуре.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Избегайте ветвления без необходимости

Выполняйте ветвление только тогда, когда ваша команда столкнется с реальной необходимостью одновременной работы над одним и тем же набором файлов. Если вы не уверены в необходимости ветвления, пометьте сборку и создайте ветвь от этой сборки позднее. Слияние ветвей может оказаться дорогостоящим мероприятием, особенно, если между ними существуют значительные различия.

Для выполнения слияния и разрешения конфликтов требуется один или несколько разработчиков. После слияния исходный код должен быть тщательно протестирован, так как при разрешении конфликтов часто делаются ошибки, способные дестабилизировать сборку.

Особенно сложно проводить слияние поперек иерархии ветвей. При этом требуется ручная обработка многих конфликтов, которые в других обстоятельствах могли бы обрабатываться автоматически.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Избегайте слияния без основы

По возможности, избегайте слияний без основы. Проводя подобные слияния, TFS не располагает информацией о связях файлов и папок в ветвях, слияние которых выполняется. Как правило, это приводит к большему количеству конфликтов слияния и дополнительным конфликтам в предстоящих слияниях.

Структурируйте деревья ветвей таким образом, чтобы слияния выполнялись только вдоль иерархии (вверх и вниз по дереву), но не поперек. Ветвление поперек иерархии как раз и требует слияния без основы.

Выполняя слияние, имейте в виду следующее. Слияние вдоль иерархии, от дочерней папки к родительской и наоборот, приводит к меньшему количеству конфликтов, чем слияние поперек иерархии. Кроме того, иерархия ветвей, основанная на родительских и дочерних ветвях, может отличаться от физической структуры исходного кода на диске. Например:

- **Физическая структура**.
  - **Development** — ветвь разработки.
  - **Main** — ветвь интеграции.
  - **Releases** — контейнер для ветвей выпусков.
    - **Release 1** — ветвь выпуска.
- **Логическая структура**.
  - **Main**.
    - **Development**.
    - **Release 1**.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Предпочитайте полное слияние выборочному

Вместо слияния отдельных изменений между ветвями (выборочное слияние) проведите полное слияние всей ветви. Выбирать отдельные изменения в ветви для слияния может быть и удобно, однако, если набор выборочных изменений попадает в середину диапазона будущего слияния, его слияние будет выполнено повторно. Это может привести к дополнительным конфликтам.

Это особенно актуально при поведении слияния из командной строки с использованием параметра **discard**. При этом может быть выбран отмененный набор изменений, попавший в середину диапазона будущего слияния.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Чаще выполняйте слияния

Не задерживайтесь со слияниями ветвей, особенно когда команды работают над одним выпуском, но в изолированных ветвях. Это обеспечит совместимость изменений.

Расписание слияний зависит от сложности структуры ветвей, а также от потребностей команды разработчиков. В структуре ветвей средней сложности, пример которой представлен ниже, перенос изменений из ветвей разработки в главную ветвь может проводиться ежедневно, а обратный перенос из ветви интеграции — раз в два дня.

- **Development** — папка для изолированных ветвей разработки.
  - **External** — ветвь внешней зависимости.
  - **Team 1** — ветвь команды.
  - **Team 2** — ветвь команды.
    - **Feature A** — ветвь функции.
    - **Feature B** — ветвь функции.
    - **Feature C** — ветвь функции.

- **Main** — ветвь интеграции.
- **Releases** — папка для ветвей выпуска.
  - **Release 2** — ветвь выпуска.
- **Safe Keeping** — папка для хранения архивных копий ветвей.
  - **Release 1** — архивная ветвь.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Всегда создавайте папку верхнего уровня для нового командного проекта

Начинайте новый проект с создания папки, обычно **Main**, в папке командного проекта. Поместите в эту папку весь исходный код главной ветви. Если вам потребуется создать новую ветвь, сделайте это непосредственно из папки Main.

Далее показан пример типичной структуры ветвления:

- **Development** — папка для изолированных ветвей разработки, ответвление Main.
  - **Feature A**.
    - **Source**.
  - **Feature B**.
    - **Source**.
- **Main** — ветвь компоновки и сборки.
  - **Source**.
  - **Папки с другими ресурсами**.
- **Releases** — папка для ветвей выпусков, ответвление Main.
  - **Release 1** — ветвь сопровождения.
    - **Source**.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Используйте перед слиянием параметры Candidate или Preview

Перед выполнением слияния используйте параметры \candidate или \preview, чтобы проверить результаты слияния. Хотя эта возможность доступна только из командной строки, пренебрегать ею не следует. Она позволяет узнать, слияние каких файлов и версий будет выполнено при запуске команды слияния. С ее помощью, например, можно удостовериться, что реальный масштаб слияния не превышает ваши ожидания и что вы правильно оценили все последствия предпринимаемого слияния. При выполнении больших слияний отчет о слиянии поможет вам разделить работу между разработчиками или группами.

Для предварительного просмотра результатов слияния запустите инструмент командной строки Tf.exe с командой **merge** и параметрами **preview** или **candidate**, например:

```
Tf merge main/source development/feature/source /preview
```

Параметр **preview** позволяет предварительно просмотреть результаты слияния, а параметр **candidate** выводит список всех наборов изменений исходного кода, слияние которых еще не выполнено. В список включаются идентификаторы наборов изменений, перенос которых еще не состоялся, и другая общая информация об этих наборах.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.

- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Если в составе слияния выполняется переименование, обращайте внимание на предлагаемый системой путь

Когда частью процесса слияния являются переименованные файлы, обращайте особое внимание на рекомендуемый системой путь и в случае необходимости внесите в него изменения. Все переименованные файлы отмечаются как конфликты. В процессе слияния переименованного объекта алгоритм слияния TFS пытается найти наилучший конечный путь. Иногда путь по умолчанию оказывается не лучшим вариантом, поэтому вам следует проверить его перед передачей файла.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Соблюдайте осторожность при разрешении конфликтов слияния

Выполняйте слияние с осторожностью — допущенные при этом ошибки могут привести к нестабильности сборки.

- Перед слиянием перепроверьте код, предназначенный для слияния.
- Проверьте, удается ли скомпилировать конечный файл слияния, перед его возвратом в систему управления исходным кодом.
- Перед возвратом файла в систему управления исходным кодом убедитесь, что успешно выполняются соответствующие модульные тесты.

- Убедитесь, что вам понятны изменения, внесенные другим разработчиком. Если вы сомневаетесь в целесообразности переноса некоторых строк, поговорите с разработчиком, внесшим изменения, убедитесь в правильности изменений и еще раз продумайте последствия выполняемого слияния.

Завершив слияние, скомпилируйте получившийся код и запустите модульные тесты для проверки на наличие серьезных ошибок.

Дополнительные ресурсы

- Дополнительную информацию о разрешении конфликтов слияний вы найдете в статье «Resolving Conflicts» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms181432(VS.80).aspx*.

Не возвращайте результаты двух и более слияний одновременно

Возвращайте результаты слияния, прежде чем выполнить другое слияние, использующее те же файлы. Завершив первое слияние, скомпилируйте исходный код и убедитесь в успешном завершении модульных тестов, после чего возвратите отложенные изменения. Далее начните второе слияние, придерживаясь той же процедуры.

Выполнение слияний по отдельности упрощает процесс и позволяет при необходимости отменить изменения.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Проводите тесты после слияния, но перед возвратом

После слияния убедитесь, что код компилируется, а соответствующие тесты успешно выполняются, и лишь потом возвращайте файл слияния. Это позволяет избежать нестабильности сборки в результате выполненных слияний.

Изначально результаты слияния изолированы в вашей рабочей области и не выгружаются на сервер, пока вы не вернете отложенные изменения.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Возврат после правки и политики возврата

- Возвращайте код, только убедившись в его надежности.
- Используйте наборы отложенных правок для архивации или совместного использования незавершенных изменений.
- Разрешайте только один рабочий элемент за один возврат.
- Применяйте политики возврата для соблюдения стандартов программирования.
- Применяйте политики возврата для проверки качества кода.
- Следите за перекрытием политики.
- Избегайте конфликтов при помощи планирования.

Возвращайте код, только убедившись в его надежности

Возвращайте обновленный код в систему управления исходным кодом только после того, как он полностью прошел модульное тестирование и готов к использованию остальной частью команды. На промежуточном этапе работы используйте наборы отложенных правок (shelvesets).

После возвращения код будет использоваться в составе следующей плановой сборки. Если работа над кодом не завершена, он может стать причиной нестабильности сборки. Кроме того, любой разработчик, выполнив команду **get latest**, перенесет ваши изменения к себе. Если они не завершены или недостаточно протестированы, у вашего коллеги могут возникнуть проблемы.

Дополнительные ресурсы

- Дополнительную информацию о возврате после правки вы найдете в статье «How to: Check In Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181411(VS.80).aspx*.

Используйте наборы отложенных правок для архивации или совместного использования незавершенных изменений

Используйте наборы отложенных правок (shelveset) для архивации файлов, содержащих не готовые к возврату изменения. Их также можно использовать для предоставления общего доступа к коду с ознакомительными целями, а также при передаче задания другому разработчику. С помощью наборов отложенных правок вы можете выгружать незавершенные изменения на сервер, не возвращая их в систему управления исходным кодом, если, например, работа над ними еще не закончена и изменения могут привести к нестабильности сборки.

Чтобы отложить правки, сначала просмотрите их список: щелкните правой кнопкой решение в окне обозревателя Solution Explorer и выберите команду **View Pending Changes**. Выберите файлы, которые хотите отложить и щелкните **Shelve**. Введите имя набора и комментарий, поясняющий его назначение, а затем щелкните **Shelve**.

Извлечение набора отложенных правок

1. Откройте меню **File**, разверните подменю **Source Control** и выберите команду **Unshelve**.
2. В поле **Owner name** введите имя создателя набора отложенных правок (например, Contoso\JimB или просто JimB) и щелкните **Find**.
3. В области **Results** выберите набор отложенных правок, который хотите загрузить в рабочую область и щелкните **Details**.
4. Чтобы удалить набор отложенных правок после его извлечения с сервера управления исходным кодом TFS, сбросьте флажок **Preserve shelveset on server**. Система TFS восстановит каждую отложенную правку в целевую рабочую область в качестве незавершенного изменения, при условии что правка не конфликтует с уже существующим отложенным изменением в этой же рабочей области.
5. При необходимости сбросьте флажок **Restore work items and check-in notes**, если не хотите загружать рабочие элементы и заметки о возврате восстанавливаемого набора. В открывшемся диалоговом окне **Details** выберите наборы отложенных правок или их элементы, которые хотите поместить в рабочую область, и щелкните **Unshelve**.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Разрешайте только один рабочий элемент за один возврат

После разрешения рабочего элемента верните отложенные изменения и обновите статус рабочего элемента. Суть выполнения данной процедуры в следующем:

- Она обеспечивает полноту информации в БД системы управления исходным кодом, что может понадобиться для анализа изменений по данному рабочему элементу или для восстановления рабочего элемента в будущем.
- Она гарантирует, что возвраты после правок производятся достаточно часто. Чем больше интервал между возвратами, тем больше вероятность возникновения конфликта, требующего проведения ручного слияния и дополнительного тестирования.

Дополнительные ресурсы

- Дополнительные сведения о возврате после правки вы найдете в статье «How to: Check In Pending Changes» по адресу http://msdn2.microsoft.com/en-us/library/ms181411(VS.80).aspx
- Дополнительную информацию о рабочих элементах и наборах изменений вы найдете в статье «How to: Associate Work Items with Changesets» по адресу *http://msdn2.microsoft.com/en-us/library/ms181410(VS.80).aspx*.
- Подробнее об анализе сведений о рабочем элементе читайте в статье «How to: View Work Item Details from Pending Changes Window» по адресу *http://msdn2.microsoft.com/en-us/library/ms245467(VS.80).aspx*.

Применяйте политики возврата для соблюдения стандартов программирования

Используйте политики возврата для соблюдения правил программирования внутри команды разработчиков. Политики возврата помогают обеспечить соответствие кода, возвращаемого в систему TFS, заданным стандартам.

Стандартные политики возврата из комплекта TFS позволяют проводить обязательный статический анализ кода перед его возвратом. Вы можете настроить политику анализа кода для проверки различных правил. Например, вы можете проверять выполнение правил проектирования, способности к взаимодействию, сопровождения, переносимости, надежности, а также соглашений об именах и пр.

Чтобы применить в командном проекте политику анализа кода при возврате, выполните следующие действия:

1. В окне Team Explorer щелкните правой кнопкой нужный командный проект, раскройте подменю **Team Project Settings** и выберите команду **Source Control**.
2. Перейдите на вкладку **Check-in Policy** и щелкните **Add**.

3. В диалоговом окне Check-in Policy установите параметр **Code Analysis** и щелкните **OK**.
4. В окне Code Analysis Policy Editor установите **Enforce C/C++ Code Analysis (/analyze)** или **Enforce Code Analysis for Managed Code**. Установите оба параметра, если проект содержит как управляемый, так и неуправляемый код.
5. Если вы выбрали параметр **Enforce Code Analysis for Managed Code**, настройте правила анализа управляемого кода в соответствии с вашими стандартами программирования.

Вы также вольны создать собственную политику возврата, позволяющую выполнять проверки, не входящие в состав стандартной политики. В частности, можно запретить вызовы функций API запрещенных приложений или задать нормативы определенного стиля программирования, например, в каких местах следует помещать фигурные скобки.

Дополнительные ресурсы

- Дополнительную информацию о создании пользовательской политики возврата вы найдете в разделе «Как создать собственную политику возврата TFS» этой книги.
- О настройке политики возврата читайте в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- Пример кода с запретом на возврат определенных фрагментов вы найдете в статье «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- Пример запрета на возврат кода без комментариев вы найдете в статье «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- О том, как зарегистрировать созданную вами политику, читайте в статье «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Применяйте политики возврата для проверки качества кода

Сочетание политик анализа и тестирования кода позволяет осуществлять контроль его качества. Например, при помощи стандартной политики тестирования можно указать, что код будет допущен к возврату в систему управления исходным кодом TFS только после проведения заданных тестов. Вы также вольны настроить политику анализа кода, чтобы гарантировать соответствие кода заданным стандартам качества, то есть, соблюдение правил безопасности, производительности, переносимости, удобства сопровождения и надежности.

Чтобы применить политику анализа кода при возврате в систему управления исходным кодом, выполните следующие действия:

1. В окне Team Explorer щелкните правой кнопкой нужный командный проект, раскройте подменю **Team Project Settings** и выберите команду **Source Control**.
2. Перейдите на вкладку **Check-in Policy**, щелкните кнопку **Add**, а затем выберите и настройте определенную политику.

Дополнительные ресурсы

- Дополнительную информацию о создании пользовательской политики возврата вы найдете в разделе «Как создать собственную политику возврата TFS» этой книги.
- О настройке политики возврата читайте в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- Пример кода с запретом на возврат определенных фрагментов вы найдете в статье «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- Пример запрета на возврат кода без комментариев вы найдете в статье «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- О том, как зарегистрировать созданную вами политику, читайте в статье «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Следите за перекрытием политики

Система Team Foundation Version Control не предотвратит перекрытие политики. Однако, выполнив следующие действия, вы сможете установить факт перекрытия политики:

- Используйте службу Team Foundation Eventing Service из Team Foundation Core Services API, чтобы внедриться в события возврата.
- Напишите метод **Notify**, который анализирует параметры набора изменений и реагирует на перекрытие, если оно случится.

Другой способ заключается в просмотре истории набора изменений с целью обнаружения перекрытий политики.

Дополнительные ресурсы

- Дополнительную информацию о перекрытии политики возврата вы найдете в статье «How to: Override a Check-in Policy» по адресу *http://msdn2.microsoft.com/en-us/library/ms245460(VS.80).aspx*.

- Дополнительную информацию о службе Team Foundation Eventing Service вы найдете в статье «Eventing Service» по адресу *http://msdn2.microsoft.com/en-us/library/bb130154(vs.80).aspx*.
- Чтобы узнать, как настроить отправку уведомлений по электронной почте при нарушении политики, читайте запись в блоге по адресу *http://blogs.infosupport.com/marcelv/archive/2005/10/18/1635.aspx*.

Избегайте конфликтов при помощи планирования

Планируйте работу так, чтобы несколько разработчиков не работали одновременно над одним и тем же фрагментом исходного кода. Иначе возникнут конфликты, разрешить которые бывает непросто. Конечно, часто помогает автоматическое разрешение конфликтов, но оно не поможет, если два или несколько разработчиков работают над одним и тем же методом или одними и теми же строками кода. Конфликты в строках требуют разрешения вручную, что усложняет слияние. Ключ к решению проблемы — эффективное взаимодействие в команде.

Перед началом работы над файлом убедитесь, что извлекли из системы управления исходным кодом последнюю версию этого файла, а также проверьте, не работает ли с этим же файлом кто-то еще. Если с файлом уже работает ваш коллега, спросите, что именно он делает, и решите, стоит ли вам дождаться окончания его работы, или же вам можно действовать параллельно, поскольку вы работаете над разными функциями в разных участках исходного кода.

Чтобы выяснить, извлек ли кто-то файл или нет, выполните следующие действия:

1. В окне Team Explorer в Visual Studio щелкните дважды **Source Control**.
2. Перейдите в папку, содержащую файл, который вы хотите проверить. Для всех незавершенных изменений указано имя пользователя, владеющего этими изменениями.

Чтобы узнать, для каких файлов в данный момент имеются незавершенные изменения, в командной строке Visual Studio 2005 введите следующую команду:

```
Tf status /format:detailed /user:*
```

Начиная работу с файлом исходного кода, который будет задействован кем-то еще, сообщите другим членам команды, что работаете над файлом и поясните, что конкретно будете обновлять.

Дополнительные ресурсы

- Дополнительную информацию о просмотре отложенных изменений в рабочей области вы найдете в статье «How to: View and Manage All Pending Changes in Your Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181400(VS.80).aspx*.

- Дополнительная информация о просмотре отложенных изменений в других рабочих областях находится в статье «How to: View Pending Changes in Other Workspaces» по адресу *http://msdn2.microsoft.com/en-us/library/ms181401(VS.80).aspx*.

Получение и блокировка кода

- Перед внесением изменений извлеките последнюю версию исходного кода.
- Осмотрительно пользуйтесь командой lock.
- Предупредите коллег о блокировке файла.

Перед внесением изменений извлеките последнюю версию исходного кода

Перед извлечением кода для редактирования убедитесь, что имеете дело с последними версиями всех файлов исходного кода проекта. Чтобы получить их, запустите команду **get latest**. Если этого не сделать, есть опасность того, что вы будете писать код в устаревших версиях файлов, а последующий возврат вашего кода на сервер приведет к проблемам со сборкой.

Чтобы получить последние копии файлов, относящихся к тому или иному проекту, щелкните правой кнопкой командный проект в окне обозревателя Source Control и выберите команду **Get Latest Version**. Если в данный момент в вашей рабочей области имеются записываемые файлы с незавершенными правками, эти файлы не будут перезаписаны. Эту команду можно запустить и из командной строки, введя **Tf get /all** в папке, сопоставленной с текущей рабочей областью.

Дополнительные ресурсы

- Дополнительные сведения о команде Get вы найдете в статье «Get Command» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/fx7sdeyf(VS.80).aspx*.

Осмотрительно пользуйтесь командой lock

Не стоит слишком увлекаться использованием команды **lock**. Блокирование файла во время работы над ним может отрицательно сказаться на производительности всего процесса разработки, поскольку другие пользователи не смогут одновременно работать над этим же файлом. Блокировка файла на время редактирования нужна только в случае, если вы опасаетесь возникновения конфликтов, которые могут вынудить вас выполнять сложное ручное слияние. По возможности разрешайте совместную работу, во время извлечения файла устанавливая тип блокировки **None**.

Ниже приведены сценарии, при которых использование команды **lock** оправдано:

- Вы корректируете файл в двоичном формате, например, изображение или документ, слияние которого затруднено.
- Вносимые вами поправки настолько распространены, что другой разработчик, работая над тем же файлом, скорее всего, изменит те же самые строки, что и вы.

Чтобы избежать любой возможности возникновения конфликта, блокируйте файл, извлекая его для редактирования. Это не даст другим пользователям извлекать этот же файл для редактирования и возвращать его. Обязательно известите своих коллег о причинах блокировки и примерном времени, которое потребуется вам для редактирования. Также сообщите им, что сняли блокировку и вернули файл с изменениями.

Можно назначить тип блокировки в процессе извлечения файла для редактирования или заблокировать файл явным образом.

Чтобы заблокировать файл в процессе извлечения, выполните следующие действия:

1. В окне обозревателя Source Control щелкните файл правой кнопкой и выберите команду **Check Out for Edit**.
2. Укажите тип блокировки — **None, Check Out** или **Check In**.
3. Чтобы заблокировать файл явно, щелкните его правой кнопкой мыши и выберите команду **Lock**. Затем укажите тип блокировки — **Check Out** или **Check In**.

В отличие Microsoft Visual Source Safe® (VSS), при извлечении файла в TFS вам не предлагается по умолчанию последняя версия. Прежде чем заблокировать файл, поместите в рабочую область его последнюю версию, выполнив команду **Get Latest Version**.

Дополнительные ресурсы

- Подробнее о блокировках читайте в статье «How to: Lock and Unlock Folders or Files» по адресу *http://msdn2.microsoft.com/en-us/library/ms181420(VS.80).aspx*.
- Дополнительную информацию о типах блокировок вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181419(VS.80).aspx*.

Предупредите коллег о блокировке файла

Блокируя файл исходного кода, оповестите об этом членов команды, чтобы они могли планировать свою работу с учетом недоступности файла. Объясните, почему вам нужен исключительный доступ к файлу и как долго будет сохраняться блокировка. Блокировка файла может стать причиной снижения производительности в цикле разработки, когда нескольким разработчикам нужно работать над одним и тем же исходным кодом.

Блокировка файла на время редактирования нужна только в случае, если вы опасаетесь возникновения конфликтов, которые могут вынудить вас выполнять сложное ручное слияние. По возможности разрешайте совместную работу, во время извлечения файла устанавливая тип блокировки **None**.

Чтобы заблокировать файл в окне обозревателя Source Control, щелкните файл правой кнопкой мыши и выберите команду **Lock.** Затем укажите тип блокировки — **Check Out** или **Check In**.

Дополнительные ресурсы

- Дополнительную информацию о типах блокировок вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181419(VS.80).aspx*.

Зависимости

- Старайтесь использовать ссылки на проект.
- Без необходимости не используйте файловые ссылки.
- Для ссылок на проект и файл используйте параметр copy local = true.
- В ссылках на веб-службы используйте динамические URL.

Старайтесь использовать ссылки на проект

По возможности используйте ссылки на проект, потому что они обладают следующими преимуществами:

- Работают на всех рабочих станциях, где загружено решение и набор проектов. Это происходит из-за того, что глобально уникальный идентификатор (GUID) проекта расположен в файле проекта, что однозначно идентифицирует проект, на который имеется ссылка, в контексте текущего решения.
- Позволяют системе сборки Visual Studio отслеживать зависимости проекта и устанавливать порядок выполнения сборок проекта.
- Позволяют избежать возможной потери ссылок на файлы сборок на конкретном компьютере.
- Автоматически отслеживают изменения конфигурации проекта. В частности, при сборке в конфигурации Debug любые ссылки проекта указывают на сборки Debug, сгенерированные проектами, на которые ссылается сборка. При сборке в конфигурации Release эти же ссылки указывают на сборки Release. Это означает, что вы можете автоматически переходить от отладочных сборок к сборкам выпуска без перенастройки ссылок.
- Позволяют Visual Studio обнаруживать и предотвращать циклические зависимости.

Вы можете использовать ссылки на проект, если файл сборки находится в наборе проектов решения. Если файл сборки находится за пределами набора

проектов решения, а вам нужна ссылка на проект, вы можете выполнить ветвление из исходного проекта в ваш проект. Когда потребуется обновить версию зависимости, произведите слияние из исходного проекта в вашу ветвь.

Дополнительные ресурсы

- Дополнительную информацию о ссылках на проект и файловых ссылках вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.
- Подробнее о добавлении ссылок читайте в статье «How to: Add or Remove References in Visual Studio» по адресу *http://msdn2.microsoft.com/en-us/library/wkze6zky(VS.80).aspx*.

Без необходимости не используйте файловые ссылки

Иногда невозможно использовать ссылку на проект, например, когда требуется создать ссылку на файл сборки за пределами набора проектов текущего решения и вы не хотите выполнять ветвление из исходного проекта в ваш проект. В этом случае вам придется установить файловую ссылку.

Дополнительные ресурсы

- Дополнительную информацию о ссылках на проект и файловых ссылках вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.

Для ссылок на проект и файл используйте параметр copy local = true

С каждой ссылкой сопоставлен атрибут **copy local**. В Visual Studio значение этого атрибута (TRUE или FALSE) задается при первом добавлении ссылки. Значение FALSE присваивается, если сборка находится в глобальном кеше сборок (GAC). В противном случае, присваивается значение TRUE. Значение, заданное по умолчанию, изменять не следует.

Если атрибут **copy local** имеет значение **true**, система сборки Visual Studio копирует любую сборку, на которую имеется ссылка, а также все зависимые нисходящие сборки, в клиентскую выходную папку. Например, если клиентский проект ссылается на сборку Lib1, а Lib1 зависит от Lib2 и Lib3, то во время сборки Visual Studio автоматически копирует Lib1, Lib2 и Lib3 в локальную выходную папку вашего проекта.

Дополнительные ресурсы

- Дополнительную информацию о ссылках на проект и файловых ссылках вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.

В ссылках на веб-службы используйте динамические URL

Чтобы вызвать веб-службу, следует добавить в проект веб-ссылку. Так генерируется прокси-класс, посредством которого вы взаимодействуйте с веб-службой. Код прокси изначально содержит URL веб-службы, например, http://localhost или http://SomeWebServer.

> **Важно!** Для веб-служб, выполняющихся на вашем компьютере, всегда используйте адрес http://localhost а не http://ИмяМоегоКомпьютера, чтобы сохранить работоспособность ссылки на любом компьютере.

Статический URL, внедренный в прокси, как правило, не совпадает с URL, который будет использоваться в испытательной и рабочей среде. Обычно URL меняется, по мере того как ваше приложение проходит путь от разработки к тестированию и производству. Есть три способа решения этого вопроса:

- Программно задавать URL веб-службы во время создания экземпляра прокси-класса.
- Присвоить свойству **URL Behavior** ссылки на веб-службу значение **Dynamic**. Это предпочтительный вариант. При этом в прокси-класс добавляется код, извлекающий URL веб-службы из пользовательского раздела файла конфигурации приложения — Web.config для веб-приложения или SomeApp.exe.config для приложений Windows.
- Сгенерировать прокси с помощью инструмента командной строки WSDL.exe, задав параметр **/urlkey**. При этом происходит примерно то же самое, что и при установке свойства **URL Behavior**, но в этом случае URL хранится в разделе <applicationSettings> файла конфигурации приложения.

Использование динамического URL позволяет создавать пользовательский файл конфигурации, который способен перекрывать параметры основного файла конфигурации приложения. Это позволяет разработчикам и членам группы тестирования временно перенаправить ссылку веб-службы в другое положение.

Дополнительные ресурсы

- Дополнительную информацию об управлении зависимостями вы найдете в главе 6 этой книги.

Распределенная и удаленная разработка

- Убедитесь, что объема диска достаточно для реализации прокси.
- Создайте задание для получения последних файлов по расписанию.
- Регулярно проверяйте счетчики производительности прокси и журнал событий.

- Настраивайте executionTimeout соответственно размеру файла и ширине канала.
- Отключайте прокси, если он не будет работать продолжительное время.
- Скрывайте рабочие области, чтобы избежать ненужного перемещения файлов.

Убедитесь, что объема диска достаточно для реализации прокси

Следуйте системным требованиям, приведенным в руководстве по установке TFS. В особенности это касается объема жесткого диска. Верхняя граница задается, исходя из объема дискового пространства, которое прокси может использовать для кеширования файлов. По достижению этого предела старые файлы удаляются из кеша, чтобы освободить место для новых файлов. При очистке диска файлы удаляются в соответствии с датой последнего доступа к ним. Файл, который не используется дольше остальных, будет удален первым.

Дополнительные ресурсы

- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Создайте задание для получения последних файлов по расписанию

Для извлечения новейших файлов в прокси-сервер регулярно запускайте запланированное задание. Это обеспечит наличие последних версий файлов в кеше прокси, а значит, последующие запросы клиентами этих файлов будут выполняться из кеша.

Дополнительные ресурсы

- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Регулярно проверяйте счетчики производительности прокси и журнал событий

Периодически проверяйте счетчики производительности прокси и журнал событий Windows на предмет наличия ошибок и предупреждений, чтобы получить наглядное представление об эффективности прокси. Убедитесь, что на прокси включена функция кеширования, и следите за производительностью кеша.

Вот список счетчиков производительности, за которыми нужно следить:

- Текущий размер кеша (Current Cache Size).
- Всего попаданий в кеш (Total Cache Hits).
- Всего запросов на загрузку (Total Download Requests).
- Всего файлов в кеше (Total Files in Cache).
- Всего пропусков кеша (Total Cache Miss).

Счетчики производительности регистрируются при установке прокси. Они рассчитаны на работу с несколькими экземплярами, то есть, каждому уровню приложений, заданному в файле Proxy.config, соответствует свой набор счетчиков. Собирая показания счетчиков, вы получите общую картину производительности прокси-сервера.

TFS Proxy сохраняет статистические данные о производительности кеша в XML-файле ProxyStatistics.xml. Вы вольны задать временной интервал для сохранения этих данных. Файл ProxyStatistics.xml находится в подпапке App_Data папки установки прокси.

Дополнительные ресурсы

- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Настраивайте executionTimeout соответственно размеру файла и ширине канала

Если вам предстоит загрузка больших файлов по сети с невысокой пропускной способностью (менее 3 Мбит/с), исправьте значение параметра executionTimeout в файле Web.config, чтобы сократить вероятность истечения срока передачи. Значение по умолчанию равно одному часу:

```
<httpRuntime executionTimeout="3600"/>.
```

Дополнительные ресурсы

- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Отключайте прокси, если он не будет работать продолжительное время

Отключайте прокси на клиентских компьютерах, если прокси не будет работать длительное время. Иначе клиенты будут предпринимать бесполезные попытки восстановить подключение. По умолчанию, эти попытки повторяются каждые пять минут.

Дополнительные ресурсы

- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Скрывайте рабочие области, чтобы избежать ненужного перемещения файлов

Во избежание ненужных перемещений файлов используйте сокрытие рабочей области. При этом не только достигается сокрытие определенных папок рабочих областей, но и повышается производительность системы обработки данных. Запретив копирование не нужных в данный момент файлов и папок в рабочую область, вы сохраните место на локальном диске. Вы можете скрыть существующее сопоставление папки в рабочей области, но лучше создать новое сопоставление, специально предназначенное для сокрытия.

Чтобы скрыть папки в рабочей области, выполните следующие действия:

1. В Visual Studio в меню **File** раскройте подменю **Source Control** и выберите команду **Workspaces**.
2. В диалоговом окне **Manage Workspaces** выберите рабочую область, к которой хотите применить сокрытие, и щелкните кнопку **Edit**.
3. В диалоговом окне **Edit Workspaces** выделите в списке **Working Folders** сопоставление, которое хотите скрыть, или создайте новое сопоставление.
4. Щелкните столбец **Status** и измените его значение с **Active** на **Cloak**.
5. Щелкните **OK**, чтобы закрыть диалоговые окна **Edit Workspaces** и **Manage Workspaces**.

Помните, что файлы не будут локально скрыты, пока вы повторно не выполните команду **get** для всей рабочей области.

Дополнительные ресурсы

- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Переход из других версий

- Переходите на Team Foundation Server Source Control при помощи VSS Converter
- Переходите на Team Foundation Server Source Control из других систем управления исходным кодом

Переходите на Team Foundation Server Source Control при помощи VSS Converter

В комплекте Team Foundation Server поставляется инструмент VSS Converter, позволяющий переносить файлы, папки, историю версий, метки и пользовательскую информацию из БД Visual SourceSafe в систему управления исходным кодом Team Foundation Server.

Следует знать, что конвертер имеет некоторые ограничения, например:

- Не сохраняется история общего доступа к файлам, поскольку система Team Foundation Server не поддерживает общий доступ. Перемещение совместно используемого файла сводится к копированию этого файла в конечную папку.
- Не сохраняется история ветвления.
- Система Team Foundation Server не поддерживает закрепление (pinning). Закрепленные файлы переносятся с созданием двух меток.
- Во время переноса не сохраняются временные метки, связанные с действиями.

Дополнительные ресурсы

- Дополнительную информацию о перемещении файлов вы найдете в разделе «Как осуществить перенос исходного кода из Visual SourceSafe» этой книги.

Переходите на Team Foundation Server Source Control из других систем управления исходным кодом

Вы можете вручную выполнить экспорт файлов из предыдущей системы управления версиями, а затем импортировать их в систему управления версиями TFS. Если вам нужно сохранить историю файлов или другие атрибуты предыдущей системы, воспользуйтесь инструментарием TFS Migration and Synchronization Toolkit, который можно загрузить по адресу *http://www.codeplex.com/MigrationSyncToolkit*. При помощи этого инструментария вы сможете написать собственное средство для выполнения миграции.

В данный момент корпорация Microsoft ведет работу по созданию конвертера ClearCase. О выпуске конвертера будет объявлено дополнительно в блоге TFS Migration, расположенном по адресу *http://blogs.msdn.com/tfs_migration*.

Существует конвертер, созданный компанией Component Software, который совместим с системами GNU RCS, CS-RCS, GNU CVS, Subversion (SVN) и Visual SourceSafe (VSS).

Дополнительные ресурсы

- Дополнительную информацию о переходе на TFS вы найдете в блоге «TFS Migration», расположенном на сайте MSDN по адресу *http://blogs.msdn.com/tfs_migration*.

- Загрузить набор инструментов TFS Migration and Synchronization Toolkit можно по адресу *http://www.codeplex.com/MigrationSyncToolkit*.
- Дополнительную информацию о конвертере компании Component Software вы найдете на сайте компании по адресу *http://www.componentsoftware.com/Products/Converter/*.

Управление проектами и рабочими областями

- Для изоляции одного разработчика используйте рабочие области, а не ветвление.
- Удаляйте и переименовывайте файлы в системе управления исходным кодом, а не в проводнике Windows.
- Удаляйте и переименовывайте файлы только при открытом решении.
- Создавайте один командный проект на приложение, если хотите переносить ресурсы от версии к версии.
- Создавайте командный проект для каждой версии, если хотите с каждой новой версией начинать все заново.
- Используйте ветвление для обеспечения совместного доступа к исходному коду и двоичным файлам.
- Не используйте сопоставление для поддержки зависимостей между различными проектами.
- Сопоставляйте рабочие области на корневом уровне командного проекта.
- Используйте уникальный путь к локальной папке на общих компьютерах.
- Сопоставляйте только части дерева исходного кода.
- Структурируйте дерево исходного кода с учетом ветвления.

Для изоляции одного разработчика используйте рабочие области, а не ветвление

Чтобы изолировать свою работу от остальной команды, используйте дополнительную рабочую область, но не создавайте новую ветвь. При этом первая рабочая область будет содержать ссылки на файлы и папки, над которыми работает остальная команда (то есть, общий исходный код), а вторая предназначается для файлов и папок, которые вы хотите изолировать. Это может потребоваться, например, если вы хотите поработать над файлами отдельно от основного процесса разработки, в частности, при внесении рискованных или экспериментальных изменений. Использование второй рабочей области позволяет избежать дополнительных сложностей при ветвлении и слиянии.

Чтобы создать вторичную рабочую область, выполните следующие действия:

1. В окне обозревателя Source Control выберите в раскрывающемся списке **Workspace** вариант **Workspaces**.

2. В диалоговом окне **Manage Workspaces** щелкните кнопку **Add**.
3. В диалоговом окне **Add Workspace** введите имя новой рабочей области, например, *ИзолированнаяРабота*. Добавьте комментарий с описанием цели создания рабочей области.
4. В списке **Working Folders** задайте для рабочей области состояние **Active**. Укажите папку с исходным кодом, которую нужно включить в рабочую область. Это может быть корневая папка командного проекта или любая вложенная папка. Задайте путь на локальном компьютере, где будут храниться файлы рабочей области.
5. Щелкните **OK** и **Close**, чтобы создать изолированную рабочую область.

Чтобы извлечь последнюю версию исходного кода и начать работать с ним в изолированной рабочей области, выполните следующие действия:

1. Убедитесь, что в окне **Source Control** в раскрывающееся списке **Workspace** выбрано имя нужной рабочей области.
2. Выберите корневую папку командного проекта (или вложенную папку, если вам нужна только часть дерева исходного кода), щелкните ее правой кнопкой и выберите команду **Get Latest Version**.

При этом будет выполнено копирование структуры папок и актуального набора файлов с сервера управления исходным кодом в локальную папку на вашем компьютере, сопоставленную с новой рабочей областью.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию о редактировании рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Удаляйте и переименовывайте файлы в системе управления исходным кодом, а не в проводнике Windows

Если вам надо удалить или переименовать файлы, добавленные в систему управления исходным кодом, это следует делать в обозревателе Source Control, запустив Tf.exe из командной строки. Нельзя удалять и переименовывать файлы в проводнике Windows, так как при этом нарушается синхронизация файлов локальной рабочей области с файлами сервера управления исходным кодом.

Порядок действий при удалении файла или папки при помощи Source Control Explorer:

1. В окне Source Control выберите файл или папку.

2. Щелкните правой кнопкой выбранный файл или папку и выберите команду **Delete**.

Слева появится значок, символизирующий удаление объекта. В столбце **Pending Change** отобразится статус **Delete**. Удаление произойдет при очередном возврате правок.

> **Примечание** Невозможно удалить объект, для которого имеется незавершенное изменение. Например, нельзя удалить файл, извлеченный для редактирования.

Особенно удобно использование команд **Tf move, delete** и **rename** при работе с несколькими файлами или папками одновременно. В обозревателе Source Control можно перемещать, переименовывать или удалять только по одному файлу или папке.

Дополнительные ресурсы

- Подробнее о параметрах команды **Tf delete** читайте в статье «Delete Command» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/k45zb450(VS.80).aspx*.
- Подробнее о параметрах команды **Tf rename** читайте в статье «Rename Command» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/a79bz90w(VS.80).aspx*.

Удаляйте и переименовывайте файлы только при открытом решении

Удаление и переименование файлов в Solution Explorer следует выполнять при открытом решении. Не удаляйте их непосредственно с диска. Это гарантирует, что при следующем возврате незавершенных изменений система управления исходным кодом TFS сохранит синхронизацию.

Дополнительные ресурсы

- Подробнее о параметрах команды **Tf delete** читайте в статье «Delete Command» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/k45zb450(VS.80).aspx*.
- Подробнее о параметрах команды **Tf rename** читайте в статье «Rename Command» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/a79bz90w(VS.80).aspx*.

Создавайте один командный проект на приложение, если хотите переносить ресурсы от версии к версии

Если вы планируете переносить из версии в версию не только исходный код, но и рабочие элементы, а также другие ресурсы TFS, используйте один командный проект на одно приложение. При использовании общего команд-

ного проекта для нескольких версий приложения все папки TFS автоматически переносятся в следующий выпуск. Когда все готово к выпуску новой версии приложения, вы можете создать ветвь внутри проекта, чтобы осуществить выпуск и изолировать код.

Используя один проект для каждого приложения, учитывайте следующее:

- Для параллельных выпусков приходится использовать общие схемы рабочих элементов, политики возврата после правки и справочник по процессу.
- Затруднено составление отчетов. Поскольку по умолчанию отчеты составляются для всего проекта, вам придется установить фильтрацию по конкретным выпускам.
- Если количество приложений исчисляется сотнями и у каждого из них есть собственный проект, вы рискуете снизить производительность TFS и выйти за пределы масштабируемости.
- Со временем, после нескольких выпусков, у вас набирается много «багажа». Проще всего решить эту проблему, создав новый проект с ответвлением в него только нужного вам кода.

Дополнительные ресурсы

- Дополнительную информацию об использовании командных проектов вы найдете в статье «When to use Team Projects» по адресу *http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx*.

Создавайте командный проект для каждой версии, если хотите с каждой новой версией начинать все заново

Если вы не хотите переносить рабочие элементы и другие ресурсы TFS из выпуска в выпуск, создавайте один проект на каждый выпуск. Это позволит вам изменять схемы рабочих элементов, рабочую процедуру, политики возврата после правки и другие элементы, не затрагивая прошлые выпуски. Такая организация работы особенно полезна, если предыдущий выпуск будет сопровождаться другой командой, например, группой непрерывной разработки, рабочий процесс в которой может отличаться от основной команды разработчиков.

Используя отдельный проект для каждого нового выпуска, имейте в виду следующее:

- Переместить код из одного проекта в другой достаточно просто, а вот перемещать рабочие элементы и прочие ресурсы TFS из одного проекта в другой очень нелегко. Рабочие элементы можно копировать в другой проект только по одному; если вы хотите скопировать набор рабочих элементов, вам потребуется написать собственную утилиту.

- Если количество ваших приложений и выпусков исчисляется сотнями, причем, все они находятся в собственных проектах, вы рискуете снизить производительность TFS и выйти за пределы масштабируемости.
- Выбирая структуру, думайте о дне завтрашнем: реструктуризация командных проектов — трудное занятие.
- Можно легко организовать общий доступ к исходному коду из нескольких командных проектов:
  - ветвлением исходного кода из одного проекта в другой;
  - сопоставлением исходного кода из одного проекта в другой.
- Система Team Foundation Server способна вместить около 500 проектов с использованием Microsoft Solution Framework (MSF) для шаблона процесса MSF Agile и до 250 проектов с использованием шаблона процесса MSF CMMI. Если вы создаете собственный процесс или выполняете настройку существующего, помните, что схемы рабочих элементов имеют огромное влияние на масштабируемость сервера. Чем сложнее схема, тем меньшее количество проектов способен поддерживать сервер.

Дополнительные ресурсы

- Дополнительную информацию об использовании командных проектов вы найдете в статье «When to use Team Projects» по адресу *http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx*.

Используйте ветвление для обеспечения совместного доступа к исходному коду и двоичным файлам

Управление общим исходным кодом или двоичными файлами производится в два этапа:

1. Определение места для хранения зависимости.

   Возможны следующие варианты:

   - Если общая зависимость четко принадлежит другой команде, храните ее в командном проекте команды-владельца.
   - Если общая зависимость не имеет определенного хозяина, создайте командный проект, специально предназначенный для общего кода.

2. Ветвление зависимости в ваш проект.

   Сохранив зависимость, выполните ветвление общего исходного или двоичного кода в ваш проект. Каждый раз при выполнении слияния из общего проекта в конечный вы получите новейшую версию исходного кода. Это позволяет копировать последние изменения по заданному расписанию и выполнять интеграционную проверку до изменения кода в главном дереве.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Не используйте сопоставление для поддержки зависимостей между различными проектами

Как правило, следует избегать зависимостей, пересекающих командные проекты. Старайтесь объединять все взаимозависимые решения и проекты в общем командном проекте. При этом вам реже придется прибегать к настройке сценария сборки. Если у вас есть зависимость, используйте для ее определения ссылки на проект или создайте для зависимости ответвление из общего проекта в свой проект. Избегайте файловых ссылок, потому что ими сложнее управлять. Исключение составляют случаи, когда разработка зависимого проекта ведется параллельно, и вам нужны изменения в реальном времени. В этом случае стоит воспользоваться сопоставлением рабочей области. Тем не менее, даже в этом случае можно использовать ветвление для изоляции, если зависимый код приводит к возникновению слишком большого числа серьезных ошибок.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию об изменении рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Сопоставляйте рабочие области на корневом уровне командного проекта

В новом командном проекте сопоставляйте корень проекта ($/MyTeamProject) с папкой на локальном диске, имеющей похожее имя, например, C:\TeamProjects. Поскольку сопоставления являются рекурсивны-

ми, вся структура локальной папки создается автоматически и будет в точности повторять структуру в системе управления исходным кодом.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию об изменении рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Используйте уникальный путь к локальной папке на общих компьютерах

Два пользователя одного компьютера не могут использовать одно и то же сопоставление рабочей области. Допустим, вы и ваш коллега не можете сопоставить один командный проект ($/MyTeamProject) с одной и той же папкой на локальном компьютере. Создавайте сопоставления в папке Мои документы (хотя это удлиняет путь) или разработайте соглашение об именах для папок на локальном компьютере (например, C:\TeamProjects\User1, C:\TeampProjects\User2 и т.д.).

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию об изменении рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Сопоставляйте только части дерева исходного кода

Чтобы повысить производительность и сократить использование диска, сопоставляйте только те файлы, которые требуются для проекта разработки. В основном, вам требуются только файлы и проекты, связанные с решением, над которым вы работаете.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию об изменении рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Структурируйте дерево исходного кода с учетом ветвления

Структура дерева исходного кода состоит из сочетания структуры папок, структуры файлов и структуры ветвей. Внутри главной ветви в различных по размеру командах хорошо зарекомендовала себя следующая структура папок и файлов:

- **Main** — контейнер для всех объектов, нужных для отправки проекта заказчику.
  - **Source** — контейнер для всех объектов, требующихся для выполнения сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, используемого совместно с другими проектами.
    - **Unit Tests** — контейнер для модульных тестов.
    - **Lib** — контейнер для двоичных зависимостей.
  - **Docs** — контейнер для документации, поставляющейся с проектом.
  - **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
  - **Tests** — контейнер с результатами тестов, проводившихся тестовой командой.

Любое ветвление, проводимое не в папке Main, повлечет за собой копирование структуры папок и файлов в новую ветвь, например:

- **Development** — ветвь разработки.
  - **Source** — контейнер для всех объектов, требующихся для выполнения сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, используемого совместно с другими проектами.
    - **Unit Tests** — контейнер для модульных тестов.
    - **Lib** — контейнер для двоичных зависимостей.
- **Main** — Ветвь интеграции.
  - **Source** — контейнер для всех объектов, требующихся для выполнения сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, используемого совместно с другими проектами.
    - **Unit Tests** — контейнер для модульных тестов.
    - **Lib** — контейнер для двоичных зависимостей.

- **Docs** — контейнер для документации, поставляющейся с проектом.
- **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
- **Tests** — контейнер с результатами тестов, проводившихся тестовой командой.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию об изменении рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Отложенные правки

- Используйте отложенные правки для предоставления общего доступа к незавершенным изменениям.
- Используйте отложенные правки для архивации незавершенных изменений на сервере.
- Используйте отложенные правки, если вас прервали для выполнения более важного задания.

Используйте отложенные правки для предоставления общего доступа к незавершенным изменениям

Если вы хотите обсудить незавершенный код с удаленным членом команды, используйте отложенные правки. Вместо того чтобы отправлять код другому члену команды по электронной почте, вы можете создать правку кода на сервере, а затем предложить коллеге извлечь эти файлы. Также можно использовать отложенную правку, если вы хотите переслать недоработанный код другому разработчику для завершения.

Используйте отложенные правки для архивации незавершенных изменений на сервере

Вы можете отложить правки, если не завершили работу к концу рабочего дня и хотите с гарантией сохранить работу на сервере. Отложив текущие изменения, вы переносите их на сервер TFS, откуда сможете их извлечь их на следующий день (или передать другому члену команды).

Используйте отложенные правки, если вас прервали для выполнения более важного задания

Используйте отложенные правки, если в процессе внесения изменений в исходный код вы получили новое более срочное задание (например, исправление ошибки). Для этого вам придется вернуться к стабильной версии кода, но вы не хотите терять свои изменения. Отложите код, чтобы позднее снова извлечь его.

Дополнительные ресурсы

- Дополнительную информацию об отложенных правках вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Дополнительные ресурсы по управлению исходным кодом

- Дополнительную информацию о системе управления исходным кодом Team Foundation Server вы найдете в статье «Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181237(VS.80).aspx*.

Практические рекомендации

В этой части

- Практические рекомендации Team Build.
- Практические рекомендации: управление проектом.
- Практические рекомендации: система управления исходным кодом.
- Практические рекомендации: работа с отчетами.

Практические рекомендации: Team Build

В этом разделе

Администрирование

- Как защитить сервер сборки.
- Как удалить сборку.
- Как удалить тип сборки.
- Как связать со сборкой рабочий элемент.

Политики возврата после правки

- Как с помощью политик возврата повысить качество возвращаемых изменений.
- Как связать рабочие элементы со сборкой при помощи политики возврата изменений.

Непрерывная интеграция

- Как автоматически запускать сборки непрерывной интеграции.
- Как определить, нужна ли скользящая сборка.
- Как определить интервал выполнения скользящей сборки.

Настройка

- Как изменить номер сборки.
- Как настроить сопоставление рабочей области для извлечения и сборки части дерева исходного кода.
- Как выполнять сборку проекта, зависящего от другого командного проекта.
- Как изменить конфигурацию сборки (release/debug).

Развертывание

- Как установить сервер сборки.
- Как определить, есть ли необходимость в нескольких серверах сборки.

Общие вопросы

- Как с помощью Team Build выполнять сборку и развертывание приложений ASP.NET.
- Как с помощью Team Build выполнять сборку приложений на основе Microsoft® .NET 1.1.
- Как с помощью Team Build выполнять сборку проектов пакетов установки и развертывания.
- Как создать тип сборки.
- Как создать несколько типов сборки.
- Как создать сценарий сборки для проекта, в котором используются сборки из другого проекта.
- Как подписаться на получение уведомлений о сборке по электронной почте.
- Как получать уведомления о сбое при выполнении сборки.
- Как запустить сборку.
- Как убедиться, что сборка была выполнена успешно.
- Как просмотреть результат сборки.
- Как изменить расположение сервера сборки.
- Как изменить место накопления результатов сборки.
- Как определить, какие наборы изменений являются частью сборки.
- Как изменить заявленное качество сборки.

Проекты

- Как применять стратегию с одиночным решением.
- Как применять стратегию с разделенным решением.
- Как применять стратегию с несколькими решениями.

Отчеты

- Как просмотреть информацию о качестве сборки.
- Как просмотреть возвраты изменений для конкретной сборки.
- Как просмотреть закрытые рабочие элементы или ошибки, связанные со сборкой.
- Как просмотреть открытые рабочие элементы или ошибки, связанные со сборкой.

- Как отследить изменение скорости выполнения проекта от сборки к сборке.
- Как отслеживать пройденные и непройденные тесты для сборки.
- Как просмотреть состояние сборки.

Плановые сборки

- Как настроить автоматический запуск сборки по ночам.
- Как выбрать для проекта частоту выполнения сборки и ее тип.

Разработка через тестирование

- Как создать простейший приемочный тест.
- Как выполнять автоматизированное тестирование в ходе сборки.
- Как выполнять анализ кода в ходе сборки.
- Как реализовать сбой сборки при непрохождении тестов.

Администрирование

- Как защитить сервер сборки.
- Как удалить сборку.
- Как удалить тип сборки.
- Как связать со сборкой рабочий элемент.

Как защитить сервер сборки

Обеспечение безопасности сервера сборки

1. Выделите под службы сборки отдельный сервер, не развертывайте их на одном сервере с уровнем приложений или уровнем данных Microsoft Visual Studio® 2005 Team Foundation Server (TFS).
2. Предоставьте процессу сборки доступ к папке сборок с правом на чтение и запись. Убедитесь, что сборка выполняется от имени учетной записи с правом доступа к этой папке.
3. Предоставьте процессу сборки доступ к общему сетевому ресурсу для накопления результатов сборки с правом на чтение и запись. Убедитесь, что сборка выполняется от имени учетной записи с правом доступа к этому ресурсу.
4. Убедитесь, что учетная запись, используемая для выполнения сборки, входит в группу **Build Services** командного проекта.

Обеспечить более высокую безопасность Team Foundation Server позволит установка сервера сборки на специальном компьютере, отдельно от уровня приложений или уровня данных. Для выполнения определенных этапов развертывания или сборки могут потребоваться дополнительные бо-

лее широкие права доступа. Например, чтобы создать виртуальную папку для развертывания веб-приложения на сервере сборки, необходимо обладать правами администратора. То есть, учетная запись Microsoft Windows®, от имени которой выполняется сборка, должна обладать такими полномочиями. Если компьютер, на котором выполняется сборка, используется еще и для уровня приложений, это может представлять угрозу безопасности. Аналогично, если на сервере сборки размещается также и уровень данных, учетная запись, используемая для сборки, имеет доступ к базам данных этого уровня и может изменять их.

> **Примечание** Из соображений безопасности нельзя добавлять учетную запись, от имени которой выполняется сборка, в группу SERVER\ Service Accounts. Члены этой группы обладают на TFS полными административными правами.

Дополнительные ресурсы

- Подробную информацию о группах TFS и правах доступа вы найдете в статье «Team Foundation Server Default Groups, Permissions, and Roles» по адресу *http://msdn2.microsoft.com/en-us/library/ms253077(VS.80).aspx*.

Как удалить сборку

Для удаления сборок используется инструмент командной строки TFSBuild. Задайте адрес сервера TFS, имя командного проекта и имя сборки, например:

```
TfsBuild delete http://mytfsserver:8080 myproject build20070606.4
```

> **Примечание** В TFS 2008 сборки можно удалять из Visual Studio. В Build Explorer выберите сборку в списке завершенных сборок, щелкните ее правой кнопкой мыши и в контекстном меню выберите **Delete**.

Дополнительные ресурсы

- Подробнее об удалении завершенных сборок читайте в статье «How to: Delete a Completed Build» по адресу *http://msdn2.microsoft.com/en-us/library/aa337656(VS.80).aspx*.
- Дополнительную информацию о команде delete вы найдете в статье «Delete Command (Team Foundation Build)» по адресу *http://msdn2.microsoft.com/en-us/library/ms244360(VS.80).aspx*.

Как удалить тип сборки

Типы сборки нельзя удалить, используя Team Explorer. Это можно сделать только из системы управления исходным кодом.

Удаление типа сборки

1. Откройте Source Control Explorer.
2. Разверните папку своего командного проекта.
3. Разверните папку TeamBuildTypes.
4. Щелкните правой кнопкой мыши папку Team Build, представляющую тип сборки, который необходимо удалить, и выберите **Delete**.
5. Еще раз щелкните правой кнопкой мыши папку Team Build и выберите **Check In Pending Changes**.
6. Откройте Team Explorer.
7. Щелкните правой кнопкой мыши папку Team Builds и выберите **Refresh**.
8. Разверните папку Team Builds и убедитесь, что тип сборки удален.

> **Примечание** Чтобы удалить описание сборки в TFS 2008, щелкните правой кнопкой описание сборки в узле Builds обозревателя Team Explorer и выберите **Delete**. Файлы tfsbuild.proj и tfsbuild.rsp необходимо удалить из системы управления исходным кодом отдельно.

Дополнительные ресурсы

- Более подробно о Team Build читайте в статье «Overview of Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms181710(vs.80).aspx*.

Как связать со сборкой рабочий элемент

Диалоговое окно **Check In** позволяет связать рабочие элементы с возвратом после правки. Эти же рабочие элементы будут автоматически связаны со следующей сборкой.

Связывание рабочего элемента со сборкой

1. Внесите в код изменения, которые хотите включить в сборку и связать с рабочим элементом.
2. Верните ожидающие изменения в систему управления исходным кодом.
3. В диалоговом окне **Check In** щелкните **Work Items**.
4. Выберите рабочие элементы, которые хотите связать с этим возвратом.

Все изменения, внесенные с момента последней успешной сборки, будут связаны со следующей сборкой. После очередной сборки Team Build внесет этот набор изменений в список наборов, связанных со сборкой, и свяжет выбранный рабочий элемент с этим набором изменений.

Дополнительные ресурсы

- Подробнее о возврате ожидающих изменений читайте в статье «How to: Check In Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181411(VS.80).aspx*.

Политики возврата после правки

- Как с помощью политик возврата повысить качество возвращаемых изменений.
- Как связать рабочие элементы со сборкой при помощи политики возврата изменений.

Как с помощью политик возврата повысить качество возвращаемых изменений

Для повышения качества возвращаемых изменений следует использовать анализ кода в сочетании с политиками тестирования. Например, прежде чем разрешить возврат кода в систему управления версиями, при помощи готовой политики тестирования убедитесь, что он проходит определенные тесты. Также можно настроить политику анализа кода, которая обеспечит его соответствие определенным стандартам качества и правилам безопасности, производительности, переносимости, удобства обслуживания и надежности.

Включение политики анализа кода при возврате изменений

1. В Team Explorer щелкните правой кнопкой мыши свой командный проект, выберите **Team Project Settings** и щелкните **Source Control**.
2. Перейдите на вкладку **Check-in Policy**.
3. Щелкните **Add**. Затем выберите и настройте политики анализа кода и тестирования.

Дополнительные ресурсы

- Дополнительную информацию о создании и применении пользовательских политик возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Чтобы узнать о настройке политики возврата после правки, читайте статью «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- Чтобы посмотреть пример кода, который запрещает возврат правок с определенными вариантами кодирования, читайте статью «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.

- Чтобы посмотреть пример кода, который обязывает вводить комментарии при возврате после правки, читайте статью «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- Чтобы узнать, как зарегистрировать новую политику возврата после правки, читайте статью «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Как связать рабочие элементы со сборкой при помощи политики возврата изменений

Политики возврата изменений позволяют обеспечить связывание каждого возврата изменений с рабочим элементом. Чтобы связать рабочие элементы с возвратом изменений, разработчики используют диалоговое окно **Check In**. Эти рабочие элементы будут также автоматически связаны со следующей сборкой.

Включение политики связи возврата изменений с рабочими элементами

1. В Team Explorer щелкните правой кнопкой мыши свой командный проект, выберите **Team Project Settings** и щелкните **Source Control**.
2. Перейдите на вкладку **Check-in Policy**.
3. Щелкните **Add**. Выберите и настройте политику возврата **Work Item**.

Дополнительные ресурсы

- Дополнительную информацию о создании и применении пользовательских политик возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Чтобы узнать о настройке политики возврата после правки, читайте статью «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.

Непрерывная интеграция

- Как автоматически запускать сборки непрерывной интеграции.
- Как определить, нужна ли скользящая сборка.
- Как определить интервал выполнения скользящей сборки.

Как автоматически запускать сборки непрерывной интеграции

Хотя Team Foundation Server 2005 не содержит встроенного средства для непрерывной интеграции, в нем есть все необходимое для реализации собственного решения непрерывной сборки.

Создание решения непрерывной интеграции

1. **Создайте и протестируйте сборку** Убедитесь, что у вас есть нужный тип сборки и что он выполняется без ошибок.
2. **Установите надстройку для непрерывной интеграции** Его можно найти по адресу *http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx*.
3. **Сконфигурируйте надстройку для непрерывной интеграции** Убедитесь, что виртуальная корневая папка веб-приложения непрерывной интеграции находится в пуле приложений **TFSAppPool**. Обновите файл Web.config веб-приложения непрерывной интеграции, чтобы оно работало с вашим сервером и сборкой, добавив следующий параметр:

   ```
   <add key="1" value="TeamServer=http://TFSRTM:8080;TeamProjectName=Advent
   ureWorks;BuildType=Test Build"/>
   ```

4. **Подпишитесь на событие CheckinEvent** Чтобы подписаться на событие возврата изменений, используйте инструмент **BisSubscribe.exe** и следующую командную строку:

   ```
   Bissubscribe /eventType CheckinEvent /address http://TFSRTM:8080/ci/no-
   tify.asmx /deliveryType Soap /domain http://TFSRTM:8080
   ```

5. **Протестируйте непрерывную интеграцию** Протестируйте сборку по завершении возврата. Подождите несколько минут, чтобы сборка завершилась, и затем проверьте на странице сборок, успешно ли она была завершена.
6. **Настройте рассылку уведомлений по электронной почте** Они проинформируют заинтересованные стороны о завершении сборки. При помощи контекстного меню командного проекта откройте диалоговое окно **Project Alerts** и установите флажок уведомления **A build completes**.

Подробнее — в разделе «Как настроить непрерывную интеграцию в Visual Studio Team Foundation Server» этой книги.

Примечание Пользователи TFS 2008 могут настраивать процесс непрерывной интеграции из Visual Studio. Для редактирования типа сборки необходимо щелкнуть правой кнопкой мыши описание сборки в узле Builds обозревателя Team Explorer, выбрать **Edit Build Definition**, щелкнуть **Trigger** и задать активацию сборок по событию возврата после правки.

Дополнительные ресурсы

- Более подробную информацию вы найдете в главе 8 этой книги.
- Подробнее о настройке непрерывной интеграции рассказывается в разделе «Как настроить непрерывную интеграцию в Visual Studio Team Foundation Server» этой книги.

- Дополнительную информацию об использовании непрерывной интеграции в Visual Studio Team System вы найдете в статье «Continuous Integration Using Team Foundation Build» по адресу http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx.
- Скачать MSI-пакет установки решения непрерывной интеграции Visual Studio Team System можно по адресу *http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi*.
- Подробнее о гибкой разработке и непрерывной интеграции в TFS читайте в статье «Extend Team Foundation Server To Enable Continuous Integration» по адресу *http://msdn.microsoft.com/msdnmag/issues/06/03/TeamSystem/default.aspx*.

Как определить, нужна ли скользящая сборка

Выполнение сборки после каждого возврата — самая простая стратегия непрерывной интеграции, которая обычно обеспечивает самую быструю реакцию. Однако, если возврат после правки производится достаточно часто, она приводит к перегрузке сервера сборок. В этом случае вам следует использовать другой подход, при котором сборка производится после определенного числа возвратов после правки или по истечении определенного времени. Чтобы решить, нужно ли вам использовать скользящую сборку, выясните следующее:

- продолжительность сборки Team Build в минутах;
- средний интервал между последовательными возвратами после правки в минутах;
- промежуток времени, в течение которого возвраты после правки производятся наиболее часто.

Если длительность сборки больше среднего интервала между возвратами после правки, сборки будут выполняться непрерывно: одна сборка еще не будет завершена, когда уже произойдет следующий возврат после правки, инициирующий следующую сборку. Если возвраты после правки производятся до завершения предшествующей сборки, это негативно отразится на производительности сервера сборок и заблокирует запуск других сборок (например, сборок по расписанию). Определите промежуток времени, в течение которого возвраты после правки производятся особенно часто, и выясните, будет ли непрерывная интеграция оказывать влияние на сборки по расписанию и другие важные командные сборки.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 8 этой книги.

Как определить интервал выполнения скользящей сборки

Важно определить интервал скользящей сборки, чтобы обеспечить эффективность процесса. Если интервал между сборками превышает время, за которое выполняется одна сборка, в промежутках между скользящими сборками сервер будет доступен для сборок других типов.

Чтобы определить идеальный интервал времени скользящей сборки, разделите среднюю частоту возвратов после правки на длительность сборки. Например, если сборка занимает 10 минут, а возвраты после правки происходят в среднем раз в 5 минут, можно задать выполнение сборки после двух возвратов после правки, а для времени ожидания выбрать значение 10 минут. Это гарантирует завершение одной сборки до начала следующей. Если нагрузка на сервер сборки возросла, увеличьте эти значения.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 8 этой книги.

Настройка

- Как изменить номер сборки.
- Как настроить сопоставление рабочей области для извлечения и сборки части дерева исходного кода.
- Как выполнять сборку проекта, зависящего от другого командного проекта.
- Как изменить конфигурацию сборки (release/debug).

Как изменить номер сборки

Чтобы изменить номер сборки в скомпилированных файлах сборки, необходимо сгенерировать новый номер сборки и записать его в исходный файл assemblyinfo.

Чтобы номер сборки правильно отображался в интерфейсе Team Build, необходимо переопределить свойство $(**BuildNumber**) цели **BuildNumberOverride**.

Настройка номера сборки в файле сборке и в интерфейсе Team Build

1. Переопределите $(BuildNumber) в цели BuildNumberOverride.
2. Переопределите цель BeforeCompile для записи файла AssemblyInfo.cs или .vb.

Пример

```
<Target Name="BuildNumberOverrideTarget">
   <Message Importance="High" Text="$(BuildNumber)" />
   <ConvertTFSBuildNumberToSolutionBuildNumber
    MajorAndMinorVersion="1.0"
    TFSBuildNumber="$(BuildNumber)"
    TFSLastBuildNumber="$(LastBuildNumber)">
      <Output TaskParameter="SolutionBuildNumber" PropertyName="SolutionBui
ldNumber" />
      <Output TaskParameter="TFSBuildNumber" PropertyName="BuildNumber" />
   </ConvertTFSBuildNumberToSolutionBuildNumber>
   <Message Importance="High" Text="$(SolutionBuildNumber)" />
   <Message Importance="High" Text="$(BuildNumber)" />
</Target>

<Target Name="BeforeCompile">
   <Message Importance="High" Text="$(SolutionBuildNumber)" />
   <CreateItem Include="$(SolutionRoot)\**\AssemblyInfo.cs">
      <Output TaskParameter="Include" ItemName="AssemblyInfoFiles"/>
   </CreateItem>
   <CreateItem Include="$(SolutionRoot)\**\AssemblyInfo.vb">
      <Output TaskParameter="Include" ItemName="AssemblyInfoFiles"/>
   </CreateItem>
   <RewriteFileVersions
    AssemblyInfoFiles="@(AssemblyInfoFiles)"
    AssemblyVersionNumber="$(SolutionBuildNumber)"
    AssemblyFileVersionNumber="$(SolutionBuildNumber)"
    AssemblyInformationalVersionNumber="$(SolutionBuildNumber)" />
  </Target>
```

Дополнительные ресурсы

- Еще один метод изменения версии сборки предлагается в блоге Аарона Холберга (Aaron Halberg) «Team Build and the AssemblyInfo Task» по адресу *http://blogs.msdn.com/aaronhallberg/archive/2007/06/08/team-build-and-the-assemblyinfo-task.aspx*.
- Задаче AssemblyInfo, которая упоминается в указанном выше сообщении блога, посвящена новая страница на GotDotNet по адресу *http://www.gotdotnet.com/Community/UserSamples/Details.aspx?SampleGuid=5C455590-332C-433B-A648-E368B9515580*.
- Подробнее о настройке Team Foundation Build читайте в статье «Customizing Team Foundation Build» *по адресу http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx*.

Как настроить сопоставление рабочей области для извлечения и сборки части дерева исходного кода

В файле сопоставления рабочей области определены папки системы управления исходным кодом, извлекаемые сервером сборки. Для выполнения сборки не всегда требуется извлекать из системы все файлы. Чтобы ограничить количество включаемых папок, измените описание рабочей области или скройте ненужные файлы, чтобы они не извлекались в ходе сборки.

Например, по умолчанию новый проект сопоставляется с папкой $/TeamProject. Если все файлы исходных кодов находятся в папке $/TeamProject/foo/bar/foobar/sources, вам следует сопоставлять только эту папку.

Сокрытие папки

1. Извлеките для правки файл WorkspaceMapping.xml.
2. Добавьте в файл записи о сокрытии.
3. Верните файл WorkspaceMapping.xml в систему управления исходным кодом.

   Файл WorkspaceMapping.xml file выглядит следующим образом:

```
<Mappings>
   <InternalMapping ServerItem="$/MyTeamProject" LocalItem="c:\projects\
teamproject" Type="Map" />
   <InternalMapping ServerItem="$/MyTeamProject/documentation" Type="Cloak"
/>
</Mappings>
```

> **Примечание** Пользователи TFS 2008 могут настраивать сопоставление рабочей области прямо из Visual Studio. Для редактирования типа сборки необходимо щелкнуть правой кнопкой описание сборки в узле Builds обозревателя Team Explorer, выбрать Edit Build Definition, щелкнуть Workspace и редактировать сопоставления рабочих областей. Сопоставления рабочих областей в WorkspaceMapping.xml более не хранятся.

Дополнительные ресурсы

- Дополнительную информацию о сокрытии папок в рабочей области вы найдете в статье «How to: Cloak and Decloak Folders in a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181378(VS.80).aspx*.
- Дополнительную информацию об исключении папок при командной сборке вы найдете в статье «How to make 'Team Build' skip getting certain folders?» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/10/13/480584.aspx*.
- Дополнительную информацию о схеме WorkspaceMapping вы найдете в статье «Schema for the WorkspaceMapping.xml file» по адресу *http://blogs.*

msdn.com/buckh/archive/2007/02/28/schema-for-the-workspacemapping-xml-file.aspx.

Как выполнять сборку проекта, зависящего от другого командного проекта

Team Build не поддерживает сборку решений, объединяющих несколько командных проектов. Осуществить сборку таких решений можно, настроив файл TFSBuild.proj. Тогда при сборке необходимый код других проектов будет извлекаться непосредственно из системы управления исходным кодом.

Чтобы выполнить сборку проекта, имеющего зависимости от другого командного проекта, необходимо поместить исходный код или сборки этого проекта в рабочую область на вашем сервере сборки, отредактировав файл TFSBuild.proj. Добавьте в него сборку или ссылку на решение и переопределите событие **BeforeGet** для извлечения сборок или исходных файлов из всех командных проектов, от которых зависит ваш проект.

Сборка проекта, зависящего от другого командного проекта

1. В Source Control Explorer извлеките для редактирования сценарий TFSBuild.proj.
2. В разделе **PropertyGroup** (группа свойств) добавьте следующие параметры:

```
<!-- должны находиться под PropertyGroup -->
<TfCommand>$(TeamBuildRefPath)\..\tf.exe</TfCommand>
<SkipInitializeWorkspace>true</SkipInitializeWorkspace>
```

   Параметр SkipInitializeWorkSpace позволяет пропустить вызов задач по умолчанию для удаления и воссоздания рабочей области на компьютере сборки. Это новое свойство используется в специальной цели **BeforeGet** (см. ниже).
3. Добавьте следующие параметры в элементы **ItemGroup**, отвечающие за сопоставление командных проектов и решений. Убедитесь в правильности задания локальных путей для компьютера сборки. Одна локальная папка не может использоваться в нескольких сопоставлениях. Это приведет к возникновению исключения **MappingConflictException** при выполнении задачи CreateWorkspace.

```
<ItemGroup>
   <!-- для каждого используемого решения добавляется по одной записи -->
   <SolutionToBuild Include="$(SolutionRoot)\DependentApp\DependentApp.sln"
/>
   <SolutionToBuild Include="$(SolutionRoot)\YourApp\YourApp.sln" />
</ItemGroup>
```

```
<ItemGroup>
   <!-- для каждого используемого Team Project добавляется по одной записи -->
   <Map Include="$/YourApp/YourApp">
      <LocalPath>$(SolutionRoot)\YourApp</LocalPath>
   </Map>
   <Map Include="$/DependentApp/DependentApp">
      <LocalPath>$(SolutionRoot)\DependentApp</LocalPath>
   </Map>
</ItemGroup>
```

4. Переопределите событие **BeforeGet**, чтобы рабочие области извлекались для каждого командного проекта:

```
<Target Name="BeforeGet">
   <DeleteWorkspaceTask
    TeamFoundationServerUrl="$(TeamFoundationServerUrl)"
    Name="$(WorkspaceName)" />
   <Exec
    WorkingDirectory="$(SolutionRoot)"
    Command="&quot;$(TfCommand)&quot; workspace /new $(WorkSpaceName) /
server:$(TeamFoundationServerUrl)"/>
   <Exec
    WorkingDirectory="$(SolutionRoot)"
    Command="&quot;$(TfCommand)&quot; workfold /unmap /workspace:$(WorkS
paceName) &quot;$(SolutionRoot)&quot;"/>
   <Exec
    WorkingDirectory="$(SolutionRoot)"
    Command="&quot;$(TfCommand)&quot; workfold /map /workspace:$(WorkSp
aceName) /server:$(TeamFoundationServerUrl) &quot;%(Map.Identity)&quot;
&quot;%(Map.LocalPath)&quot;"/>
</Target>
```

5. Верните изменения сценария в систему управления исходным кодом и выполните сборку.

> **Примечание** В TFS 2008 для типов сборки нет ограничений на извлечение файлов из других командных проектов. Можно просто редактировать сопоставления рабочих областей при помощи диалогового окна Build Definition Editor и сопоставлять необходимые файлы.

Дополнительные ресурсы

- Более подробную информацию вы найдете найти в сообщении блога Маниша Агарвала (Manish Agarwal) «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Как изменить конфигурацию сборки (release/debug)

Чтобы изменить конфигурацию существующего сценария сборки, необходимо редактировать тег **<ConfigurationToBuild>** в файле TFSBuild.proj.

Изменение конфигурации сборки

1. Откройте Source Control Explorer.
2. Разверните папку своего командного проекта.
3. Разверните папку TeamBuildTypes.
4. Выберите папку сценария сборки, для которого хотите включить анализ кода.
5. Извлеките файл TFSBuild.proj из системы управления версиями. Возможно, сначала понадобится выполнить для папки операцию **Get Latest Version**.
6. Чтобы открыть TFSBuild.Proj, в окне Source Control Explorer щелкните его дважды.
7. Измените конфигурацию сборки в теге **<ConfigurationToBuild>**.
8. Сохраните TFSBuild.proj и верните его в систему управления версиями .

Дополнительные ресурсы

- Подробнее о настройке Team Foundation Build читайте в статье «Customizing Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx*.

Развертывание

- Как установить сервер сборки.
- Как определить, есть ли необходимость в нескольких серверах сборки.

Как установить сервер сборки

Сервер сборки устанавливается отдельно от TFS. Поскольку сервер сборки должен компилировать код, выполнять тесты и анализ кода, на нем должны быть установлены все инструменты, необходимые для этих действий.

Установка сервера сборки

1. Установите Visual Studio.
   - Чтобы гарантированно установить инструменты, необходимые для любого сценария сборки, полностью установите Team Suite.
   - Если вам требуется выполнять Team Build, но не нужны различные варианты тестирования, установите Visual Studio Team Developer Edition.

- Если предполагается выполнять автоматизированные тесты в составе процесса сборки, установите Visual Studio Team Test Edition.

2. В папке на DVD-диске Team Foundation Server откройте папку \build.
3. Запустите мастер установки.

Учетная запись, используемая для запуска сервера сборки, должна обладать следующими свойствами:

- должна обладать разрешением **Log On Locally** на компьютерах TFS;
- не должна быть локальной учетной записью администратора на компьютерах TFS;
- не может быть делегирована в домене Microsoft Active Directory®.

Дополнительные ресурсы

- Руководство по установке Team Foundation Server можно скачать из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=e54bf6ff-026b-43a4-ade4-a690388f310e&displaylang=en*.

Как определить, есть ли необходимость в нескольких серверах сборки

Выполнение на одном сервере несколько сценариев сборки может привести к чрезмерной нагрузке на сервер. В случае возникновения такой проблемы вам следует рассмотреть возможность выполнения разных типов сборки на разных серверах.

Процесс сборки может выполняться довольно длительное время, особенно если это сборка большого проекта. При использовании непрерывной интеграции или частых плановых сборок существует вероятность того, что сервер сборки не сможет справиться с их потоком. Для распределения нагрузки рекомендуется установить несколько серверов сборки. Для равномерного распределения нагрузки назначьте каждому из серверов разные типы сборки.

Общие вопросы

- Как с помощью Team Build выполнять сборку и развертывание приложений ASP.NET.
- Как с помощью Team Build выполнять сборку приложений на основе Microsoft® .NET 1.1.
- Как с помощью Team Build выполнять сборку проектов пакетов установки и развертывания.
- Как создать тип сборки.
- Как создать несколько типов сборки.
- Как создать сценарий сборки для проекта, в котором используются сборки из другого проекта.

- Как подписаться на получение уведомлений о сборке по электронной почте.
- Как получать уведомления о сбое при выполнении сборки.
- Как запустить сборку.
- Как убедиться, что сборка была выполнена успешно.
- Как просмотреть результат сборки.
- Как изменить расположение сервера сборки.
- Как изменить место накопления результатов сборки.
- Как определить, какие наборы изменений являются частью сборки.
- Как изменить заявленное качество сборки.

Как с помощью Team Build выполнять сборку и развертывание приложений ASP.NET

Если требуется выполнить сборку решения, содержащего только веб-приложения ASP.NET, важно правильно выбрать конфигурацию. Создавая тип сборки, убедитесь, что выбрана платформа **.NET** и конфигурация **Debug**:

Сборка решения, содержащего только проекты веб-приложений ASP.NET

1. Запустите New Team Build Type Creation Wizard.
2. Задайте имя типа сборки.
3. Выберите решение, содержащее только веб-приложение ASP.NET.
4. Выберите соответствующую конфигурацию.
5. В раскрывающемся списке **Platform** выберите платформу .NET.
6. Задайте расположение типа сборки.
7. Выберите тесты и правила анализа кода.
8. Сохраните тип сборки, щелкнув **Finish**.

При сборке решения, включающего как веб-приложения ASP.NET, так и другие .NET-проекты, необходимо выбрать тип платформы **Mixed**.

Сборка решения, содержащего веб-приложения ASP.NET и другие .NET-проекты

1. Запустите New Team Build Type Creation Wizard.
2. Задайте имя типа сборки.
3. Выберите решение, содержащее веб-приложения ASP.NET и другие проекты.
4. Выберите соответствующую конфигурацию.
5. В раскрывающемся списке **Platform** выберите вариант **Mixed**.
6. Задайте расположение типа сборки.

7. Выберите тесты и правила анализа кода.
8. Сохраните тип сборки, щелкнув **Finish**.

Местом накопления скомпилированных двоичных файлов будет папка {Тип сборки}\{Название конфигурации}\{Платформа}_PublishedWebsites.

Team Build не поддерживает развертывание веб-приложений на Internet Information Services (IIS). Существует два способа включить развертывание приложения на IIS в сценарий сборки: добавить в тип сборки специальный этап или использовать проект развертывания веб-приложений Web Deployment Project.

Если работа над командным проектом только начинается, рассмотрите вариант Web Deployment Project, чтобы выяснить, возможно ли его использование в вашей разработке. Если веб-сайт уже существует, использование проекта развертывания веб-приложений может нарушить процесс разработки. В этом случае необходимо рассмотреть использование специального этапа после сборки MSBuild. В обоих случаях учетная запись службы, от имени которой выполняется сборка, должна входить в локальную группу администраторов, что позволит ей создать виртуальную папку в IIS.

Послесборочный этап для развертывания веб-приложения

1. Скачайте библиотеку SDC Tasks Library с сайта Codeplex по адресу *http://www.codeplex.com/sdctasks*.
2. Извлеките из системы управления исходным кодом тип сборки, который собираетесь использовать для развертывания веб-приложения.
3. Из скачанного zip-архива извлеките файл Microsoft.Sdc.Tasks.dll и поместите его в ту же в папку, что и тип сборки.
4. Добавьте библиотеку DLL в систему управления исходным кодом и возвратите ее.
5. Внесите правки в файл TFSBuild.proj, чтобы файлы при сборке копировались в соответствующую папку, а затем сделайте эту папку виртуальной:
   a. Добавьте элемент **<PropertyGroup>**, определяющий расположение скомпилированных веб-приложений:

```
<PropertyGroup>
<WebBinariesLocation>$(SolutionRoot)\..\Binaries\.NET\Release\_
PublishedWebSites\MyWebSite</WebBinariesLocation>
</PropertyGroup>
```

   б. Добавьте два элемента **UsingTask** со ссылками на задачи CreateVirtualDirectory и DeleteVirtualDirectory:

```
<UsingTask TaskName="Microsoft.Sdc.Tasks.Web.WebSite.
CreateVirtualDirectory" AssemblyFile="Microsoft.Sdc.Tasks.dll" />
<UsingTask TaskName="Microsoft.Sdc.Tasks.Web.WebSite.
CreateVirtualDirectory" AssemblyFile="Microsoft.Sdc.Tasks.dll" />
```

в. Добавьте цель **AfterCompile** для создания виртуальной папки и копирования файлов в нее:

```
<Target Name="AfterCompile">
   <MakeDir Directories="C:\Deploy\MyWebsite" />
   <CreateVirtualDirectory VirtualDirectoryName="MyWebSite" Path="C:\
Deploy\Website" />
   <DeleteVirtualDirectory VirtualDirectoryName="MyWebSite" />
   <Exec Command="xcopy /y /e $(WebBinariesLocation) C:\Deploy\
MyWebsite"/>
</Target>
```

6. Сохраните файл и передайте его в хранилище системы управления исходным кодом.

Теперь при выполнении сборки будет создаваться веб-приложение и виртуальная папка, в которую это веб-приложение будет копироваться.

Проект развертывания веб-приложения

1. Скачайте и установите на клиентский компьютер Visual Studio 2005 Web Deployment Projects.
2. Скачайте и установите на сервере сборки Visual Studio 2005 Web Deployment Projects.
3. Откройте решение, содержащее веб-приложение.
4. В меню **Build** выберите команду **Add Web Deployment Project**.
5. Задайте имя и расположение проекта развертывания.
6. В Solution Explorer правой кнопкой мыши щелкните **Web Deployment Project** и выберите **Property Pages**.
7. В диалоговом окне задайте значение **Configuration** (Debug или Release), соответствующее тому, какую сборку должен выполнять Team Build.
8. В разделе **Deployment** установите флажок **Create an IIS virtual directory for the output folder** и задайте имя виртуальной папки.
9. Щелкните **OK**.
10. Верните изменения в систему управления исходным кодом.

При выполнении сборки, содержащей это решение, будет создано веб-приложение и виртуальная папка в папке, где выполняется сборка веб-приложения — {Папка сборки}\{Имя командного проекта}\{Тип сборки}\Binaries\{Название конфигурации}\{Платформа}_PublishedWebSite\{Имя проекта развертывания веб-приложения.

Дополнительные ресурсы

- Подробнее о сборке приложений ASP.NET с помощью Team Build читайте в статье «TN_1600: Building ASP.NET projects with Team Build» по адресу *http://msdn2.microsoft.com/en-us/teamsystem/aa718894.aspx*.

- Более подробную информацию об использовании проектов Web Deployment Project вы найдете в статье «TN_1601: Team Build and Web Deployment Projects» по адресу *http://msdn2.microsoft.com/en-us/teamsystem/aa718895.aspx*.
- Дополнительную информацию о сборке проектов Web Deployment Project вы найдете в статье «Visual Studio 2005 Web Deployment Projects» по адресу *http://msdn2.microsoft.com/en-us/asp.net/aa336619.aspx*.

Как с помощью Team Build выполнять сборку приложений на основе Microsoft® .NET 1.1

MSBuild не поддерживает .NET 1.1, поэтому такой поддержки нет и в Team Build. Компания Microsoft опубликовала на сайте CodePlex проект под названием MSBuild Extras (MSBee), который поддерживает сборку приложений на основе .NET 1.1.

Чтобы выполнять сборку приложений .NET 1.1, необходимо обновить файл проекта до .NET 2.0. Также придется отредактировать файл типа сценария сборки, чтобы при сборке использовались инструменты .NET 1.1, а не инструменты .NET 2.0.

Сборка приложений .NET 1.1 с использованием Team Build 2005

1. Обновите решения .NET 1.1 до .NET 2.0. Для этого откройте решение в Visual Studio 2005 и запустите Conversion Wizard или выполните команду devenv [имя_проекта] /upgrade.
2. Убедитесь, что на сервере сборки установлен .NET 1.1 Software Development Kit (SDK).
3. Скачайте и установите MSBuild Extras (*http://www.codeplex.com/MSBee*).
4. Скачайте BuildingFx11inTB.targets с сайта *http://blogs.msdn.com/gautamg/attachment/578915.ashx*.
5. Извлеките для редактирования из системы управления исходным кодом тип сборки, по которому будет выполняться сборка проекта .NET 1.1.
6. Скопируйте BuildingFx11inTB.targets в папку, содержащую тип сборки, и верните файл в систему управления исходным кодом.
7. Отредактируйте файл TFSBuild.proj:
   a. Импортируйте файл BuildingFx11inTB.targets:

```
<Import Project="$(MSBuildProjectDirectory)\BuildingFx11inTB.targets" />
```

   б. Добавьте свойство, определяющее цели CSharp:

```
<PropertyGroup>
<AdditionalPropertiesForBuildTarget>
   CustomAfterMicrosoftCommonTargets=$(ProgramFiles)\MSBuild\MSBee\
MSBuildExtras.Fx1_1.CSharp.targets
```

```
</AdditionalPropertiesForBuildTarget>
</PropertyGroup>
```

8. Верните файл TFSBuild.proj в систему управления исходным кодом.

> **Примечание** В TFS 2008 в файл TFSBuild.proj достаточно добавить в элемент SolutionToBuild следующий элемент Properties.

```
<SolutionToBuild Include="$(BuildProjectFolderPath)/path/MySolution.
sln">
   <Properties>
      CustomAfterMicrosoftCommonTargets=$(ProgramFiles)\MSBuild\MSBee\
MSBuildExtras.Fx1_1.CSharp.targets
   </Properties>
</SolutionToBuild>
```

Дополнительные ресурсы

- Подробнее о MSBuild Extras читайте в статье «MSBuild Extras – Toolkit for .NET 1.1 "MSBee"» по адресу *http://www.codeplex.com/MSBee*.
- Дополнительную информацию об использовании MSBuild Extras для сборки приложений .NET 1.1 вы найдете в статье «Building .NET 1.1 application using Team Build» по адресу *http://blogs.msdn.com/gautamg/archive/2006/04/19/578915.aspx*.

Как с помощью Team Build выполнять сборку проектов пакетов установки и развертывания

По умолчанию Team Build не поддерживает проекты пакетов установки приложений. Для компиляции такого проекта и копирования двоичных файлов в место накопления результатов сборки используйте специальный послесборочный этап.

Протестируйте сборку

Убедитесь в работоспособности типа сборки, который намерены использовать для создания пакета установки. Если он не работает, устраните ошибку, прежде чем двигаться дальше.

> **Совет** В большинство сборок включается как сборка пакета установки, так и сборка основного проекта. Если вы создаете новый сценарий сборки исключительно для проекта установки, это необходимо сделать перед переходом к следующему шагу.

Задайте выполнение сборки пакета установки

1. В Solution Explorer щелкните правой кнопкой мыши проект установки, для которого будет создаваться сценарий сборки.
2. Щелкните **Properties**.
3. Щелкните **Configuration Manager**.
4. Выберите необходимые конфигурации (Debug, Release, обе).
5. Установите флажок **Build** для проекта пакета установки.

Сделайте все пути сборки в файле проекта относительными

1. Откройте папку решения для проекта пакета установки.
2. Откройте файл .vdproj в любом другом редакторе (не в Visual Studio).
3. Извлеките файл .vdproj из системы управления исходным кодом для редактирования.
4. Найдите в нем следующие записи: **SccLocalPath** (локальный путь в системе управления исходным кодом), **SccAuxPath** (вспомогательный путь в системе управления исходным кодом), **OutputFileName** (имя выходного файла) и **SourcePath** (путь к исходному файлу).
5. Убедитесь, что все пути являются относительными, а не абсолютными (по умолчанию они должны быть относительными). Абсолютные пути начинаются с имени диска, а относительные пути — с двойной косой черты (\\) или вообще не имеют никаких дополнительных символов.
6. Все абсолютные пути (если таковые обнаружатся) замените относительными (не меняя константы, которые будут раскрыты программой установки позже), например:

   ```
   "OutputFileName" = "8:c:\\temp\\SetupProject.msi"
   ```

   замените на

   ```
   "OutputFileName" = "8:debug\\SetupProject.msi"
   ```

Совет Относительные пути разрешаются относительно папки проекта.

Совет При создании путей нужно использовать двойную косую черту ('\\'), потому что вместо нее потом подставляется одиночная косая черта ('\ ).

---

7. Если в файл .vdproj были внесены какие-либо изменения, верните его в систему управления исходным кодом.

Добавьте в сценарий сборки задачу, выполняющуюся после компиляции

1. Откройте сценарий сборки, который хотите использовать для проекта пакета установки.
2. Извлеките сценарий сборки из системы управления исходным кодом для редактирования.
   a. Файл типа сценария сборки TFSBuild.proj находится в системе управления исходным кодом в папке командного проекта в подпапке папки TeamBuildTypes.
   б. Возможно, сначала понадобится выполнить операцию **Get Latest Version**.
3. В окне Source Control Explorer щелкните дважды файл TFSBuild.Proj, чтобы открыть его для редактирования.

> **Примечание** Просмотр типа сборки из Team Explorer ничего не даст, поскольку копия файла будет открыта только для чтения.

4. Добавьте следующий код в конец файла между последним тегом **</ItemGroup>** и последним тегом **</Project>**:

```
<Target Name="AfterCompile">
   <Exec Command="&amp;quot;C:\Program Files\Microsoft Visual Studio
8\Common7\IDE\devenv&amp;quot; &amp;quot;C:\Documents and Settings\dar-
ren\My Documents\Visual Studio 2005\Projects\HelloWorldTest\
HelloWorldTestInstaller\HelloWorldTestInstaller.vdproj&amp;quot; /Build
&amp;quot;Debug&amp;quot;"/>
   <Copy SourceFiles="C:\Documents and Settings\darren\My Documents\
Visual Studio 2005\Projects\HelloWorldTest\HelloWorldTestInstaller\
Debug\HelloWorldTestInstaller.msi" DestinationFolder="$(OutDir)" />
   <Copy SourceFiles="C:\Documents and Settings\darren\My Documents\
Visual Studio 2005\Projects\HelloWorldTest\HelloWorldTestInstaller\
Debug\setup.exe" DestinationFolder="$(OutDir)" />
</Target>
```

5. При необходимости исправьте пути во вставленном фрагменте кода в соответствии со своим сервером сборки.

> **Совет** Проверьте путь из тега <exec command> в командной строке, заменив " кавычками, например:

```
"C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\devenv"
   "C:\Documents and Settings\darren\My Documents\Visual Studio 2005
   \Projects\HelloWorldTest\HelloWorldTestInstaller\
HelloWorldTestInstaller.vdproj" /Build "Debug"
```

Совет Используйте командную строку и для проверки путей SourceFiles.

Протестируйте изменения, внесенные в сборку

1. В Team Explorer правой кнопкой мыши щелкните тип сборки и выберите команду **Build Team Project**.
2. Посмотрите в отчете о результатах сборки, была ли она успешной или дала сбой.
3. Если сборка дала сбой, щелкните ссылку на протокол сборки. Как правило, причины сбоя таковы:
   а. Неверный путь команды exec.
   б. Неверно заданы права доступа, из-за чего не удалось скопировать выходные файлы в папку сборки. Убедитесь, что у учетной записи tfsservice есть право копировать файлы из папки двоичных файлов в папку сборки. Папка двоичных файлов — это папка, в которую помещается файл msi после сборки. Папка сборки определена в файле в теге <BuildDirectoryPath>.
   в. Неверно заданы права доступа, что препятствует сборке проекта пакета установки. Убедитесь, что учетная запись tfsserver обладает правом чтения файла .vdproj и исходных файлов проекта, а также приложения, с которым связан проект установки. Кроме того, учетной записи tfsserver необходимо право записи двоичных файлов в выходную папку, например, Debug или Release.
   г. Неверная конфигурация сборки. Убедитесь, что заданная в теге exec command конфигурация сборки существует. Например, вместо конфигурации «Debug» в проекте может иметься конфигурация «Debug|Any CPU». Это можно проверить, посмотрев свойства решения в Solution Explorer.
4. Если в протоколе сборки нет достаточной информации, создайте выходной файл для команды exec и попробуйте найти в нем недостающие сведения. Например:

```
<Exec Command="&quot;C:\Program Files\Microsoft Visual Studio 8\Common7\
IDE\devenv&quot; &quot;C:\Documents and Settings\darren\My Documents\
Visual Studio 2005\Projects\HelloWorldTest\HelloWorldTestInstaller\
HelloWorldTestInstaller.vdproj&quot; /Build &quot;Debug&quot; > c:\temp\
output.txt"/>
```

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Walkthrough: Configuring Team Foundation Build to Build a Visual Studio Setup Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms404859(VS.80).aspx*.

Как создать тип сборки

Новый сценарий сборки создается из папки Team Builds в Team Explorer.

Создание сценария сборки

1. Откройте Team Explorer.
2. Разверните командный проект, для которого хотите создать тип сборки.
3. Щелкните правой кнопкой мыши папку Team Builds.
4. Выберите команду **New Team Build Type**.
5. Задайте имя нового типа сборки и щелкните **Next**.
6. Выберите проекты, которые хотите включить в сборку. Среди них должен быть проект с модульными тестами.
7. Выберите конфигурацию сборки (например, Release или Debug) и щелкните **Next**.
8. Задайте имя сервера сборки, например, **TFSRTM**.
9. Задайте локальную папку сборки на сервере сборки, например, **c:\builds**.
10. Задайте место накопления результатов сборки, например, **TFSRTM****NightlyBuilds**, и щелкните **Next**.
11. Установите флажок **Run tests**, выберите список тестов, которые хотите связать со сборкой, и щелкните **Next**.
12. Щелкните **Finish**, чтобы создать новый тип сборки.

Как создать несколько типов сборки

Чтобы создать несколько типов сборки, например, Release для заказчиков и Debug для группы тестирования, необходимо использовать разные сценарии для каждого типа сборки. Каждый из них создается так, как описано в предыдущем разделе.

Как создать сценарий сборки для проекта, в котором используются сборки из другого проекта

Чтобы выполнить сборку проекта, имеющего зависимости от другого командного проекта, необходимо перенести исходный код или сборки из этого проекта в рабочую область на сервере сборки. Для этого понадобится отредактировать файл TFSBuild.proj, добавив в него сборку или ссылку на решение, и переопределить событие **BeforeGet** для получения всех необходимых сборок или исходных файлов из используемых командных проектов.

Сборка проекта, использующего сборки из другого командного проекта

1. В Source Control Explorer извлеките для редактирования сценарий TFSBuild.proj.

2. Под разделом **PropertyGroup** добавьте следующие параметры:

```
<!-- добавить под PropertyGroup -->
<TfCommand>$(TeamBuildRefPath)\..\tf.exe</TfCommand>
<SkipInitializeWorkspace>true</SkipInitializeWorkspace>
```

   Параметр SkipInitializeWorkSpace позволяет пропустить вызов задач по умолчанию для удаления и воссоздания рабочей области на компьютере сборки. Это новое свойство используется в специальной цели **BeforeGet** (см. ниже).

3. Добавьте следующие параметры в элементы **ItemGroup**, отвечающие за сопоставление как командных проектов, так и решений. Убедитесь в правильности задания локальных путей для компьютера сборки. Одна локальная папка не может использоваться в нескольких сопоставлениях. Это приведет к исключению **MappingConflictException** при выполнении задачи CreateWorkspace.

```
<ItemGroup>
   <!-- добавляется по одной записи для каждого используемого решения -->
   <SolutionToBuild Include="$(SolutionRoot)\DependentApp\DependentApp.
sln" />
   <SolutionToBuild Include="$(SolutionRoot)\YourApp\YourApp.sln" />
</ItemGroup>

<ItemGroup>
   <!-- добавляется по одной записи для каждого используемого проекта -->
   <Map Include="$/YourApp/YourApp">
      <LocalPath>$(SolutionRoot)\YourApp</LocalPath>
   </Map>
   <Map Include="$/DependentApp/DependentApp">
      <LocalPath>$(SolutionRoot)\DependentApp</LocalPath>
   </Map>
</ItemGroup>
```

4. Переопределите событие **BeforeGet**, чтобы рабочие области извлекались для каждого командного проекта:

```
<Target Name="BeforeGet">
   <DeleteWorkspaceTask
    TeamFoundationServerUrl="$(TeamFoundationServerUrl)"
    Name="$(WorkspaceName)" />
   <Exec
    WorkingDirectory="$(SolutionRoot)"
    Command="&amp;quot;$(TfCommand)&amp;quot; workspace /new $(WorkSpaceName) /
server:$(TeamFoundationServerUrl)"/>
   <Exec
    WorkingDirectory="$(SolutionRoot)"
    Command="&amp;quot;$(TfCommand)&amp;quot; workfold /unmap /workspace:$(WorkS
```

```
paceName) &quot;$(SolutionRoot)&quot;"/>
   <Exec
    WorkingDirectory="$(SolutionRoot)"
    Command="&quot;$(TfCommand)&quot; workfold /map /workspace:$(WorkSp
aceName) /server:$(TeamFoundationServerUrl) &quot;%(Map.Identity)&quot;
&quot;%(Map.LocalPath)&quot;"/>
</Target>
```

5. Верните изменения сценария в систему управления исходным кодом и выполните сборку.

> **Примечание** В TFS 2008 нет ограничений для сценариев сборки на извлечение файлов из других командных проектов. Можно просто редактировать сопоставления рабочих областей при помощи диалогового окна Build Definition Editor и сопоставлять необходимые файлы.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Как подписаться на получение уведомлений о сборке по электронной почте

Чтобы узнавать о завершении сборки по электронной почте, создайте уведомление проекта.

Создание уведомления о выполнении сборки

1. Щелкните правой кнопкой командный проект в Team Explorer.
2. Выберите команду **Project Alerts**.
3. Выберите все варианты уведомлений, которые хотите получать, и введите адрес электронной почты.

Дополнительные ресурсы

- Подробнее об уведомлениях проекта читайте в статье «Setting Alerts» по адресу *http://msdn2.microsoft.com/en-us/library/ms181334(VS.80).aspx*.

Как получать уведомления о сбое при выполнении сборки

Чтобы получать сообщения по электронной почте только при сбое сборки, создайте уведомление проекта с фильтрованием.

Создание уведомления о сбое сборки

1. Создайте и протестируйте сборку. Убедитесь, что у вас соответствующий тип сборки и что он выполняется без ошибок.
2. С помощью инструмента командной строки **BisSubscribe.exe** подпишитесь на событие **BuildCompletionEvent** и задайте фильтр, определяющий, что вы хотите получать уведомления по электронной почте только при сбое сборки. Используйте для этого следующий синтаксис:

```
bissubscribe /eventType BuildCompletionEvent /address myemail@domain.com
/deliveryType EmailPlaintext /server tfsserver1 /filter "TeamProject =
'MyTeamProject' AND CompletionStatus='Failed'"
```

3. Протестируйте сборку. Верните в систему код, который заведомо не компилируется, выполните сборку и убедитесь в получении уведомления по электронной почте.

Дополнительные ресурсы

- Дополнительную информацию о фильтрах событий завершения сборки вы найдете в статье «How to Filter the Build Completion Event» по адресу *http://blogs.msdn.com/jpricket/archive/2006/09/05/how-to-filter-the-build-completion-event.aspx*.
- Подробнее о фильтрах BuildCompletionEvent читайте в статье «Useful BuildCompletionEvent Filters» по адресу *http://blogs.msdn.com/jpricket/archive/2006/09/05/useful-buildcompletionevent-filters.aspx*.

Как запустить сборку

Запустить тип сборки можно из папки Team Builds в Team Explorer.

Ручной запуск сборки

1. Откройте Team Explorer.
2. Разверните командный проект, сборку которого хотите начать.
3. Разверните папку Team Builds.
4. Правой кнопкой мыши щелкните тип сборки, который хотите запустить.
5. Выберите команду **Build Team Project**.

Как убедиться, что сборка была выполнена успешно

Состояние сборки можно проверить в окне Builds, которое доступно в Team Explorer.

Проверка успешного выполнения сборки

1. Откройте Team Explorer.
2. Разверните проект, для которого хотите посмотреть результаты.

3. Разверните папку Team Builds.
4. Щелкните дважды тип сборки, для которого хотите просмотреть результаты.

Как просмотреть результат сборки

Результат сборки можно просмотреть в окне Builds, которое доступно в Team Explorer.

Просмотр результата сборки

1. Откройте Team Explorer.
2. Разверните командный проект, для которого хотите увидеть результат сборки.
3. Разверните папку Team Builds.
4. Щелкните дважды тип сборки, для которого хотите увидеть результат.
5. Щелкните дважды результирующий элемент Team Build, соответствующий номеру сборки, для которой хотите увидеть результат.
6. Чтобы просмотреть папку с результатом сборки, щелкните ссылку **Build name**.
7. Чтобы просмотреть протокол сборки, щелкните ссылку **Log**.

Как изменить расположение сервера сборки

Чтобы задать другой сервер сборки для существующего типа сборки, измените тег **<BuildMachine>** в файле TFSBuild.proj.

Изменение сервера сборки для существующего типа сборки

1. Откройте Source Control Explorer.
2. В Source Control Explorer разверните папку своего командного проекта.
3. Разверните папку TeamBuildTypes.
4. Выберите папку сценария сборки, для которого хотите изменить сервер сборки.
5. Извлеките файл TFSBuild.proj из системы управления исходным кодом. Возможно, сначала придется выполнить операцию **Get Latest Version**.
6. Чтобы открыть TFSBuild.Proj, щелкните его дважды в Source Control Explorer.
7. Измените тег **<BuildMachine>**, подставив в него указание на другой сервер.
8. Сохраните TFSBuild.proj и верните его в систему управления исходным кодом.

Примечание В TFS 2008 тег <**BuildMachine**> уже не используется. Вместо этого в описании сборки задается агент сборки. Редактирование агентов сборки выполняется при помощи параметра Manage Build Agents. Чтобы найти его, щелкните правой кнопкой узел Builds в Team Explorer Build. Агент сборки задается при запуске новой сборки.

Дополнительные ресурсы

- Подробнее о настройке Team Foundation Build читайте в статье «Customizing Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx*.

Как изменить место накопления результатов сборки

Чтобы изменить место накопления результатов сборки, отредактируйте тег **<DropLocation>** в файле TFSBuild.proj.

Изменение места накопления для существующего типа сборки

1. Откройте Source Control Explorer.
2. Разверните папку своего командного проекта.
3. Разверните папку TeamBuildTypes.
4. Выберите папку сценария сборки, для которого хотите изменить место накопления.
5. Извлеките файл TFSBuild.proj из системы управления исходным кодом. Возможно, сначала придется выполнить операцию **Get Latest Version**.
6. Чтобы открыть TFSBuild.Proj, щелкните его дважды в Source Control Explorer.
7. Измените тег **<DropLocation>**, чтобы он указывал на новое положение.
8. Сохраните TFSBuild.proj и верните его в систему управления исходным кодом.

Примечание В TFS 2008 тег <**DropLocation**> не используется. Чтобы задать место накопления результатов сборки, щелкните правой кнопкой мыши описание сборки в узле Builds дерева Team Explorer, выберите команду **Edit Build Definition**, щелкните **Build Defaults** и задайте место публикации.

Дополнительные ресурсы

- Подробнее о настройке Team Foundation Build читайте в статье «Customizing Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx*.

Как определить, какие наборы изменений являются частью сборки

Увидеть, какие наборы изменений связаны со сборкой, можно из окна Builds, которое доступно в Team Explorer.

Просмотр наборов изменений, связанных со сборкой

1. Откройте Team Explorer.
2. Разверните командный проект, для которого хотите просмотреть наборы изменений.
3. Разверните папку Team Builds.
4. Щелкните дважды тип сборки, для которого хотите просмотреть наборы изменений.
5. Щелкните дважды результирующий элемент Team Build, соответствующий номеру сборки, для которой вы хотите увидеть наборы изменений.
6. Разверните элемент **Associated changesets**.
7. Чтобы увидеть больше информации для конкретного набора изменений, щелкните ссылку **ID**.

Как изменить заявленное качество сборки

Качество сборки можно изменить из окна Builds, которое доступно в Team Explorer.

Изменение показателя качества сборки

1. Откройте Team Explorer.
2. Разверните командный проект, для которого хотите задать другие качество сборки.
3. Разверните папку Team Builds.
4. Щелкните дважды тип сборки, для которого хотите изменить качество сборки.
5. В раскрывающемся списке **Build Quality** выберите сборку, для которой хотите задать качество сборки, и задайте значение качества сборки.

Проекты

- Как применять стратегию с одиночным решением.
- Как применять стратегию с разделенным решением.
- Как применять стратегию с несколькими решениями.

Как применять стратегию с одиночным решением

Если вы работаете в небольшой группе, вам стоит использовать стратегию единого решения Visual Studio, содержащего все проекты. Эта структура упрощает разработку, поскольку при открытии решения сразу доступен весь код. При такой стратегии также легко устанавливать ссылки, поскольку все они связывают проекты одного решения. Вы вольны использовать и файловые ссылки, чтобы ссылаться на сборки сторонних производителей, например, приобретенные компоненты, которые находятся вне вашего решения.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 3 этой книги.

Как применять стратегию с разделенным решением

Если вы работаете в большой группе, вам выгодно использовать несколько решений, каждое из которых представляет определенную подсистему приложения. Разработчики будут применять эти решения для работы над мелкими частями системы без необходимости загрузки всего кода по всем проектам. Проектируйте структуру решения так, чтобы любые проекты, связанные зависимостями, группировались вместе. Это позволит вам использовать ссылки на проекты вместо ссылок на файлы. Подумайте о создании главного решения, которое содержало бы все проекты, для сборки всего приложения.

Работая с несколькими решениями, используйте во всех проектах плоскую структуру файлов. Типичный пример — приложение, включающее проект Microsoft Windows® Forms, проект ASP.NET, службу Windows и ряд библиотек классов, которые совместно используются некоторыми или всеми перечисленными проектами.

Для всех проектов может использоваться следующая плоская структура:

- /Source
  - /WinFormsProject
  - /WebProject
  - /WindowsServiceProject
  - /ClassLibrary1
  - /ClassLibrary2
  - /ClassLibrary3
  - Web.sln
  - Service.sln
  - All.sln

Плоская структура обеспечивает большую гибкость и возможность использовать решения для создания разных представлений проектов. Физическую структуру папок решения менять очень трудно, особенно если используется библиотека классов из другого решения.

Примечание Если вы осуществляете сборку при помощи Team Build (на основе MSBuild), можете создавать решения, не включающие в себя все связанные проекты. При условии что сначала вы сбираете решение целиком, генерируя двоичный файл для каждого решения, MSBuild сможет проследовать по ссылкам в проекты вне решения и произвести успешную сборку. Решения, создаваемые таким образом, не будут собираться при помощи команды сборки из Visual Studio. Они будут работать только с Team Build и MSBuild.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 3 этой книги.

Как применять стратегию с несколькими решениями

Работая над очень большим решением, для которого требуется несколько десятков проектов, вы рискуете столкнуться с ограничениями масштабируемости. Если это случилось, разбейте приложение на несколько решений, но не создавайте главное решение для всего приложения. Все ссылки внутри каждого решения будут ссылками на проекты. Ссылки на проекты вне решения (например, на библиотеки сторонних разработчиков в другом решении) будут ссылками на файлы. Это означает, что главного решения быть не может. Вместо этого нужно использовать сценарий, в котором учтен порядок сборки решений. Одной из задач обслуживания структуры из нескольких решений является контроль за тем, чтобы разработчики по невнимательности не создали кольцевых ссылок между решениями. Такая структура требует создания сложного сценария сборки и явного сопоставления отношений зависимости. Собрать приложение во всей его полноте из Visual Studio невозможно. Вместо этого вам придется использовать TFS Team Build или непосредственно MSBuild.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в главе 3 этой книги.

Отчеты

- Как просмотреть информацию о качестве сборки.
- Как просмотреть возвраты изменений для конкретной сборки.
- Как просмотреть закрытые рабочие элементы или ошибки, связанные со сборкой.
- Как просмотреть открытые рабочие элементы или ошибки, связанные со сборкой.
- Как отследить изменение скорости выполнения проекта от сборки к сборке.

- Как отслеживать пройденные и непройденные тесты для сборки.
- Как просмотреть состояние сборки.

Как просмотреть информацию о качестве сборки

Информацию о качестве сборки можно найти в окне Builds, которое доступно из Team Explorer.

Просмотр информации о качестве сборки

1. Откройте Team Explorer.
2. Разверните проект, для которого хотите посмотреть качество сборки.
3. Разверните папку Team Builds.
4. Щелкните дважды тип сборки, качество выполнения которой хотите оценить.

Если используется шаблон процесса MSF CMMI, информация о качестве сборки, а также дополнительные данные по результатам тестов, покрытию кода тестами и степени изменения кода, записываются в отчет Builds.

Просмотр информации о качестве сборки в MSF CMMI

1. В Team Explorer разверните узел нужного командного проекта.
2. Щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
3. На сайте отчетов выберите отчет **Builds**.

Как просмотреть возвраты изменений для конкретной сборки

Возвращенные изменения, связанные с каждой сборкой, можно увидеть в окне Builds, которое доступно из Team Explorer.

Просмотр возвратов изменений, связанных со сборкой

1. Откройте Team Explorer.
2. Разверните командный проект, для которого хотите увидеть возвраты изменений, связанные со сборкой.
3. Разверните папку Team Builds.
4. Щелкните дважды тип сборки, для которой хотите просмотреть возвраты изменений.
5. Щелкните дважды результирующий элемент Team Build, для которого хотите увидеть возвраты изменений.
6. Разверните элемент **Associated changesets**, чтобы просмотреть возвраты изменений, связанные со сборкой.
7. Чтобы получить больше информации по конкретному набору изменений, щелкните ссылку **ID**.

Как просмотреть закрытые рабочие элементы или ошибки, связанные со сборкой

Рабочие элементы и ошибки, которые были закрыты в конкретной сборке, можно просматривать в окне Builds, доступном в Team Explorer.

Просмотр закрытых рабочих элементов, связанных со сборкой

1. Откройте Team Explorer.
2. Разверните командный проект, для которого хотите просмотреть рабочие элементы.
3. Разверните папку Team Builds.
4. Щелкните дважды тип сборки, рабочие элементы которой хотите просмотреть.
5. Щелкните дважды результирующий элемент Team Build, соответствующий номеру сборки, для которой вы хотите просмотреть рабочие элементы.
6. Разверните элемент **Associated work items**.

Как просмотреть открытые рабочие элементы или ошибки, связанные со сборкой

Если используется шаблон процесса MSF CMMI, открытые, решенные и закрытые рабочие элементы за указанный период времени можно просмотреть в отчете Open Issues and Blocked Work Items Trend. Однако в отчете информация отсортирована по датам, а не по сборкам, поэтому вам необходимо будет сгруппировать результаты по сборкам вручную.

Просмотр открытых рабочих элементов или ошибок для конкретной сборки

1. В Team Explorer разверните узел своего командного проекта.
2. Щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
3. На сайте отчетов выберите отчет **Open Issues and Blocked Work Items Trend**.

Как отследить изменение скорости выполнения проекта от сборки к сборке

Отслеживать прогресс проекта и темпы выполнения рабочих элементов от сборки к сборке можно с помощью отчета Velocity. Этот отчет доступен как в шаблоне проекта MSF CMMI, так и в шаблоне MSF Agile.

Контроль за темпом выполнения проекта

1. В Team Explorer разверните узел своего командного проекта.
2. Щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
3. На сайте отчетов выберите отчет **Velocity**.

Как отслеживать пройденные и непройденные тесты для сборки

Отслеживать количество пройденных и непройденных сценариев тестирования за указанный период времени позволяет отчет Quality Indicators. Информация в нем отсортирована по дате, а не по сборке, т.е. вам необходимо будет сгруппировать результаты по сборкам вручную. Этот отчет доступен как в MSF CMMI, так и в MSF Agile.

Просмотр пройденных и непройденных тестов для сборки

1. В Team Explorer разверните узел своего командного проекта.
2. Щелкните правой кнопкой мыши **Reports** и затем щелкните **Show Report Site**.
3. На сайте отчетов выберите отчет **Quality Indicators**.

Как просмотреть состояние сборки

Если вы используете шаблон процесса MSF CMMI, результаты тестов верификации сборки (build verification testing, BVT) можно найти в отчете Builds.

Просмотр состояния сборки

1. В Team Explorer разверните узел своего командного проекта.
2. Щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
3. На сайте отчетов выберите отчет **Builds**.

Плановые сборки

- Как настроить автоматический запуск сборки по ночам.
- Как выбрать для проекта частоту выполнения сборки и ее тип.

Как настроить автоматический запуск сборки по ночам

Функция Team Build в TFS не поддерживает создание плановых сборок из пользовательского интерфейса. Поэтому для запуска сборок в заданное время используется планировщик задач Windows Task Scheduler и утилита командной строки TFSBuild.

Создание плановой сборки

1. Создайте строку запуска команды TFSBuild:

```
TfsBuild start <<TeamFoundationServer>> <<TeamProject>>
<<BuildTypeName>>
```

2. Поместите строку команды запуска в командный файл.
3. Создайте задачу планировщика, которая будет выполнять командный файл через заданные промежутки времени.

Подробнее — в разделе «Как настроить плановую сборку в Visual Studio Team Foundation Server» этой книги.

> **Примечание** Пользователи TFS 2008 могут планировать сборки прямо в Visual Studio. Для редактирования описания типа сценария сборки необходимо щелкнуть правой кнопкой описание сборки в узле Builds дерева Team Explorer, выбрать **Edit Build Definition**, щелкнуть **Trigger** и задать расписание сборки.

Дополнительные ресурсы

- Более подробную информацию вы найдете в главе 9 этой книги.

Как выбрать для проекта частоту выполнения сборки и ее тип

Выбор частоты выполнения сборок — одно из самых важных решений при создании плановой сборки. Если проект претерпевает существенные изменения в течение часа, и при этом непрерывная интеграция не используется, можно выполнять сборки каждый час. Это позволит разработчикам оперативно получить информацию о качестве кода, в частности, от тестировщиков и членов других групп.

Если проект претерпевает существенные изменения на протяжении дня, можно использовать ежедневную плановую сборку. Группы тестирования и разработки будут каждое утро получать готовую к тестированию сборку, которая включает в себя все изменения, внесенные за предыдущий рабочий день.

В больших сложных проектах процесс сборки может занимать несколько дней. В этом случае следует остановиться на еженедельных сборках. Готовую к тестированию сборку, включающую все внесенные на предыдущей неделе изменения, группа разработки будет получать в начале каждой недели.

Дополнительные ресурсы

- Подробнее о настройке плановых сборок читайте в разделе «Как настроить плановую сборку в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию вы найдете в главе 9 этой книги.

Разработка через тестирование

- Как создать простейший приемочный тест.
- Как выполнять автоматизированное тестирование в ходе сборки.

- Как выполнять анализ кода в ходе сборки.
- Как реализовать сбой сборки при непрохождении тестов.

Как создать простейший приемочный тест

Простейший приемочный тест поможет вам убедиться в собственном умении создавать модульные тесты и связывать их с процессом сборки.

Чтобы создать список тестов, связанных со сборкой, необходимо установить Visual Studio Test Edition или Visual Studio Team Suite. Автоматизированные тесты можно выполнять на сервере сборки, только если на нем установлены Visual Studio Developer Edition, Visual Studio Test Edition или Visual Studio Team Suite.

Создание приемочного теста

1. В меню **Test** выберите команду **New Test**.
2. Щелкните **Unit Test** и **OK**.
3. Введите имя проекта и щелкните **Create**.
4. Откомпилируйте новый проект.
5. Верните проект в систему управления исходным кодом.
6. В Visual Studio, открыв решение модульного теста, в меню **Test** выберите команду **Create New Test List**.
7. В диалоговом окне **Create New Test List** задайте имя списка тестов и щелкните **OK**.
8. В диспетчере Test Manager щелкните узел **All Loaded Tests**.
9. Перетащите нужные элементы из списка доступных тестов в свой узел списка тестов.
10. Верните в систему управления исходным кодом измененный VSMDI-файл проекта модульного теста.

Созданный список тестов будет доступен в новой сборке и сможет выполняться автоматически как часть процесса сборки.

Дополнительные ресурсы

- Подробнее об автоматическом выполнении тестов верификации сборки читайте в статье «How to: Configure and Run Build Verification Tests (BVTs)» по адресу *http://msdn2.microsoft.com/en-us/library/ms182465(VS.80).aspx*.
- Об автоматизированном тестировании сборки без Visual Studio Test Edition или файла VSMDI читайте в статье «How to run tests in a build without test metadata files and test lists (.vsmdi files)» по адресу *http://blogs.msdn.com/buckh/archive/2006/11/04/how-to-run-tests-without-test-metadata-files-and-test-lists-vsmdi-files.aspx*.

Как выполнять автоматизированное тестирование в ходе сборки

Автоматизированное тестирование применяется для получения информации о качестве кода после каждой сборки. Для выполнения автоматизированных тестов необходимо, чтобы на сервере сборки были установлены Visual Studio Developer Edition и Visual Studio Test Edition или полностью Team Suite. Версия Visual Studio Developer Edition необходима для выполнения сборки, а без Test Edition невозможно будет настроить выполняемые тесты и списки тестов.

Выполнение автоматизированного тестирования в процессе сборки

1. Создайте один или несколько автоматизированных тестов, которые нужно выполнять в процессе сборки.
2. В меню **Test** выберите команду **Create New Test List**.
3. С помощью Test Manager сгруппируйте тесты в новый список, перетаскивая их из раздела **Test View** в **Test List**.
4. Создайте новый тип сборки.
5. Установите флажок автоматизированного тестирования.
6. Выберите проект, в рамках которого были созданы тесты и список тестов.
7. Выберите список тестов, который хотите выполнять.

Дополнительные ресурсы

- Подробнее об автоматическом выполнении тестов верификации сборки читайте в статье «How to: Configure and Run Build Verification Tests (BVTs)» по адресу *http://msdn2.microsoft.com/en-us/library/ms182465(VS.80).aspx*.
- Об автоматизированном тестировании сборки без Visual Studio Test Edition или файла VSMDI читайте в статье «How to run tests in a build without test metadata files and test lists (.vsmdi files)» по адресу *http://blogs.msdn.com/buckh/archive/2006/11/04/how-to-run-tests-without-test-metadata-files-and-test-lists-vsmdi-files.aspx*.

Как выполнять анализ кода в ходе сборки

Включить анализ кода в сценарий сборки можно как при создании нового типа сборки, установив соответствующий флажок в New Team Build Type Creation Wizard, так и позже, изменяя файл TFSBuild.proj.

Включение анализа кода в файле TFSBuild.proj

- Если требуется, чтобы анализ кода выполнялся для всех проектов, независимо от их настроек, присвойте тегу **<RunCodeAnalysis>** значение **Always**.

- Если требуется, чтобы анализ кода выполнялся для проектов соответственно их настройкам, присвойте тегу **<RunCodeAnalysis>** значение **Default**.

Дополнительные ресурсы

- Дополнительную информацию об автоматическом анализе кода в ходе сборки вы найдете в разделе «Как автоматически выполнять анализ кода при помощи Team Build» этой книги.
- Подробнее об инструментах анализа кода читайте в статье «Guidelines for Using Code Analysis Tools» по адресу *http://msdn2.microsoft.com/en-us/library/ms182023(VS.80).aspx*.

Как реализовать сбой сборки при непрохождении тестов

Если в сборке происходит сбой из-за ошибок компиляции, создается рабочий элемент для отслеживания ошибки, а сама сборка помечается как неудачная. Однако, если сборка не проходит автоматизированный тест, это не считается сбоем. Непройденный тест преобразуется в предупреждение, и сборка продолжается.

Можно завершать сборку сбоем и при непрохождении автоматизированного теста, попутно автоматически формируя рабочий элемент для отслеживания непройденного теста.

Завершение сборки сбоем при невыполнении теста

1. Откройте файл Microsoft.TeamFoundation.Build.targets из папки Program Files\MSBuild\Microsoft\VisualStudio\v8.0\TeamBuild.
2. Извлеките из системы управления исходным кодом и откройте для редактирования файл TFSBuild.proj для типа сборки, который должен завершаться сбоем в случае непрохождения теста (тестов).
3. Скопируйте цель **RunTestWithConfiguration** из Microsoft.TeamFoundation.Build.targets в самый конец файла TFSBuild.proj, непосредственно перед закрывающим тегом **</Project>**.
4. Измените значение атрибута **ContinueOnError** с **true** на **false**.

> **Примечание** Вам доступны две задачи тестирования. Измените задачу серверной сборки, чтобы изменить поведение сборок только на сервере сборки. Задача клиентской сборки используется при выполнении сборки на компьютере разработчика.

Если вы хотите, чтобы сборка всегда считалась неудачной при завершении теста с ошибкой, измените непосредственно Microsoft.TeamFoundation.Build.targets. Это повлияет на поведение всех типов командной сборки.

Рекомендованное выше решение легко реализуется, но нет гарантии, что оно будет работать в будущих версиях Visual Studio. Чтобы реализовать ре-

шение, которое обязательно будет работать после обновления, читайте статью «Determining Whether Tests Passed in Team Build» в блоге Аарона Холлберга (Aaron Hallberg) по адресу *http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether-tests-passed-in-team-build.aspx*.

Дополнительные ресурсы

- Дополнительную информацию о настройке сборки для создания рабочих элементов при завершении теста с ошибкой вы найдете в статье «Create Workitems for Test Failures in TeamBuild» по адресу *http://blogs.msdn.com/nagarajp/archive/2005/10/14/481290.aspx*.
- Дополнительную информацию о решении, которое будет работать после обновления Visual Studio, вы найдете в статье «Determining Whether Tests Passed in Team Build» в блоге Аарона Халлберга (Aaron Hallberg) по адресу *http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether -tests-passed-in-team-build.aspx*.

Дополнительные ресурсы по Team Build

- Более общую информацию о Team Foundation Build вы найдете в обзоре «Overview of Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms181710(VS.80).aspx*.

Практические рекомендации: управление проектом

В этом разделе

Политики возврата после правки

- Как повысить качество кода при помощи политики возврата после правки.
- Как при помощи политики обязать разработчиков связывать рабочие элементы с возвратом после правки.
- Как настроить политики возврата для соблюдения стандартов программирования.

Управление проектом

- Как управлять проектом при помощи Microsoft Project.
- Как управлять проектом при помощи Microsoft Excel.
- Как создать минимальный шаблон процесса.
- Как настроить шаблон процесса.
- Как настроить тип рабочего элемента в шаблоне процесса.
- Как настроить тип рабочего элемента в существующем командном проекте.
- Как создать итерацию.
- Как создать область.
- Как добавить уведомление о возврате после правки.
- Как настроить панель отчетов.
- Как создавать папки в хранилище системы управления исходным кодом.
- Как удалить проект из Team Foundation Server.

Политики возврата после правки

- Как повысить качество кода при помощи политики возврата после правки.
- Как при помощи политики обязать разработчиков связывать рабочие элементы с возвратом после правки.
- Как настроить политики возврата для соблюдения стандартов программирования.

Как повысить качество кода при помощи политики возврата после правки

Сочетая политики анализа и тестирования, вы обеспечите соблюдение стандартов качества кода. Например, встроенная в VSTS политика тестирования обеспечит обязательное проведение специальных тестов перед возвратом кода в систему управления исходным кодом Microsoft® Visual Studio® 2005 Team Foundation Server (TFS). Также можно настроить политику анализа кода, обеспечив с ее помощью соответствие кода определенным нормам безопасности, производительности, переносимости, удобства сопровождения и надежности.

Применяя этот тип политики возврата после правки в дополнение к политикам, направленным на укрепление стандартов и нормативов программирования, вы гарантируете соответствие кода специфическим критериям качества.

Применение политики анализа при возврате после правки

1. В окне Team Explorer щелкните правой кнопкой нужный командный проект, раскройте подменю **Team Project Settings** и выберите команду **Source Control**.
2. Перейдите на вкладку **Check-in Policy**.
3. Щелкните кнопку **Add**, а затем выберите и настройте соответствующую политику.

Дополнительные ресурсы

- Дополнительную информацию о создании и использовании пользовательской политики возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о настройке политики возврата после правки вы найдете в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- С примером кода, запрещающего определенные варианты программирования, можно ознакомиться по материалам статьи «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.

- С примером кода, заставляющего добавлять комментарии при возврате после правки, можно познакомиться в статье «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- Дополнительную информацию о том, как зарегистрировать новую политику возврата после правки, вы найдете в статье «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Как при помощи политики обязать разработчиков связывать рабочие элементы с возвратом после правки

Используйте новую стандартную политику возврата после правки **Work Item**, которая заставляет разработчиков связывать с возвратом рабочие элементы. Это поможет определить связь между изменениями, внесенными в исходный код, и рабочими элементами, отслеживающими ошибки и задания.

Применение политики, обязывающей разработчиков связывать рабочие элементы с возвратом после правки

1. В окне Team Explorer правой кнопкой мыши щелкните нужный командный проект, раскройте подменю **Team Project Settings** и выберите команду **Source Control**.
2. Перейдите на вкладку **Check-in Policy**.
3. Щелкните кнопку **Add**, а затем выделите и настройте политику **Work Item**.

Дополнительные ресурсы

- Дополнительную информацию о создании и использовании пользовательской политики возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о настройке политики возврата после правки вы найдете в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.

Как настроить политики возврата для соблюдения стандартов программирования

Политика анализа кода при возврате после правки, включенная в комплект Team Foundation Server, позволяет автоматически проводить статический анализ кода, гарантируя его соответствие соответствующим нормативам. Вы можете детально настроить политику анализа для проверки множества различных правил, определяющих требования к дизайну, интероперабельности, удобству сопровождения, соглашениям об именах, надежности и др.

Применение политики анализа кода для соблюдения стандартов программирования

1. В окне Team Explorer правой кнопкой мыши щелкните нужный командный проект, раскройте подменю **Team Project Settings** и щелкните команду **Source Control**.
2. Перейдите на вкладку **Check-in Policy** и щелкните кнопку **Add**.
3. В диалоговом окне **Add Check-in Policy** установите параметр **Code Analysis** и щелкните **OK**.
4. В редакторе **Code Analysis Policy Editor** задайте параметр **Enforce C/C++ Code Analysis (/analyze)**, **Enforce Code Analysis For Managed Code** или оба, если ваш проект содержит сочетание управляемого и неуправляемого кода.
5. Если вы выбрали анализ управляемого кода, задайте необходимые правила, руководствуясь собственными стандартами программирования.

Кроме того, вы вольны создать пользовательскую политику возврата после правки для выполнения проверок, не выполняющихся по умолчанию. Можно запретить определенные виды кода, например, вызовы запрещенных функций API, или написать политику для соблюдения стиля программирования, принятого в вашей команде, задав, например, как расставлять фигурные скобки в исходном коде.

Дополнительные ресурсы

- Дополнительную информацию о создании и использовании пользовательской политики возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о настройке политики возврата после правки вы найдете в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- С примером кода, запрещающего определенные варианты программирования, можно познакомиться в статье «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- С примером кода, заставляющего добавлять комментарии при возврате после правки, можно ознакомиться по материалам статьи «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- Дополнительную информацию о том, как зарегистрировать новую политику возврата после правки, вы найдете в статье «I've Made a New Check-

In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Управление проектом

- Как управлять проектом при помощи Microsoft Project.
- Как управлять проектом при помощи Microsoft Excel.
- Как создать минимальный шаблон процесса.
- Как настроить шаблон процесса.
- Как настроить тип рабочего элемента в шаблоне процесса.
- Как настроить тип рабочего элемента в существующем командном проекте.
- Как создать итерацию.
- Как создать область.
- Как добавить уведомление о возврате после правки.
- Как настроить панель отчетов.
- Как создавать папки в хранилище системы управления исходным кодом.
- Как удалить проект из Team Foundation Server.

Как управлять проектом при помощи Microsoft Project

Используйте Microsoft Office Project для создания заданий по расписанию, выявления зависимостей задач, сбалансированного распределения ресурсов и определения сроков завершения. Чтобы управлять проектом при помощи Microsoft Project, нужно предпринять следующие шаги:

- Создать план проекта.
- Составить набор задач, разработать для них расписание и опубликовать их в Team Foundation Server.
  - Эти задачи отображаются в очереди рабочих элементов соответствующего разработчика.
  - Члены команды работают над заданиями и сообщают о продвижении работ в Team Explorer, устанавливая статус рабочего объекта.
- Обновить план проекта для извлечения актуальной информации. Это позволяет следить за развитием проекта в Microsoft Project.

Публикация плана проекта в TFS

1. Создавая план нового проекта, определите задачи, время их выполнения, распределение ресурсов, зависимости и другие сведения, которые вы обычно задаете в Microsoft Office Project.
2. В окне Microsoft Office Project в меню **Team** выберите команду **Choose Team Project**.

3. Выберите Team Foundation Server для своего командного проекта.
4. Выделите свой командный проект.
5. Щелкните **OK**.
6. В столбце **Work Item Type** укажите тип для каждого рабочего элемента, который хотите опубликовать в TFS.
7. В столбце Sync выберите вариант **Do Not Publish** для суммарных задач, которые не хотите публиковать в TFS.
8. Если у вас есть задачи, назначенные более чем одному ресурсу, разделите их на отдельные задачи, порученные одному ресурсу. (В настоящий момент Team Foundation Server не поддерживает назначения рабочего элемента нескольким ресурсам.) При необходимости сгруппируйте отдельные задачи в суммарную задачу, чтобы получить возможность пользоваться автоматическим вычислением.

> **Совет** Чтобы сгруппировать наборы задач, создайте в TFS набор областей, а затем сгруппируйте задачи при помощи столбца **Area**. Подробнее — в статье «How to: Modify the Team Project Areas» по адресу *http://msdn2.microsoft.com/en-us/library/ms181479(VS.80).aspx*.

9. Щелкните команду **Publish** на панели инструментов **Work Item**, чтобы опубликовать план проекта в TFS.

Дополнительные ресурсы

- Дополнительную информацию о работе с рабочими элементами в Microsoft Project вы найдете в статье «Working with Work Items in Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms244368(VS.80).aspx*.
- Дополнительную информацию об импорте рабочих элементов вы найдете в статье «How to: Import Work Items in Microsoft Excel or Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms181676(VS.80).aspx*.
- Дополнительную информацию об управлении проектом в Microsoft Project вы найдете в статье «Tracking Team Projects in Microsoft Excel and Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms244373(VS.80).aspx*.

Как управлять проектом при помощи Microsoft Excel

Используйте Microsoft Office Excel® для хранения, сортировки, фильтрования и управления требованиями, сценариями, проблемами, ошибками, рисками и рабочими элементами.

Существует два способа создания списка рабочих элементов. Первый позволяет выбрать запрос рабочего элемента в Team Explorer и создать привязанную к нему электронную таблицу со списком рабочих элементов, за-

полненным данными из запроса. Второй способ — создать список рабочих элементов в Excel, используя надстройку для выбора проекта и импорта рабочих элементов.

Создание списка рабочих элементов в Excel

1. В окне Microsoft Office Excel выберите в меню **Team** команду **New List**.
2. В окне **Connect to a Team Foundation Server** выберите сервер, к которому нужно подключиться, или щелкните **Servers** для ввода информации о сервере вручную.
3. В окне **Team Projects** выберите командный проект на сервере TFS, с которым хотите работать. Документ привязан к командному проекту.
4. Щелкните **OK**.
5. Выберите тип списка. Чтобы создать список на базе запроса выберите **Query List** и выберите запрос в раскрывающемся списке **Select a Query**.
6. Укажите столбцы, которые должны присутствовать в новом списке рабочих элементов.
7. Импортируйте нужные рабочие элементы.
8. Сохраните электронную таблицу или опубликуйте новые рабочие элементы в БД рабочих элементов, выбрав команду **Publish Changes** в меню **Team**.

Дополнительные ресурсы

- Дополнительную информацию о работе со списками рабочих элементов вы найдете в статье «Working with Work Item Lists in Microsoft Excel» по адресу *http://msdn2.microsoft.com/en-us/library/ms181694(VS.80).aspx*.
- Дополнительную информацию о создании списков рабочих элементов вы найдете в статье «How to: Create a Work Item List» по адресу *http://msdn2.microsoft.com/en-us/library/ms181695(VS.80).aspx*.
- Дополнительную информацию об импорте рабочих элементов вы найдете в статье «How to: Import Work Items in Microsoft Excel or Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms181676(VS.80).aspx*.

Как создать минимальный шаблон процесса

Если вам не требуется поддержка функций управления проектом TFS, выходящих за рамки системы управления исходным кодом, вам вполне хватит минимального шаблона процесса. Чтобы создать минимальный шаблон процесса при помощи диспетчера Process Template Manager, загрузите шаблон на локальный компьютер, отредактируйте его, убрав неиспользуемые разделы, а затем выгрузите шаблон обратно на сервер.

Создание шаблона процесса с поддержкой только управления исходным кодом

1. В окне Team Explorer щелкните правой кнопкой имя сервера. Затем выберите команды **Team Foundation Server Settings** и **Process Template Manager**.
2. Выберите шаблон, который хотите редактировать, и щелкните команду **Download**.
3. В папке, в которой вы сохранили шаблон, откройте для редактирования файл Process Template.xml.
4. Введите в элемент <name> имя процесса.
5. В разделе <plugins> удалите модули **Reporting**, **Portal** и **WorkItemTracking**.
6. В разделе <groups> удалите группы **Reporting**, **Portal** и **WorkItemTracking**.
7. В группе VersionControl найдите раздел <dependencies> и удалите зависимость **WorkItemTracking**.
8. Удалите из папки, в которой сохранен шаблон, подпапки Reports, Windows Sharepoint Services и WorkItem Tracking.
9. Сохраните изменения.
10. В окне Team Explorer щелкните правой кнопкой имя сервера и выберите команды **Team Foundation Server Settings** и **Process Template Manager**.
11. Щелкните **Upload** и выберите шаблон, который вы хотите выгрузить.

Выгрузив измененный шаблон процесса, вы сможете использовать его при создании новых командных проектов.

Дополнительные ресурсы

- Дополнительную информацию о шаблонах процессов вы найдете в главе 13 этой книги.
- Дополнительную информацию о настройке шаблона процесса вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги и в статье «Customizing Process Templates» по адресу *http://msdn2.microsoft.com/en-us/library/ms243782(VS.80).aspx*.

Как настроить шаблон процесса

Настройка шаблона процесса требуется, чтобы привести стандартные типы рабочих элементов, параметры безопасности, параметры системы управления исходным кодом и отчетов в соответствие с требованиями, предъявляемыми к процессу в вашей организации.

Настройка шаблона процесса

1. Просмотрите шаблоны, имеющиеся в системе, и выберите один из них, наиболее подходящий для вашей организации.

2. Загрузите шаблон выбранного процесса.
3. Произведите необходимую настройку шаблона, измените типы рабочих элементов, параметры безопасности и параметры системы управления исходным кодом.
4. Выгрузите настроенный шаблон в TFS.
5. Убедитесь, что внесенные вами изменения удовлетворяют нужным требованиям.

   Есть два способа настройки выполняющегося процесса:

- **Вручную отредактировать XML-файлы.** Любое действие вручную чревато многочисленными ошибками, но с другой стороны оно позволяет настроить шаблон с высочайшей точностью.
- **Воспользоваться инструментом Process Template Editor** из комплекта Microsoft Visual Studio 2005 Team Foundation Server Power Tool — графическим редактором для просмотра и настройки шаблонов процесса. Подключившись к TFS, его можно использовать для настройки определений типов рабочих элементов и глобальных списков активного проекта.

Дополнительные ресурсы

- Дополнительную информацию о шаблонах процессов вы найдете в главе 13 этой книги.
- Дополнительную информацию о настройке шаблона процесса вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги и в статье «Customizing Process Templates» по адресу *http://msdn2.microsoft.com/en-us/library/ms243782(VS.80).aspx*.

Как настроить тип рабочего элемента в шаблоне процесса

Для настройки типа рабочего элемента используется инструмент Process Editor из последней версии TFS Power Tool. Его можно загрузить по адресу *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.

Настройка типов рабочего элемента

Загрузите наиболее подходящий шаблон, выполнив следующие действия:

1. В Visual Studio выберите в меню **Team** команду **Team Foundation Server Settings**.
2. Щелкните **Process Template Manager**.
3. В диалоговом окне **Process Template Manager** выберите шаблон процесса, который хотите изменить, и щелкните кнопку **Download**.
4. Выберите папку на локальном диске, в которой хотите сохранить шаблон, и щелкните кнопку **Save**.

Откройте шаблон процесса в Process Editor и отредактируйте типы рабочих элементов, как описано ниже:

1. В окне Visual Studio откройте меню **Team**.
2. Щелкните **Process Editor** и **Open Process Template**.
3. В диалоговом окне **Open Process Template fileset** перейдите к загруженному шаблону процесса и щелкните **Open**. Файл ProcessTemplate.xml будет загружен в Visual Studio.
4. Задайте имя для настраиваемой методологии.
5. В окне **Process Template Explorer** щелкните кнопку **Work Item Tracking**.
6. Перейдите на вкладку **Type Definition**.
7. Щелкните кнопку **Add**, расположенную на панели инструментов справа, чтобы создать новый рабочий элемент.
8. В диалоговом окне **New Work Item Type** введите имя типа рабочего элемента и выберите существующий тип рабочего элемента из раскрывающегося списка **Copy From**. Создается новый тип рабочего элемента, отображающийся в списке **Item List** на правой панели вкладки **Type Definition**.
9. Чтобы сохранить изменения, выберите в меню **File** команду **Save**.
10. На вкладке **Type Definitions** щелкните правой кнопкой тип рабочих элементов, который хотите отредактировать, и выберите команду **Open**. Выбранный тип откроется в новом окне Visual Studio.
11. При необходимости добавьте и удалите атрибуты или поля в редактируемом типе рабочих элементов.

Дополнительные ресурсы

- Загрузить TFS Power Tool можно из источника, размещенного по адресу *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.
- Дополнительные информацию о рабочих элементах и о настройке типов рабочих элементов вы найдете в главе 12 этой книги.

Как настроить тип рабочего элемента в существующем командном проекте

Есть два способа редактирования существующего типа рабочего элемента: из командной строки и при помощи инструмента Process Template Editor из комплекта TFS Power Tool.

Для редактирования типа рабочего элемента из командной строки используются инструменты witexport и witimport, которые находятся в папке %programfiles%\Program Files\Microsoft Visual Studio 8\Common7\IDE компьютера, на котором установлен Team Explorer.

Экспорт, редактирование и импорт типа рабочего элемента

1. Запустите команду **witexport**, как показано ниже, чтобы экспортировать тип рабочего элемента:

```
witexport /f task.xml /t http://TFSServer:8080 /p MyTeamProject /n Task
```

2. Отредактируйте определение типа рабочего элемента.
3. Запустите команду **witimport**, как показано ниже, чтобы импортировать измененный тип:

```
witimport /f task.xml /t http://TFSServer:8080 /p MyTeamProject
```

В предыдущем примере подразумевается экспорт типа рабочего элемента Task из проекта MyTeamProject на сервер с именем TFSServer.

Редактирование типа рабочего элемента с помощью Process Template Editor

Чтобы импортировать и отредактировать тип рабочего элемента, выполните следующие действия:

1. В окне Visual Studio в меню **Team** выберите команду **Process Template Editor**. Затем щелкните **Work Item Types** и выберите **Export WIT**.
2. В диалоговом окне **Connect to Team Foundation Server** введите URL сервера.
3. В диалоговом окне **Select Work Item Type** выберите тип рабочего элемента, экспорт которого хотите осуществить.
4. Сохраните тип рабочего элемента.
5. Сохраните все глобальные списки, которые собираетесь редактировать.
6. Отредактируйте тип рабочего элемента и сохраните изменения.

Чтобы экспортировать тип рабочего элемента, выполните следующие действия:

1. В Visual Studio в меню **Team** выберите команду **Process Editor**. Затем щелкните **Work Item Types** и выберите **Import WIT**.
2. В диалоговом окне **Connect to Team Foundation Server** введите URL сервера.
3. В диалоговом окне **Import Work Item Type** перейдите к отредактированному типу рабочего элемента и выберите командный проект, в который хотите его экспортировать.
4. Щелкните **OK**.

Дополнительные ресурсы

- Загрузить TFS Power Tool можно из источника, размещенного по адресу *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.

- Дополнительную информацию об использовании инструмента Process Editor вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.
- Дополнительные сведения об использовании команды **witimport** вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms253163(VS.80).aspx*.
- Дополнительные сведения об использовании команды **witexport** вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms253051(VS.80).aspx*.

Как создать итерацию

Итерации применяются для группирования рабочих элементов, которое может использоваться при создании рабочих элементов, в запросах рабочих элементов и при составлении отчетов.

Создание итерации

1. В Team Explorer щелкните командный проект.
2. В меню **Team** раскройте подменю **Team Project Settings** и выберите команду **Areas and Iterations**.
3. В диалоговом окне **Areas and Iterations** перейдите на вкладку **Iteration**.
4. Щелкните кнопку **Add a child node** на панели инструментов.
5. Щелкните правой кнопкой новый узел, выберите команду **Rename** и введите имя итерации.
6. Щелкните узел **Iteration**.
7. Повторите шаги 2, 3 и 4, чтобы создать дополнительные итерации.
8. Щелкните **Close**.

> **Примечание** В шаблон процесса MS Agile включены три предопределенные итерации. При необходимости вы вольны удалить эти итерации, переименовать их или оставить неизменными.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Walkthrough: Creating a New Team Project» по адресу *http://msdn2.microsoft.com/en-us/library/dhedaeb2(VS.80).aspx*.

Как создать область

Области создаются для разделения работ по проекту на подобласти, связанные, например, с разработкой интерфейса, приложения и БД. Создав области, вы можете назначать им сценарии и рабочие элементы.

Начинайте с одиночной области корневого уровня. Позднее, с развитием проекта, вы будете по мере необходимости создавать новые области.

Создание областей

1. В Team Explorer щелкните ваш проект.
2. В меню **Team** раскройте подменю **Team Project Settings** и выберите команду **Areas and Iterations**.
3. В диалоговом окне **Areas and Iterations** перейдите на вкладку **Area**.
4. Щелкните кнопку **Add a child node** на панели инструментов.
5. Щелкните новый узел правой кнопкой, выберите команду **Rename** и введите имя области.
6. Щелкните узел **Area**.
7. Повторяйте шаги 2, 3 и 4, чтобы создать дополнительные области. Так вы постепенно создадите иерархию структуры проекта.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Walkthrough: Creating a New Team Project» по адресу *http://msdn2.microsoft.com/en-us/library/dhedaeb2(VS.80).aspx*.

Как добавить уведомление о возврате после правки

Для подписки на уведомления о возврате после правки используется инструмент bissubscribe, расположенный в папке %programfiles%\Microsoft Visual Studio 2005 Team Foundation Server\TF Setup на сервере TFS.

Создание уведомления о возврате после правки

1. Откройте командную строку и перейдите в папку C:\Program Files\Microsoft Visual Studio 2005 Team Foundation Server\TF Setup\.
2. Если вы хотите получать уведомления по электронной почте, используйте команду

```
bissubscribe /eventType CheckinEvent /address someone@domain.com /
deliveryType EmailHtml /domain http://TFSRTM:8080
```

3. Чтобы задать уведомление посредством веб-службы, используйте команду

```
bissubscribe /eventType CheckinEvent /address http://TFSRTM:8080/ci/
notify.asmx /deliveryType Soap /domain http://TFSRTM:8080
```

Если у вас возникли ошибки, а также если вы хотите удостовериться в правильности регистрации уведомления, выполните следующие действия:

1. Откройте SQL Server Management Studio.
2. Откройте БД **tfsIntegration**.

3. Просмотрите таблицу **tbl_subscription.**

   В таблице **tbl_subscription** перечислены все события, на которые задана подписка. Найдите в этой таблице записи о событиях, на которые вы подписаны. Чтобы отписаться от события, удалите из таблицы запись о нем или воспользуйтесь командой **bissubscribe** с параметром **/unsubscribe** и идентификатором события, например:

```
bissubscribe /delete/id [id] /server http://TFSRTM:8080
```

Дополнительные ресурсы

- Дополнительную информацию о непрерывной сборке вы найдете в разделе «Как настроить непрерывную сборку в Visual Studio Team Foundation Server» этой книги и в статье «Continuous Integration using Team Build» по адресу *http://blogs.msdn.com/khushboo/archive/2006/01/04/509122.aspx*.
- Дополнительную информацию об использовании команды bissubscribe вы найдете в статье «Adding a path filter to a CheckinEvent subscription using bissubscribe» по адресу *http://blogs.msdn.com/buckh/archive/2006/09/29/checkinevent-path-filter.aspx*.

Как настроить панель отчетов

Чтобы собрать разнообразную информацию о проекте в одном месте, создайте на портале командного проекта Microsoft Office SharePoint® панель отчетов (report dashboard). Вот список отчетов, которые на ней можно разместить:

- **Remaining Work** Оставшаяся работа.
- **Quality Indicators** Показатели качества.
- **Bug Rates** Частота появления ошибок.
- **Project Velocity** Темп выполнения проекта.

В каждый отчет, который вы хотите отобразить на панели, нужно добавить веб-часть Report Viewer.

Изменение портала командного проекта и создание панели отчетов

1. Установите веб-часть Report Viewer на сервер отчетов. Для этого используются инструмент stsadm.exe и файл RSWebParts.cab, входящие в дистрибутив Microsoft Office SharePoint и Report Services.
   - Инструмент STSADM.EXE находится в папке C:\Program Files\Common Files\Microsoft Shared\web server extensions\60\BIN.
   - Файл RSWebParts.Cab находится в папке C:\ Program Files\Microsoft SQL Server\90\Tools\Reporting Services\SharePoint.

   Пример использования:

```
STSADM.EXE -o addwppack -filename "C:\ Program Files\Microsoft SQL
Server\90\Tools\Reporting Services\SharePoint\RSWebParts.cab" -globalin-
stall
```

2. В Team Explorer щелкните правой кнопкой нужный проект и выберите команду **Show Project Portal**.
3. Щелкните **Modify Shared Page**, разверните подменю **Browse** и выберите команду **Add Web Parts**.
4. Щелкните **Virtual Server Gallery**.
5. В списке **Web Part List** выберите вариант **Report Viewer**.
6. Щелкните кнопку **Add**.
7. Введите имя диспетчера отчетов (Report Manager), например, http://<*сервер отчетов*>/reports.
8. Введите путь к отчету, который хотите отобразить, например, <*мой проект*>/Quality Indicators.

Дополнительные ресурсы

- Дополнительную информацию о добавлении компонента Report Viewer вы найдете в статье «Viewing Reports with SharePoint 2.0 Web Parts» по адресу *http://msdn2.microsoft.com/en-us/library/ms159772(SQL.90).aspx*.
- Дополнительную информацию о портале командного проекта вы найдете в статье «Using the Team Project Portal» по адресу *http://msdn2.microsoft.com/en-us/library/ms242883(VS.80).aspx*.

Как создавать папки в хранилище системы управления исходным кодом

Создать папки в хранилище системы управления рабочим кодом можно при помощи Team Explorer, а также путем создания структуры папок в вашей рабочей области и последующим возвратом незавершенных изменений.

Создание структуры папок на сервере

1. В Team Explorer разверните нужный командный проект.
2. Дважды щелкните **Source Control**.
3. В окне Source Control Explorer выберите корневой узел, щелкните правой кнопкой панель **Local Path** и выберите команду **New Folder**.
4. Введите имя корневой папки и нажмите **Enter**.
5. Повторяйте шаги 3 и 4 для создания других папок в хранилище системы управления исходным кодом.

Создание структуры папок на клиенте

1. Создайте сопоставление рабочей области на локальном компьютере.
2. Создайте требующуюся для проекта структуру папок.

При первом возврате незавершенных изменений структура папок будет скопирована на сервер.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию о создании дерева исходного кода вы найдете в разделе «Как создать дерево кода в Team Foundation Server» этой книги.

Как удалить проект из Team Foundation Server

Средствами Team Explorer удалить проект нельзя, придется воспользоваться инструментом командной строки TfsDeleteProject. Он находится в папке Program Files\Microsoft Visual Studio 8\Common7\IDE на компьютере, где установлен Team Explorer.

Удаление проекта из TFS

1. Откройте командную строку и перейдите в папку C:\Program Files\Microsoft Visual Studio 2005 Team Foundation Server\TF Setup\.
2. Запустите команду **TfsDeleteProject**, как показано в примере:

```
TfsDeleteProject /server:TfsServer TeamProjectName
```

Дополнительные ресурсы

- Дополнительную информацию об удалении проектов вы найдете в статье «TFSDeleteProject» по адресу *http://msdn2.microsoft.com/en-us/library/ms181482(VS.80).aspx*.

Дополнительные ресурсы по управлению проектами Team Foundation

- Дополнительную информацию о шаблонах процесса MSF вы найдете в статье «Process Templates» по адресу *http://msdn2.microsoft.com/en-us/teamsystem/aa718801.aspx*.

Практические рекомендации: работа с отчетами

В этом разделе

Администрирование

- Как создать панель отчетов.
- Как предоставлять разрешения на доступ к отчетам.

Создание и настройка

- Как модифицировать существующий отчет.
- Как создать новый отчет в Visual Studio.
- Как создать новый отчет в Excel.
- Как создать снимок отчета по расписанию.
- Как подписаться на отчет.
- Как добавить отчет в существующий шаблон процесса.

Просмотр

- Как проанализировать состояние проекта.
- Как проанализировать качество приложения.
- Как просмотреть оставшуюся работу.
- Как просмотреть состояние сборки.
- Как просмотреть ошибки и результаты тестов.
- Как сравнить запланированную работу с фактической.
- Как определить владельца последней редакции файла.
- Как найти все изменения, внесенные в код разработчиком.
- Как найти все изменения, внесенные в файл.

- Как найти все изменения в коде, связанные с конкретным рабочим элементом.
- Как сгенерировать показатели изменяемости кода.
- Как сгенерировать показатели рабочей области (файлов, строки кода, количество проектов).

Администрирование

- Как создать панель отчетов.
- Как предоставлять разрешения на доступ к отчетам.

Как создать панель отчетов

Модифицируйте сайт портала Microsoft® Office SharePoint® командного проекта, чтобы создать панель отчета. Она позволяет обобщать разнообразную информацию проекта в едином расположении. Функциональная панель отчетов должна, вероятно, включать следующие отчеты:

- об оставшейся работе;
- о показателях качества;
- о частоте появления ошибок;
- о темпе продвижения проекта.

Вы вольны добавлять новые отчеты на страницу портала SharePoint. Для этого в каждый отчет, который вы хотите отобразить на странице, нужно добавить компонент Report Viewer Web Part.

Изменение портала командного проекта и создание панели отчетов

1. Установите компонент Report Viewer Web Part на сервер отчетов. Для этого используются инструмент stsadm.exe и файл RSWebParts.cab, которые входят в дистрибутив Microsoft Office SharePoint и Report Services, например:

   ```
   STSADM.EXE -o addwppack -filename "C:\ Program Files\Microsoft SQL
   Server\90\Tools\Reporting Services\SharePoint\RSWebParts.cab" -
   globalinstall
   ```

   - Инструмент STSADM.EXE находится в папке C:\Program Files\Common Files\Microsoft Shared\web server extensions\60\BIN.
   - Файл RSWebParts.Cab находится в папке C:\ Program Files\Microsoft SQL Server\90\Tools\Reporting Services\SharePoint.
2. В Team Explorer щелкните правой кнопкой ваш проект.
3. Выберите команду **Show Project Portal**.
4. Щелкните **Modify Shared Page**.
5. Наведите указатель на **Browse** и щелкните **Add Web Parts**.

6. Щелкните **Virtual Server Gallery**.
7. В списке **Web Part List** выберите вариант **Report Viewer**.
8. Щелкните кнопку **Add**.
9. Введите имя диспетчера отчетов, например, http://<*сервер отчетов*>/reports.
10. Введите путь отчета, который хотите отобразить, например: <*мой проект*>/Quality Indicators.

Дополнительные ресурсы

- Дополнительную информацию о добавлении компонента Report Viewer Web Part вы найдете в статье «Viewing Reports with SharePoint 2.0 Web Parts» по адресу *http://msdn2.microsoft.com/en-us/library/ms159772(SQL.90).aspx*.
- Дополнительную информацию о портале командного проекта вы найдете в статье «Using the Team Project Portal» по адресу *http://msdn2.microsoft.com/en-us/library/ms242883(VS.80).aspx*.

Как предоставлять разрешения на доступ к отчетам

При помощи списка разрешений отчета вы определяете пользователей, которым можно редактировать и просматривать отчеты. Для установки разрешений вы должны быть членом роли Content Manager в Microsoft SQL Server™ Reporting Services.

Предоставление разрешение на доступ ко всем отчетам командного проекта

1. В Team Explorer разверните узел проекта.
2. Щелкните правой кнопкой элемент **Reports** и выберите команду **Show Report Site**.
3. Перейдите на вкладку **Properties**.
4. Щелкните **Security**.
5. Щелкните **Edit Item Security**.
6. Чтобы изменить разрешения безопасности для уже определенной роли, щелкните **Edit**.
7. Чтобы задать разрешения безопасности для роли, которой нет в списке, щелкните **New Role Assignment**.

Установка разрешений для одного отчета

1. В Team Explorer разверните узел проекта.
2. Щелкните правой кнопкой элемент **Reports** и выберите **Show Report Site**.

3. На сайте отчетов выберите отчет, для которого хотите задать разрешения.
4. Перейдите на вкладку **Properties**.
5. Щелкните **Security**.
6. Щелкните **Edit Item Security**.
7. Чтобы изменить разрешения безопасности для уже определенной роли, щелкните **Edit**.
8. Чтобы задать разрешения безопасности для роли, которой нет в списке, щелкните **New Role Assignment**.

Дополнительные ресурсы

- Дополнительную информацию о разрешениях отчетов вы найдете в статье «How to: Set Permissions for a Report» по адресу *http://msdn2.microsoft.com/en-us/library/ms181645(VS.80).aspx*.

Создание и настройка

- Как модифицировать существующий отчет.
- Как создать новый отчет в Visual Studio.
- Как создать новый отчет в Excel.
- Как создать снимок отчета по расписанию.
- Как подписаться на отчет.
- Как добавить отчет в существующий шаблон процесса.

Как модифицировать существующий отчет

Существующие отчеты модифицируются при помощи инструмента Microsoft SQL Server™ 2005 Reporting Services Designer, входящего в Visual Studio (Business Intelligence Development Studio), который поставляется с клиентскими инструментами SQL Server 2005. Часто модифицировать существующий отчет проще, чем создать новый.

Создание проекта отчета

1. В Visual Studio откройте меню **File** и выберите команды **New** и **Project**.
2. Выберите тип отчета **Business Intelligence Project**.
3. Выберите шаблон **Report Server Project**.
4. Укажите имя проекта в поле **Name** и его расположение в поле **Location**. Щелкните **OK**.

Экспорт модифицируемого отчета

1. Щелкните проект правой кнопкой и в контекстном меню выберите команду **Show Project Portal**.

2. На панели **Quick Launch**, находящейся слева, щелкните **Reports**.
3. Выделите отчет, который хотите настроить.
4. Щелкните **Properties**.
5. Выберите **Edit**.
6. Сохраните файл отчета .rdl в папке проекта, созданного ранее.

Добавление источников данных

1. Создайте источник данных хранилища:
   a. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
   б. На вкладке **General** введите **TfsReportDS** в текстовое поле **Name**.
   в. В списке **Type** выберите **Microsoft SQL Server**.
   г. Щелкните **Edit**.
   д. Введите имя сервера уровня данных.
   е. Выберите базу данных **TFSWarehouse**.
   ж. Дважды щелкните **ОК**, чтобы добавить источник данных.
2. Создайте источник данных OLAP:
   a. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
   б. На вкладке **General** введите **TfsOlapReportDS** в поле **Name**.
   в. В списке **Type** выберите **Microsoft SQL Server**.
   г. Щелкните **Edit**.
   д. Введите имя сервера уровня данных.
   е. Выберите базу данных **TFSWarehouse**.
   ж. Дважды щелкните **ОК**, чтобы добавить источник данных.

Добавление отчета в проект

1. В окне Solution Explorer щелкните правой кнопкой **Reports**, затем щелкните **Add** и **Existing Item**.
2. Найдите файлу .rdl, экспорт которого выполнили ранее.

Редактирование отчета

1. Измените операторы запросов на панели данных.
2. Перетащите на панель данных новые критерии или членов.
3. Измените разметку отчета на панели Layout Pane.

Примечание Вы, конечно, можете использовать построитель отчетов (Report Builder), который имеется на сайте отчетов команды, но этот инструмент не очень хорошо поддерживается сценариями отчетов Visual Studio, поэтому работать с ним не рекомендуется.

Дополнительные ресурсы

- Более подробную информацию вы найдете в разделе «Как настроить отчет в Visual Studio 2005 Team Foundation Server» этой книги.

Как создать новый отчет в Visual Studio

Создание отчетов также осуществляется при помощи инструмента Microsoft SQL Server™ 2005 Reporting Services Designer, входящего в Visual Studio (Business Intelligence Development Studio) из клиентского комплекта SQL Server 2005.

Новый отчет следует создавать лишь в том случае, если у вас нет отчетов, которые при изменении будут удовлетворять вашим новым потребностям. Часто модифицировать существующий отчет проще, чем создать новый.

Создание проекта отчета

1. В Visual Studio откройте меню **File** и выберите команды **New** и **Project**.
2. Выберите тип отчета **Business Intelligence Project**.
3. Выберите шаблон **Report Server Project**.
4. Укажите имя проекта в поле **Name** и его расположение в поле **Location**. Щелкните **OK**.

Добавление источников данных

1. Создайте источник данных хранилища:
   а. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
   б. На вкладке **General** введите **TfsReportDS** в текстовое поле **Name**.
   в. В списке **Type** выберите **Microsoft SQL Server**.
   г. Щелкните **Edit**.
   д. Введите имя сервера уровня данных.
   е. Выберите базу данных **TFSWarehouse**.
   ж. Дважды щелкните **OK**, чтобы добавить источник данных.
2. Создайте источник данных OLAP:
   а. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
   б. На вкладке **General** введите **TfsOlapReportDS** в поле **Name**.
   в. В списке **Type** выберите **Microsoft SQL Server**.
   г. Щелкните **Edit**.
   д. Введите имя сервера уровня данных.
   е. Выберите базу данных **TFSWarehouse**.
   ж. Дважды щелкните **OK**, чтобы добавить источник данных.

Создание нового шаблона

1. В окне Solution Explorer щелкните правой кнопкой **Reports** и выберите команды **Add** и **New Item**.
2. Выберите шаблон **Report**.
3. Присвойте имя шаблону и щелкните **OK**.

Редактирование шаблона

1. Если окно Report Designer не открывается автоматически, откройте отчет для редактирования, дважды щелкнув его в Solution Explorer.
2. В раскрывающемся списке **Dataset** выберите вариант **New Dataset**.
3. Задайте имя набора данных, например, **TestDataSet**.
4. Задайте параметр **TFSOlapReportDS (shared).**
5. Щелкните **OK**.
6. Щелкните многоточие рядом с кнопкой **Build** (прямо под списком **Dataset**), после чего выберите **Team System**.

Теперь вы можете изменять отчет, перетаскивая меры и измерения из дерева **Dataset** на панели Query и Filter. Макет шаблона изменяется на вкладке **Layout**. Предварительно просмотреть отчет можно на вкладке **Preview**.

> **Примечание** Вы, конечно, можете использовать построитель отчетов (Report Builder), который имеется на сайте отчетов команды, но этот инструмент не очень хорошо поддерживается сценариями отчетов Visual Studio, поэтому работать с ним не рекомендуется.

Дополнительные ресурсы

- Более подробную информацию вы найдете в разделе «Как настроить отчет в Visual Studio 2005 Team Foundation Server» этой книги.

Как создать новый отчет в Excel

Вы можете создавать пользовательские отчеты, подключив Microsoft Office Excel® напрямую к кубу TFS Reporting OLAP. Excel позволяет отображать данные отчета в форме сводных таблиц или сводных диаграмм.

Создание отчета в форме сводной таблицы Excel

1. Убедитесь, что у вас установлен поставщик Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB Provider, доступный по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc*.
2. Запустите Excel.

3. Выберите электронную таблицу, к которой хотите добавить сводную таблицу.
4. В меню **Data** выберите команду **PivotTable and PivotChart Report**.
5. Выберите **External Data Source.**
6. Щелкните **Next**.
7. Щелкните **Get Data**.
8. Перейдите на вкладку **OLAP Cubes**.
9. Выберите **New Data Source** и щелкните **OK**.
10. Введите имя источника данных.
11. Выберите поставщик **Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB**.
12. Щелкните **Connect**.
13. Выберите **Analysis Server**.
14. Введите имя сервера отчетов, например, TFSRTM.
15. Щелкните **Next**.
16. Выберите **TFSWarehouse** и щелкните **Finish**.
17. Выберите куб, из которого хотите создать отчет (например, Code Churn, Work Items, Test Result) и щелкните **OK**.
18. Еще раз щелкните **OK**, чтобы вернуться в мастер Pivot Table and Pivot Chart Wizard.
19. Щелкните **Finish**, чтобы добавить сводную таблицу на лист.

Перетащите столбцы и измерения в сводную таблицу из списка PivotTable Field List. Ниже приведен пример отображения количества строк для каждого командного проекта на сервере:

1. На шаге 17 выберите куб **Code Churn**.
2. Перетащите **TeamProject.TeamProject** в раздел **Column Fields** сводной таблицы.
3. Перетащите **Total Lines** в раздел **Data Items** сводной таблицы.

Дополнительные ресурсы

- Дополнительную информацию о создании отчетов с помощью Excel вы найдеде в статье «Using Microsoft Excel for Team Foundation Server Reporting» по адресу *http://msdn2.microsoft.com/en-us/library/ms244713(VS.80).aspx*.
- Загрузить Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB Provider можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc*.

Как создать снимок отчета по расписанию

Чтобы лучше понимать тенденции развития проекта, воспользуйтесь плановыми снимками отчета. Кроме того, они позволяют сохранить важные информационные точки проекта.

Плановое создание снимка отчета

1. В окне Team Explorer правой кнопкой щелкните **Reports** и выберите команду **Show Report Site**.
2. Откройте отчет на сайте отчетов.
3. Перейдите на вкладку **Properties**.
4. Щелкните ссылку **History**.
5. Установите расписание для запуска снимка.

После создания расписания вы сможете просматривать отчеты на вкладке **History** данного отчета. Там же можно создавать снимки вручную.

Как подписаться на отчет

Подписки на отчеты используются для генерации отчетов и их экспорта в общий ресурс. Подписки можно настроить на перезапись старых отчетов. Вы также вольны создавать набор отчетов в течение некоторого времени, чтобы просмотреть снимки данных проекта.

Создание подписки на отчет

1. В окне Team Explorer щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. Откройте отчет на сайте отчетов.
3. Перейдите на вкладку **Subscriptions**.
4. Щелкните **New Subscription**, чтобы создать новую подписку.

Как добавить отчет в существующий шаблон процесса

Для добавления новых отчетов в существующий шаблон процесса применяется инструмент Process Editor, поставляющийся с последней версией Team Foundation Server Power Tool. Инструментарий Team Foundation Server Power Tool можно загрузить по адресу from *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.

Добавление нового отчета

1. Загрузите шаблон процесса, наиболее отвечающий вашим требованиям:
   а. В окне Visual Studio щелкните **Team** и выберите **Team Foundation Server Settings**.
   б. Щелкните **Process Template Manager**.

   в. В диалоговом окне **Process Template Manager** выберите шаблон процесса, который хотите изменить, и щелкните **Download**.
   г. В диалоговом окне **Process Template Manager** выберите расположение на локальном диске и щелкните **Save**.
2. Откройте шаблон процесса в окне Process Editor:
   а. В окне Visual Studio раскройте меню **Team**.
   б. Выберите **Process Editor** и щелкните **Open Process Template**.
   в. В диалоговом окне **Open Process Template fileset** перейдите к загруженному шаблону процесса, а затем щелкните **Open**. В окне Visual Studio откроется файл ProcessTemplate.xml.
   г. Заполните поле **Name** (имя) для методологии, к которой вы применяете настройки.
3. В окне Process Template Explorer щелкните **Reports**.
4. На панели инструментов щелкните **Add**.
5. На вкладке **Report Detail** диалогового окна **Report** введите имя отчета.
6. Перейдите в расположение файла .rdl, который хотите добавить в поле **File Name**. Остальные поля оставьте без изменений. Не следует также вносить изменения в данные, содержащиеся на вкладках **Properties** и **Parameters**.
7. На вкладке **DataSources** введите источники данных. Стандартные источники данных для шаблонов процесса, поставляющихся с TFS, — /TfsOlapReportDS и /TfsReportDS.
8. Щелкните **OK**.

Дополнительные ресурсы

- Загрузить инструментарий Team Foundation Server Power Tool можно из источника, расположенного по адресу *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.
- Дополнительную информацию об использовании инструмента Process Editor при настройке типов рабочих элементов вы найдете в разделе «Как настроить шаблон процесса в Visual Studio Team Foundation Server» этой книги.

Просмотр

- Как проанализировать состояние проекта.
- Как проанализировать качество приложения.
- Как просмотреть оставшуюся работу.
- Как просмотреть состояние сборки.
- Как просмотреть ошибки и результаты тестов.

- Как сравнить запланированную работу с фактической.
- Как определить владельца последней редакции файла.
- Как найти все изменения, внесенные в код разработчиком.
- Как найти все изменения, внесенные в файл.
- Как найти все изменения в коде, связанные с конкретным рабочим элементом.
- Как сгенерировать показатели изменяемости кода.
- Как сгенерировать показатели рабочей области (файлов, строки кода, количество проектов).

Как проанализировать состояние проекта

Для анализа состояния проекта используется отчет Velocity. В нем показано, насколько быстро команда справляется с работой и как темп работы изменяется ото дня ко дню.

Просмотр состояния приложения

1. В окне Team Explorer разверните узел проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите отчет **Velocity**.

Как проанализировать качество приложения

Для анализа качества приложения используйте отчет Quality Indicators. В нем собраны результаты, ошибки, данные о покрытии кода тестами и изменяемости кода.

Анализ качества приложения

1. В окне Team Explorer разверните узел проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите отчет **Quality Indicators**.

Как просмотреть оставшуюся работу

Для просмотра оставшейся части работы используется отчет Remaining Work. В нем показано, сколько работ выполнено и закрыто и сколько работы еще предстоит выполнить. Опираясь на эти сведения, вы сможете примерно рассчитать дату завершения работы над кодом.

Просмотр оставшейся части работы

1. В окне Team Explorer разверните узел проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.

2. На сайте отчетов выберите отчет **Remaining Work**.

Как просмотреть состояние сборки

Если вы используете шаблон процесса MSF CMMI, отчет Builds позволит вам просматривать результаты тестов BVT. В этом отчете содержится список имеющихся сборок, а также сведения об их качестве и другая информация.

Просмотр состояния сборки

1. В окне Team Explorer разверните узел проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите отчет **Builds**.

Как просмотреть ошибки и результаты тестов

Для просмотра ошибок используется отчет Bugs by Priority, отображающий соотношение высокоприоритетных и низкоприоритетных ошибок. Отчет Quality Indicators универсален — он применяется для просмотра результатов тестов, ошибок, покрытия кода тестами и изменяемости кода.

Просмотр ошибок и результатов тестов

1. В окне Team Explorer разверните узел проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите отчет **Bugs by Priority** для просмотра ошибок или отчет **Quality Indicators** для просмотра результатов тестов.

Как сравнить запланированную работу с фактической

Для сравнения запланированной и реально выполненной работы используйте отчет Unplanned Work. Он отображает полную работу в сравнении с оставшейся работой, а также отделяет запланированные задачи от внеплановых.

Просмотр отчета Unplanned Work

1. В окне Team Explorer разверните узел проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите отчет **Unplanned Work**.

Как определить владельца последней редакции файла

Для определения владельца последней редакции файла воспользуйтесь историей файла в окне Source Control Explorer.

Определение пользователя, изменившего файл последним

1. В окне Source Control Explorer выберите нужный файл.

2. Щелкните его правой кнопкой мыши и выберите команду **View History**.
3. На панели **History** просмотрите историю изменений, включая их автора.

Дополнительные ресурсы

- Дополнительную информацию о Source Control Explorer вы найдете в статье «Using Source Control Explorer» по адресу *http://msdn2.microsoft.com/en-us/library/ms181370(VS.80).aspx*.

Как найти все изменения, внесенные в код разработчиком

Чтобы найти все изменения, внесенные в код проекта конкретным разработчиком, пользуйтесь командой **TF History**. Вот как выглядит команда, отображающая все изменения, внесенные пользователем Mario:

```
tf history $/ /r /user:Mario
```

Ключ $/ используется для организации поиска по всему хранилищу. Чтобы ограничить область поиска только вашим командным проектом, задайте параметр $/Имя Командного Проекта.

Дополнительные ресурсы

- Дополнительную информацию о команде TF History вы найдете в статье «History Command» по адресу *http://msdn2.microsoft.com/en-us/library/yxtbh4yh(VS.80).aspx*.

Как найти все изменения, внесенные в файл

При помощи истории файла исходного кода можно из окна Source Control Explorer находить изменения, внесенные в файл.

Определение всех изменений, внесенных в файл

1. В окне Source Control Explorer выберите нужный файл.
2. Щелкните его правой кнопкой мыши и выберите команду **View History**.
3. На панели **History** просмотрите историю изменений.

Дополнительные ресурсы

- Дополнительную информацию о Source Control Explorer вы найдете в статье «Using Source Control Explorer» по адресу *http://msdn2.microsoft.com/en-us/library/ms181370(VS.80).aspx*.

Как найти все изменения в коде, связанные с конкретным рабочим элементом

Если во время возврата после правки с набором изменений был связан рабочий элемент, вы сможете просматривать эти изменения на вкладке **Links** рабочего элемента.

Просмотр изменений кода, связанных с рабочим элементом

1. Откройте интересующий вас рабочий элемент.
2. Перейдите на вкладку **Links**. Если с рабочим элементом связан набор изменений, он будет перечислен в списке на панели **Links**.
3. Дважды щелкните набор изменений для просмотра возвращенных файлов и комментариев.

Дополнительные ресурсы

- Дополнительную информацию о наборах изменений вы найдете в статье «Working with Source Control Changesets» по адресу *http://msdn2.microsoft.com/en-us/library/ms181408(VS.80).aspx*.

Как сгенерировать показатели изменяемости кода

Для просмотра подробностей автоматической генерации кода используется отчет Quality Indicators. В нем собраны результаты, ошибки, данные о покрытии кода тестами и изменяемости кода.

Просмотр отчета Quality Indicators

1. В окне Team Explorer разверните узел проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите отчет **Quality Indicators**.

В качестве альтернативы вы можете генерировать отчет об изменяемости кода в Excel. Дополнительные сведения содержатся в разделе «Как создать новый отчет в Excel» этой главы.

Как сгенерировать показатели рабочей области (файлов, строки кода, количество проектов)

Создавайте отчеты по различным показателям рабочей области, подключив Excel напрямую к кубу OLAP TFS Reporting. С помощью Excel отобразите данные отчета в форме сводных таблиц или сводных диаграмм.

Создание сводной таблицы Excel

1. Убедитесь, что у вас установлен поставщик Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB Provider, доступный по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc*.
2. Запустите Excel.
3. Выберите электронную таблицу, к которой хотите добавить сводную таблицу.
4. В меню **Data** выберите команду **PivotTable and PivotChart Report**.

5. Выберите **External Data Source.**
6. Щелкните **Next**.
7. Щелкните **Get Data**.
8. Перейдите на вкладку **OLAP Cubes**.
9. Выберите **New Data Source** и щелкните **OK**.
10. Введите имя источника данных.
11. Выберите поставщик **Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB**.
12. Щелкните **Connect**.
13. Выберите **Analysis Server**.
14. Введите имя сервера отчетов, например, TFSRTM.
15. Щелкните **Next**.
16. Выберите **TFSWarehouse** и щелкните **Finish**.
17. Выберите куб **Code Churn** и щелкните **OK**.
18. Еще раз щелкните **OK**, чтобы вернуться в мастер Pivot Table and Pivot Chart Wizard.
19. Щелкните **Finish**, чтобы добавить сводную таблицу на лист.

При помощи списка **PivotTable Field List** перетащите в сводную таблицу столбцы и меры.

Подсчет файлов в каждом командном проекте

1. Перетащите элемент **TeamProject.TeamProject** в раздел **Page Fields** сводной таблицы.
2. Перетащите **FileName.FilePath** в раздел **Row Fields** сводной таблицы.
3. Для фильтрации по командным проектам используйте раскрывающийся список **Team Project** в разделе **Page Fields**.

Обратите внимание на количество отображенных строк. Это и есть количество файлов.

Подсчет строк в каждом командном проекте

1. Перетащите элемент **TeamProject.TeamProject** в раздел **Column Fields** сводной таблицы.
2. Перетащите **Total Lines** в раздел **Data Items** сводной таблицы.

Подсчет командных проектов, находящихся на сервере

- Перетащите элемент **TeamProject.TeamProject** в раздел **Row Fields** сводной таблицы.

Дополнительные ресурсы

- Дополнительную информацию об использовании Excel для создания специальных отчетов вы найдете в статье «Using Microsoft Excel for Team Foundation Server Reporting» по адресу *http://msdn2.microsoft.com/en-us/library/ms244713(VS.80).aspx*.
- Загрузить Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB Provider можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc*.

Дополнительные ресурсы по отчетам Team Foundation

- Дополнительную информацию об отчетах вы найдете в статье «Team Foundation Server Reporting» по адресу *http://msdn2.microsoft.com/en-us/library/ms194922(VS.80).aspx*.

Практические рекомендации: система управления исходным кодом

В этом разделе

Доступ к системе управления версиями

- Как работать с системой управления версиями на клиентах, работающих не под управлением Visual Studio.
- Как автоматизировать типовые задачи, связанные с управлением версиями.
- Как работать в отсутствие подключения к серверу.

Администрирование

- Как добавить нового разработчика в проект.
- Как удалить покидающего команду разработчика.
- Как предоставлять разрешения в пределах дерева исходного кода.
- Как переместить систему управления версиями Team Foundation Server на другой сервер.

Ветвление, метки и слияние

- Как работать с метками.
- Как выполнять ветвление.
- Как планировать структуру ветвей.
- Как осуществлять поддержку выпуска при помощи ветвления.
- Как осуществлять сопровождение предыдущего выпуска при помощи ветвления.
- Как стабилизировать процесс разработки и сборки при помощи ветвления.
- Как стабилизировать разработку функций при помощи ветвления.

- Как с помощью ветвления стабилизировать параллельную разработку.
- Как с помощью ветвления изолировать внешние зависимости.
- Как прекратить поддержку старого выпуска.
- Как выполнять слияние.
- Как выполнять слияние без основы.
- Как разрешать конфликты слияния.
- Как избегать конфликтов.

Сборки

- Как с помощью TFS производить непрерывную сборку.

Возврат после правки и соответствующие политики

- Как работать с наборами изменений.
- Как обеспечить выполнение стандартов программирования.
- Как перекрыть политику возврата после правки.
- Как отменить возврат после правки.
- Как создать пользовательскую политику возврата после правки.

Отладка, извлечение и блокировка

- Как синхронизировать компьютер с TFS.
- Как подготовить файл к редактированию.

Совместное использование кода

- Как организовать общий доступ к коду.
- Как управлять общими двоичными файлами.

Зависимости

- Как управлять зависимостями веб-служб.
- Как управлять зависимостями БД.

Распределенная и удаленная разработка

- Как получить доступ к TFS через Интернет.
- Как повысить производительность TFS-прокси.

Миграция

- Как осуществить перенос исходного кода с Visual SourceSafe.
- Как осуществить перенос исходного кода из других систем управления версиями.

Управление проектом и рабочей областью

- Как выбрать между созданием одного командного проекта и нескольких.
- Как организовать дерево исходного кода.
- Как определять сопоставления рабочей области.
- Как изолировать изменения кода на компьютере с помощью рабочих областей.

Безопасность

- Как защитить канал между рабочей станцией разработчика и TFS.

Отложенные правки

- Как использовать отложенные правки для создания резервной копии незавершенной работы.
- Как с помощью отложенных правок передать код другому члену команды.

Доступ к системе управления версиями

- Как работать с системой управления версиями на клиентах, работающих не под управлением Visual Studio.
- Как автоматизировать типовые задачи, связанные с управлением версиями.
- Как работать в отсутствие подключения к серверу.

Как работать с системой управления версиями на клиентах, работающих не под управлением Visual Studio

Чтобы получить доступ к системе управления версиями Microsoft® Visual Studio® 2005 Team System (VSTS) Team Foundation Server (TFS), работая на клиентах под управлением других систем, воспользуйтесь следующими способами:

- интеграцией при помощи Microsoft Source Code Control Interface (MSSCCI);
- интеграцией при помощи продуктов сторонних производителей;
- пользовательской интеграцией.

Интеграция при помощи MSSCCI

Ниже приведен список клиентов, способных работать с системой управления версиями TFS через поставщика MSSCCI:

- Microsoft Visual Studio .NET 2003.
- Microsoft Visual C++® 6 SP6.
- Microsoft Visual Basic® 6 SP6.

- Microsoft Visual FoxPro® 9 SP1.
- Microsoft Access™ 2003 SP2.
- Microsoft SQL Server™ Management Studio.
- Sparx Systems Enterprise Architect 61.
- Sybase PowerBuilder 105.
- Toad for SQL Server 2.0.

Загрузить провайдер MSSCCI можно из базы знаний MSDN по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyId=87E1FFBD-A484-4C3A-8776-D560AB1E6198&displaylang=en*.

Интеграция при помощи продуктов сторонних производителей

Для следующих клиентов существуют решения интеграции, предлагаемые независимыми производителями:

- Eclipse.
- Клиент Linux.
- Клиент Apple Macintosh.
- Веб-клиент HTML.

Чтобы получить доступ к системе управления версиями TFS из среды разработки Eclipse, а также из клиентов Linux или Macintosh, установите приложение Teamprise (*http://www.teamprise.com*).

Чтобы получить доступ к системе управления версиями TFS только для чтения через Интернет, воспользуйтесь приложением Team System Web Access (*http://msdn2.microsoft.com/en-us/teamsystem/bb676728.aspx*).

Пользовательская интеграция

На сегодняшний день, решений интеграции для других клиентов не существует. Вы можете получить доступ к TFS либо из командной строки, либо создав собственное интегрирующее решение.

Чтобы автоматизировать работу с командной строкой, используйте управляющие сценарии и командные файлы.

Дополнительные ресурсы

- Дополнительную информацию о работе с управляющими сценариями и командными файлами вы найдете в статье «Team Foundation Source Control Scripts and Command Files» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/1az5ay5c(VS80).aspx*.
- Дополнительную информацию о расширяемости системы TFS Version Control вы найдете в статье «Walkthru: The Version Control Object Model» по адресу *http://msdn2.microsoft.com/en-us/library/bb187335(VS.80).aspx*.
- Дополнительную информацию о работе с системой управления версиями TFS вы найдете в статье Walkthrough: «Working with Team Foundation

Source Control from Command Line» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/zthc5x3f(VS.80).aspx*.

Как автоматизировать типовые задачи, связанные с управлением версиями

Автоматизация наиболее распространенных задач, связанных с управлением версиями, осуществляется с помощью инструмента командной строки tf.exe. Он позволяет выполнять те же действия, что и Source Control Explorer, включая операции управления исходным кодом (**add**, **check-in**, **checkout**, **get**, **lock**, **label** и т. д.), ветвление, создание отложенных правок, манипуляции с рабочей областью и основные административные функции.

Основные плюсы работы с данным инструментом — автоматизация часто повторяющихся действий и возможность создания расписания действий, выполнение которых происходит при наступлении определенных событий, или при помощи планировщика задач Windows. Кроме того, в командной строке доступны следующие действия:

- удаление рабочей области другого пользователя;
- отмена извлечения файлов другим пользователем для редактирования;
- снятие блокировки, установленной другим пользователем;
- определение области видимости метки;
- выполнение слияния без основы.

Чтобы правильно установить пути и других переменные среды, следует запускать инструмент из окна командной строки Visual Studio 2005 или выполнить пакетный файл Vsvars32, который, как правило, расположен в папке *Диск*:\Program Files\Microsoft Visual Studio 8\Common7\Tools.

Инструмент Tf.exe устанавливается в составе клиента TFS и по умолчанию расположен в папке C:\Program Files\Microsoft Visual Studio 8\Common 7\IDE.

При запуске инструмента командной строки следует задать имя сервера при помощи параметра **/s**. Далее приведен пример команды, отображающей файлы в системе управления исходным кодом, расположенной на сервере YourTFSServer:

```
tf.exe dir /s:YourTFSServer
```

Дополнительные ресурсы

- Дополнительную информацию о работе с командной строкой вы найдете в статье «Walkthrough: Working with Team Foundation Source Control from Command Line» на сайте MSDN по адресу http://msdn2.microsoft.com/en-us/library/zthc5x3f(VS.80).aspx
- Справочную информацию о работе с командной строкой вы найдете в статье «MSDN Team Foundation Source Control Command-Line Reference» по адресу *http://msdn2.microsoft.com/en-us/library/cc31bk2e(VS.80).aspx*.

Как работать в отсутствие подключения к серверу

Автономный режим работы системой управления версиями TFS не поддерживается. Чтобы все-таки поработать автономно, вы должны в точности выполнить следующие действия:

1. Вручную снять флаги «только для чтения». По умолчанию все файлы в рабочей области, не извлеченные для правки, доступны только для чтения. При отсутствии подключения к серверу, прежде чем редактировать или удалять файлы, вы должны вручную снять с них флажки «только для чтения». Щелкните файл правой кнопкой в проводнике Windows, выберите команду **Свойства (Properties)**, снимите флажок **Только чтение (Read-only)** и щелкните **OK**. То же действие можно выполнить с помощью команды **attrib -r**.
2. Отредактируйте файлы, с которых сняли метку «только для чтения».
3. **Добавьте или удалите файлы**, с которых сняли метку «только для чтения». Не переименовывайте файлы, потому что инструмент TFTP **online** не способен отличить операцию переименования (rename) от операции удаления (delete) в сочетании с операцией добавления (add).

> **Примечание** Команда **Tfpt online** ищет удаленные файлы только при указании соответствующего параметра, поскольку это довольно продолжительная операция.

4. Запустите команду TFPT online, вернувшись в оперативный режим работы. Для этого нужно ввести в командной строке **TFTP online**. Эта команда проверит рабочую область на предмет наличия записываемых файлов и определит, какие изменения следует отправить на сервер. Если вы удалили какие-либо файлы, задайте параметр **/delete**. Затем инструмент отобразит окно оперативного режима, в котором можно выбрать, какие изменения следует перенести в вашу рабочую область.

> **Важно!** Во время автономной работы нельзя переименовывать файлы.

Дополнительные ресурсы

- Загрузить Team Foundation Power Tools можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?familyid= 7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Дополнительную информацию об инструменте Visual Studio Team Foundation Power Tool вы найдете в статье «Power Toy: tfptexe» по адресу *http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx*.

Администрирование

- Как добавить нового разработчика в проект.
- Как удалить покидающего команду разработчика.
- Как предоставлять разрешения в пределах дерева исходного кода.
- Как переместить систему управления версиями Team Foundation Server на другой сервер.

Как добавить нового разработчика в проект

Чтобы подключить к работе нового разработчика, предоставьте ему доступ к соответствующему командному проекту и связанному с ним сайту Microsoft Office SharePoint®. Чтобы разработчик мог просматривать отчеты, предоставьте учетной записи разработчика доступ к SQL Server Reporting Services.

Предоставление доступа к командному проекту

1. Войдите в систему Visual Studio с учетной записью, входящую в группу администраторов Team Foundation.
2. Добавьте нужный проект в Team Explorer, если его там еще нет.
3. Щелкните правой кнопкой командный проект, раскройте подменю **Team Project Settings** и выберите команду **Group Membership**.
4. Выберите вариант **Project\Contributors**, щелкните **Properties** и добавьте в эту группу учетную запись нового разработчика.

> **Примечание** Члены группы Contributors обладают набором типичных разрешений, требующихся разработчику, включая возможность добавлять, изменять и удалять элементы командного проекта и выполнять сборки.

Предоставление доступа к сайту SharePoint

1. Войдите на сайт командного проекта с учетной записью группы SharePoint Administrator. Сайт проекта ВашПроект по умолчанию располагается по адресу *http://server/sites/ВашПроект/default.aspx*.
2. Щелкните **Site Settings**.
3. Под заголовком **Administration** щелкните **Manage Users**.
4. Щелкните кнопку **Add Users**.
5. Введите имя учетной записи нового разработчика в формате домен\имя_пользователя, выберите группу **Contributor** и щелкните кнопку **Next**.
6. Введите электронный адрес разработчика и, при желании, его приветственное сообщение на сайте.
7. Щелкните кнопку **Finish**.

> **Примечание** Члены группы Contributors обладают набором типичных разрешений, требующихся разработчику, включая возможность добавлять, изменять и удалять элементы командного проекта и выполнять сборки. Если вам нужно ограничить доступ разработчика к некоторым решениям Visual Studio или к некоторым папкам командного проекта, задайте разрешения на уровне папки или файла.

Предоставление доступа к SQL Server Reporting Services

1. Войдите на административный сайт SQL Server Reporting Services с учетной записью администратора. Сайт расположен по адресу *http://server/reports*.
2. Щелкните имя вашего командного проекта.
3. Перейдите на вкладку **Properties**.
4. Перейдите на вкладку **Security**.
5. Щелкните **New Role Assignment**.
6. Введите имя учетной записи Windows вашего разработчика, выберите вариант **Browser** и щелкните **OK**.

> **Примечание** Членство в группе Browser позволяет разработчику просматривать отчеты и подписываться на них.

Дополнительные ресурсы

- Дополнительную информацию о группах, разрешениях и ролях вы найдете в статье «Team Foundation Server Default Groups, Permissions, and Roles» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms253077.aspx*.
- Дополнительную информацию о правах и разрешениях вы найдете в статье «Source Control Security Rights and Permissions» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms181761.aspx*.

Как удалить покидающего команду разработчика

Если разработчик покинул проект, убедитесь, что удалили его рабочую область. Выполнение этой операции не только обеспечивает безопасность проекта, но и приводит к удалению всех незавершенных изменений разработчика и снятию всех установленных им блокировок.

> **Примечание** Если к командному проекту применена монопольная блокировка, вы не сможете снять блокировку, не удалив изменения.

Чтобы выяснить, какие файлы были заблокированы разработчиком, выполните следующую команду:

```
tf workspaces /owner:domain\devuser /computer:* /server:servername
```

Чтобы удалить рабочую область и снять блокировки, выполните следующую команду:

```
tf workspace /delete workspacename;domain\devuser /s:servername
```

Затем удалите учетную запись разработчика из групп безопасности, внеся изменения в три области:

- **Командный проект TFS** Войдите в Visual Studio с учетной записью из группы администраторов Team Foundation. В окне Team Explorer щелкните правой кнопкой нужный проект, раскройте подменю **Team Project Settings**, выберите команду **Group Membership** и удалите ученую запись разработчика из соответствующих групп (как правило, Contributors).
- **Сайт проекта SharePoint** Войдите на сайт команды, расположенный по адресу *http://server/sites/ИмяВашегоПроекта/default.aspx*, с учетной записью администратора. Щелкните **Site Settings**, **Manage Users** и удалите учетную запись разработчика.
- **SQL Server Reporting Services** Войдите на административный сайт SQL Server Reporting Services с учетной записью администратора. Сайт расположен по адресу *http://server/reports*. Щелкните имя командного проекта, перейдите на вкладку **Properties**, затем на вкладку **Security** и удалите учетную запись разработчика.

Дополнительные ресурсы

- Дополнительные сведения о корректном удалении разработчика, выходящего из проекта, вы найдете в статье «How to: Clean Up Files When Users Leave» по адресу *http://msdn2.microsoft.com/en-us/library/ms194958(VS.80).aspx*.
- Дополнительную информацию о команде **Workspace** вы найдете в статье «Workspace Command» по адресу *http://msdn2.microsoft.com/en-us/library/y901w7se(VS.80).aspx*.
- Дополнительную информацию о поиске отложенных правок вы найдете в статье «How do I tell who has files checked out or locked?» по адресу *http://blogs.vertigosoftware.com/teamsystem/archive/2006/07/24/3125.aspx*.

Как предоставлять разрешения в пределах дерева исходного кода

Вы можете предоставлять разрешения в пределах дерева исходного кода. Для этого в обозревателе Source Control щелкните правой кнопкой папку или файл и выберите команду **Properties**. Перейдите на вкладку **Security**, выберите группу пользователей, разрешения которой хотите изменить, и внесите нужные исправления. Можно также установить разрешения с помощью утилиты командной строки tf.exe с параметром **Permissions**.

Вы вольны назначать разрешения на доступ к системе управления исходным кодом для отдельных папок и файлов, однако по умолчанию разрешения внутри дерева исходного кода наследуются от разрешений, примененных к папке проекта. Если разработчики являются членами группы Project\Contributors, они могут читать, изменять, возвращать после правки, присваивать метки и блокировать файлы исходного кода. Чтобы ограничить доступ к подмножеству папок или файлов исходного кода в командном проекте, например, позволить разработчикам работать только над определенными файлами в проекте, задайте разрешения на уровне папки или файла.

Дополнительные ресурсы

- Дополнительную информацию о группах, разрешениях и ролях вы найдете в статье «Team Foundation Server Default Groups, Permissions, and Roles» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms253077.aspx*.
- Дополнительную информацию о правах и разрешениях вы найдете в статье «Source Control Security Rights and Permissions» на сайте MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms181761.aspx*.
- Дополнительную информацию о параметрах команды **tf permission** вы найдете в статье «Permission Command» по адресу *http://msdn2.microsoft.com/en-us/library/0dsd05ft(VS.80).aspx*.

Как переместить систему управления версиями Team Foundation Server на другой сервер

Система Team Foundation Server не поддерживает ни копирование сервера из одного расположения в другое, ни зеркалирование. Вы можете создавать и восстанавливать резервные копии всего сервера, перемещать оборудование сервера в новый домен или выполнить обновление до раздельной системы развертывания. Нельзя осуществлять частичное перемещение, например, переместить одни проекты и оставить другие.

Система Team Foundation Server поддерживает три типа переноса:

- **Восстановление из резервной копии** Этот тип используется для переноса TFS на другой компьютер. Подробнее — в статье «How to: Move Your Team Foundation Server from One Hardware Configuration to Another» по адресу *http://msdn2.microsoft.com/en-us/library/ms404869(VS.80).aspx*.
- **Среда** Этот тип используется при переноса сервера Team Foundation Server в новый домен или рабочую группу. Подробнее — в статье «How to: Move Your Team Foundation Server from One Environment to Another» по адресу *http://msdn2.microsoft.com/en-us/library/ms404883(VS.80).aspx*.
- **Переход с односерверного развертывания на раздельное** Используйте этот тип при переходе от односерверного развертывания к раздельному. Подробнее — в статье «How to: Move from a Single-Server to a Dual-

Server Deployment» по адресу *http://msdn2.microsoft.com/en-us/library/ms404854(VS.80).aspx*.

Перенося Team Foundation Server, учитывайте следующие моменты:

- Если вы изменили имя сервера уровня приложений TFS, все клиенты должны подключаться к нему по новому имени.
- При изменении имени сервера перестанут работать все документы Microsoft Office, связанные с запросами. Документы привязаны к серверу, для которого были созданы. Это относится ко всем документам Microsoft Office, формируемым при помощи запросов и создаваемым автоматически в узле **Documents** во время разработки проекта.
- Если имя сервера изменено, все встроенные ссылки на документы будут указывать на некорректное имя сервера.
- На исходном TFS существовали локальные учетные записи. Вам предстоит решить, как воссоздавать их: как локальные учетные записи на перенесенном сервере TFS или как доменные учетные записи в новом домене перенесенного TFS.
- Допустим, на исходном TFS существовали доменные учетные записи, и вы перемещаете TFS в другой домен, у которого нет доверительных отношений с первоначальным доменом. Решите, что лучше: воссоздать учетные записи на перенесенном TFS как локальные, или создать доменные учетные записи в новом домене перенесенного сервера TFS.

Следует проверить сервер после переноса, убедившись, что во время переноса не произошло серьезных ошибок. Тестирование должно охватывать следующие аспекты:

- Все ли элементы перенесены правильно? Проверьте дерево исходного кода, рабочие элементы и страницы команды.
- Работают ли учетные записи пользователей? Попытайтесь войти в систему с несколькими учетными записями различных типов, чтобы убедиться в их работоспособности.

Дополнительные ресурсы

- Дополнительные сведения о перемещении TFS вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms404879(VS.80).aspx*.

Ветвление, метки и слияние

- Как работать с метками.
- Как выполнять ветвление.
- Как планировать структуру ветвей.
- Как осуществлять поддержку выпуска при помощи ветвления.
- Как осуществлять сопровождение предыдущего выпуска при помощи ветвления.

- Как стабилизировать процесс разработки и сборки при помощи ветвления.
- Как стабилизировать разработку функций при помощи ветвления.
- Как с помощью ветвления стабилизировать параллельную разработку.
- Как с помощью ветвления изолировать внешние зависимости.
- Как прекратить поддержку старого выпуска.
- Как выполнять слияние.
- Как выполнять слияние без основы.
- Как разрешать конфликты слияния.
- Как избегать конфликтов.

Как работать с метками

При помощи меток вы объединяете файлы и папки в наборы для проведения операций над ними в будущем. Метки используются для ветвления, слияния, сравнения или получения файлов. Метка представляет собой маркер, к которому можно вернуться позднее при выполнении перечисленных выше операций.

Присвоение метки файлу или папке

1. Щелкните правой кнопкой файл или папку в окне обозревателя Source Control и выберите команду **Apply Label**.
2. В диалоговом окне **Choose Item Version** уточните имя файла или папки, выберите версию файла или папки, которую хотите пометить, и щелкните **OK**, чтобы применить метку.

При применении меток следует учитывать следующее:

- Метку можно применить только к одной версии файла или папки.
- Одной версии файла можно присвоить несколько меток.
- Метки, присваиваемые в Source Control Explorer, автоматически видны в корневой папке проекта, внутри которого они были созданы. Нельзя создать две метки с одинаковыми именами в одной зоне видимости.
- Метки не имеют версий, и с ними не связанно никакой истории.
- Применение меток происходит мгновенно, они не требуют возврата после правки.
- Система Team Build автоматически присваивает метки набору файлов, задействованному в любой создаваемой ею сборке.
- Метки не применяются к удаляемым объектам. Это означает, что при слиянии на основе меток не будет выполнен перенос удаляемых файлов.

Поиск существующей метки

1. В меню **File** откройте подменю **Source Control**, затем выберите команду **Label**, щелкните **Find Label** и перейдите в расположение метки.

2. Найдя метку, вы можете в диалоговом окне **Find Label** изменить или удалить ее.

Дополнительные ресурсы

- Дополнительную информацию об использовании меток вы найдете в статье «Working with Labels» по адресу *http://msdn2.microsoft.com/en-us/library/ms181439(VS.80).aspx*.
- Дополнительную информацию о применении меток вы найдете в статье «How to: Apply Labels» по адресу *http://msdn2.microsoft.com/en-us/library/ms181440(VS.80).aspx*.

Как выполнять ветвление

Чтобы создать ветвь, используйте Source Control Explorer или команду **tf branch** из командной строки.

Для реализации ветвления из Source Control Explorer щелкните правой кнопкой папку самого высокого уровня с исходным кодом вашего проекта, выберите команду **Branch** и укажите расположение и имя конечной папки, поясняющее назначение ветви, например, **MyProject_Release1.0_Branch**.

Чтобы выполнить ветвление из командной строки Visual Studio 2005, используйте команду **tf branch**, например:

```
tf branch C:\MyProject $/MyProject_Release1.0_Branch
```

Ветви следует использовать, только если вам нужно изолировать параллельную разработку. При этом вам придется периодически переносить изменения в основную ветвь, что влечет за собой дополнительные расходы и требует разрешения конфликтов. Не выполняйте ветвление без необходимости.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как планировать структуру ветвей

Ветви — механизм изоляции. Они используются, чтобы позволить нескольким пользователям работать над одними и теми же файлами одновременно. При разработке способа ветвления следует рассмотреть типичные сценарии и избрать стратегию, опираясь на размер и структуру команды, периодичность выпусков и требования к стабильности сборок.

Ниже приведены типовые сценарии ветвления:

- **Ветвь выпуска** Стабилизация кода выпуска. Ветвление стоит провести перед выпуском, избежав блокировки основной ветви.
- **Ветвь обслуживания** Обслуживание выпущенной ранее сборки.
- **Ветвь функции** Ветвление для изолированной разработки функций, способной привести к нестабильной работе всего проекта.
- **Ветвь группы** Ветвление для изоляции подгруппы, члены которой должны работать обособленно, не испытывая влияния крупных изменений или двигаясь к собственным вехам. Такая ветвь похожа на ветвь функции.

Не выполняйте ветвление без необходимости, поскольку оно добавит работы по обслуживанию дополнительного дерева исходного кода и по слиянию. Большинство команд разработчиков, создающих коммерческие приложения (LOB), работают по короткому циклу выпуска и не нуждаются в ветвлении. Команды разработчиков, работающие с более продолжительными циклами, например, независимые поставщики ПО, нуждаются в ветвлении чаще.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как осуществлять поддержку выпуска при помощи ветвления

Перед выпуском, когда вы готовы стабилизировать сборку, создайте ветвь

выпуска.

После ветвления структура папок будет выглядеть примерно так:

- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.
- **Releases** — контейнер для ветвей выпусков.
  - **Release 1** — ветвь выпуска.
    - **Source**.

Учитывайте следующие рекомендации по работе с ветвью выпуска:

- **Когда выполнять ветвление** Когда вы готовы к выпуску, соберите все в главную ветвь (Main), а затем создайте ветвь выпуска (Release), целью которой будет стабилизация приложения перед выпуском.
- **Когда не следует выполнять ветвление** Если каждому выпуску соответствует отдельный проект TFS, вы можете продолжать разработку, создав новый проект, не выполняя ветвление текущего проекта.
- **Разрешения в ветви**
  - **До выпуска** Предоставьте разрешения на чтение и запись всем разработчикам.
  - **После выпуска** Предоставьте разрешения на чтение и запись разработчикам, принимающим участие в работе над исправлениями. Остальным достаточно разрешения на чтение.
- **Частота производства сборок в ветви** Сборки производятся по мере необходимости.
- **Тесты в ветви** Прекращаются после выпуска.

Ветвь Release применяется для внесения конкретных исправлений и изменений, требующихся для стабилизации сборки перед выпуском. Параллельно в ветвях Development или Main может продолжаться разработка следующих версий приложения. В них также могут понадобиться стабилизирующие изменения, внесенные вами в ветви Release. После создания окончательной сборки выпуска, перенесите изменения из ветви Release в ветви Development или Main.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

- Дополнительную информауию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как осуществлять сопровождение предыдущего выпуска при помощи ветвления

Для поддержки ранее выпущенных сборок используйте ветви сопровождения. Они очень похожи на ветви выпусков, но существуют более продолжительное время.

Примерно так будет выглядеть структура ветвей, после того как вы выпустите приложение и создадите ветвь для его сопровождения:

- **Main** — Главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.
- **Releases** — контейнер для ветвей выпусков.
  - **Release 1** — ветвь сопровождения.
    - **Source**.

    Другие папки ресурсов.

Учитывайте следующие рекомендации по работе с ветвью сопровождения:

- **Когда выполнять ветвление** После выпуска.
- **Когда не следует выполнять ветвление** Если сопровождение выпуска не требуется, обозначайте сборки предыдущих выпусков с помощью меток и продолжайте работать в основной ветви.
- **Разрешения на доступ к ветви** Предоставьте разрешения на чтение и запись разработчикам, работающим над исправлениями, остальным — разрешения только на чтение.
- **Частота производства сборок в ветви** Сборки производятся по мере необходимости.
- **Тесты в ветви** Не проводятся.

Ветви сопровождения используются для поддержки предыдущих версий приложения. Вы можете перенести изменения в главную ветвь сборки или оставить их исключительно в ветви сопровождения.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.

- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как стабилизировать процесс разработки и сборки при помощи ветвления

Во избежание ошибок сборки проводите активную разработку в ветви Development, а общую сборку — в ветви Main. Примерно так будет выглядеть структура ветвей после создания ветви разработки Development:

- **Development** — ветвь разработки.
  - **Source**.
- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.

Учитывайте следующие рекомендации по работе с ветвью разработки:

- **Когда выполнять ветвление** При ежедневном создании сборок и наличии проблем со стабилизацией и интеграцией сборок создайте две ветви — Main и Development, чтобы сделать ежедневные сборки более предсказуемыми. Стоит также подумать об ужесточении политики возврата после правки.
- **Когда не следует выполнять ветвление** Если вы используете только непрерывную сборку и ежедневные сборки достаточно стабильны, нет большого смысла нести дополнительные расходы, связанные с ветвью разработки.
- **Разрешения на доступ к ветви**
  - Ветвь Main должна быть доступна для чтения и записи разработчикам, отвечающим за слияние и сборку. Остальные получают разрешения только на чтение.
  - Ветвь Development должна быть доступна всем для чтения и записи.
- **Частота производства сборок в ветви**
  - В ветви Main — ежедневно.
  - В ветви Development — непрерывная сборка.
- **Тесты в ветви**
  - В ветви Main проводятся испытания целостности, производительности и безопасности.

- В ветви Development проводятся беглое тестирование, а также испытания функций.

Используйте ветвь Main для интеграции изменений, внесенных в ветви разработки. В ветви Development следует выполнять всю активную разработку с последующим переносом в ветвь Main изменений, не приводящих к ошибкам сборки.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как стабилизировать разработку функций при помощи ветвления

Используйте ветвь функций для стабилизации разработки отдельных компонентов и функций.

После создания ветвей функций структура папок будет выглядеть примерно так:

- **Development** — контейнер для ветвей функций.
  - **Функция А** — ветвь функции.
    - **Source**.
  - **Функция Б** — ветвь функции.
    - **Source**.

    Функция В — ветвь функции.
  - **Source**.
- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.

Учитывайте следующие рекомендации по работе с ветвью функции:

- **Когда выполнять ветвление** Файлы, с которыми работают разработчики различных функций, часто перекрываются, что может привести к ошибкам и конфликтам при сборке и возврате после правки. Если у вас возникают подобные проблемы, обдумайте возможность ветвления по каждой функции, чтобы обеспечить ее изоляцию. Разветвить можно папку Main или папки отдельных групп (в больших проектах).
- **Когда не следует выполнять ветвление** Если вы используете только непрерывную сборку и ежедневные сборки достаточно стабильны, нет большого смысла нести дополнительные расходы, связанные с ветвью разработки.
- **Разрешения на доступ к ветви** Предоставьте разрешения на чтение и запись разработчикам, работающим над функцией в данной ветви, а всем остальным — разрешения только на чтение.
- **Частота производства сборок в ветви** В этой ветви производится непрерывная сборка.
- **Тесты в ветви** Производятся испытания функций и беглое тестирование сборки.

Ветвление позволяет вести разработку функций параллельно. При этом вся активная разработка выполняется в ветвях функций, а последующая интеграция кода — в ветви Main.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как с помощью ветвления стабилизировать параллельную разработку

Используйте ветви команд для изоляции критических ошибок в командах.

После создания ветвей команд структура папок будет выглядеть примерно так:

- **Development** — контейнер для ветвей команд.
  - **Team 1** — ветвь команды.
    - **Source**.
  - **Team 2** — ветвь команды.
    - **Source**.
- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.

Учитывайте следующие рекомендации по работе с ветвями команд:

- **Когда выполнять ветвление** Если файлы и компоненты, с которыми работают разные команды, перекрываются, используйте ветви команд для изоляции их работы. Внутри ветвей команд можно создать ветви функций.
- **Когда не следует выполнять ветвление** Если дерево исходного кода упорядочено по компонентам и вы уверены, что не будет ни крупных изменений в интерфейсе, ни большого количества конфликтов, возможно, создавать ветви команд нет необходимости.
- **Разрешения на доступ к ветви** Предоставьте разрешения на чтение и запись членам команды, остальным — разрешения только на чтение.
- **Частота производства сборок в ветви** Производится непрерывная сборка.
- **Тесты в ветви** Производятся испытания функций и беглое тестирование сборки.

Не пренебрегайте ветвлением команд, чтобы команды могли работать над своими задачами одновременно. Ветви помогают изолировать ошибки команды, а также позволяют командам двигаться к различным вехам. Вся активная разработка должна проводиться в ветвях команд с последующим переносом в главную ветвь (Main). Внутри ветвей команд можно также создавать ветви функций.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как с помощью ветвления изолировать внешние зависимости

Используйте ветви внешних зависимостей для изоляции критических ошибок во внешних зависимостях.

После создания ветвей внешних зависимостей структура папок может выглядеть примерно так:

- **External** — ветвь внешней зависимости.
  - **Source**.
- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.

Учитывайте следующие рекомендации по работе с ветвью внешней зависимости:

- **Когда выполнять ветвление** Если у вас есть внешняя зависимость, способная вызвать критические ошибки в проекте, а также связанная с внесением изменений в интерфейс или логику кода, создайте ветвь внешней зависимости (External) для изоляции этих изменений.
- **Когда не следует выполнять ветвление** Если внешние зависимости стабильны и вы уверены, что они не приведут к критическим изменениям.
- **Разрешения на доступ к ветви** Предоставьте разрешение на чтение и запись разработчикам, отвечающим за интеграцию внешней зависимости. Остальным достаточно разрешения только на чтение.
- **Частота производства сборок в ветви** Сборки производятся по мере необходимости.
- **Тесты в ветви** Проверка взаимодействия и функционирования компонентов.

Ветвь внешней зависимости следует использовать для экспериментальных изменений во внешних зависимостях перед переносом этих изменений в ветвь Main. Стоит также изолировать команду разработчиков от критических ошибок, вызванных внешними зависимостями, например, заголовочными файлами или библиотеками.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.

- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как прекратить поддержку старого выпуска

Создайте архивную папку для хранения старых выпусков, которые уже нет смысла поддерживать. После создания архивной ветви структура папок будет выглядеть примерно так:

- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.
- **Releases** — контейнер для ветвей выпусков.
  - **Release 2** — ветвь сопровождения.
    - **Source**.
    - **Другие папки ресурсов**.
- **Архив** — контейнер для архивного хранения ветвей.
  - **Release 1** — архивная ветвь.
    - **Source**.

    Другие папки ресурсов.

Перемещая ветви из папки Releases в архив, вы разгружаете папку Releases и одновременно сохраняете старые выпуски. Это не создание новой ветви, а, скорее, перемещение старой ветви в новую папку.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

- Дополнительную информацию о методах ветвления и слияния в Visual Studio 2005 вы найдете в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как выполнять слияние

Слияние заключается в переносе изменений из одной ветви в другую. Его можно выполнять, используя возможности обозревателя Source Control или команду **tf merge**. Слияние выполняется по набору изменений, метке, дате или версии. Чтобы приступить к слиянию, щелкните правой кнопкой ветвь в Source Control и выберите команду **Merge**. Мастер Source Control Merge Wizard поможет выбрать целевую ветвь (в которую будет выполнено слияние).

В зависимости от структуры ветвей изменения можно переносить вверх по иерархии, вниз по иерархии или поперек иерархии. При поперечном слиянии выполняется слияние без основы. Для выполнения последнего вам придется воспользоваться командой **tf merge**, поскольку слияние без основы в Visual Studio не поддерживается. Слияние без основы позволяет переносить файлы, не имеющих связей по ветви или слиянию. После проведения слияния без основы необходимые связи устанавливаются, и следующие слияния уже будут иметь основу. Вам по-прежнему придется выполнять их из командной строки, однако число конфликтов слияния сократится.

Имейте в виду, что слияние вдоль иерархии — от родительской к дочерней ветви или от дочерней к родительской ветви — завершается с меньшим количеством конфликтов, чем слияние поперек иерархии. Иерархия ветвей основана на родительских и дочерних ветвях и может отличаться от физической структуры, которую вы видите в Source Control. Например:

Физическая структура.

- **Development** — ветвь разработки.
- **Main** — главная ветвь сборки.
- **Releases** — контейнер для ветвей выпусков.
  - **Release 1** — ветвь выпуска.
    - **Логическая структура**.
- **Main**.
- **Development**.
  - **Release 1**.

Дополнительные ресурсы

- Дополнительную информацию о слиянии вы найдете в статье «Understanding Merging» по адресу *http://msdn2.microsoft.com/en-us/library/ms181427(VS.80).aspx*.

- О том, как выполнить слияние, читайте в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

Как выполнять слияние без основы

Слияние без основы производится при помощи команды **tf merge /baseless** из командной строки Visual Studio 2005.

В следующем примере команды выполняется слияние без основы из исходной ветви в целевую. Параметр **/recursive** используется для рекурсивного слияния всех файлов и папок, находящихся в указанной папке:

```
merge /baseless <<путь_к_источнику>> <<конечный_путь>> /recursive
```

Например,

```
tf merge /baseless c:\data\proj1 c:\data proj2 /recursive
```

Процесс переноса элементов ветвей, не являющихся прямыми ветвями друг друга, называется *слиянием без основы* (baseless merge). Слиянием без основы можно считать слияние изменений между двумя ветвями выпусков, находящимися на одном уровне, но не имеющих общей родительской ветви. Слияние без основы выполнимо только из командной строки; из Visual Studio его осуществить нельзя.

Когда выполняется слияние без основы, у TFS нет никакой информации о связи между файлами в ветвях. Например, переименование файла будет расценено, как удаление из ветви одного файла и появление в ней другого. Поэтому вам придется разрешать вручную больше конфликтов, чем при обычном слиянии. Тем не менее, с подобным разрешением конфликтов вы столкнетесь только один раз. В процессе слияния без основы TFS записывает его историю и устанавливает связи между папками и файлами. При этом последующие слияния все равно должны будут выполняться из командной строки.

Дополнительные ресурсы

- Описание синтаксиса команды **merge** вы найдете в статье «Merge Command» по адресу *http://msdn2.microsoft.com/en-us/library/bd6dxhfy(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «Understanding Merging» по адресу *http://msdn2.microsoft.com/en-us/library/ms181427(VS.80).aspx*.
- О том, как выполнить слияние, читайте в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

Как разрешать конфликты слияния

Для разрешения конфликтов слияния используйте инструментарий слияния Visual Studio. Обнаружив конфликт в процессе слияния, вы можете раз-

решить его автоматически или вручную. Разрешая конфликт вручную, вы вольны сохранить изменения из исходной ветви, сохранить изменения из целевой ветви или разрешить конфликт при помощи инструмента слияния.

Необходимость в разрешении конфликтов возникает при выполнении слияния ветвей, извлечении файлов в рабочую область или возврате новых версий файлов. Существуют три типа конфликтов:

- **Версии** Файл эволюционировал по нескольким различным траекториям. Это может быть результатом редактирования, переименования, удаления и отмены удаления файла.
- **Неоднозначность имени файла** Два или несколько элементов пытаются занять одно расположение.
- **Локальная перезапись** Возникает только во время выполнения операции **get** при попытке перезаписать редактируемый файл.

Большинство конфликтов может быть разрешено автоматически. Личного вмешательства требует только конфликт версий. Чаще всего ручное разрешение конфликтов происходит в следующих сценариях:

- В обеих ветвях было выполнено редактирование файла, затронувшее одинаковые строки кода.
- Проводится слияние без основы, когда TFS не известны связи файлов ветвей.

Инструментарий слияния отображает подробности каждого конфликта и позволяет выбрать изменения, которые следует сохранить после слияния. Вы можете сохранить изменения источника или целевого файла, объединить изменения или вручную модифицировать окончательную версию, осуществив ввод непосредственно в файл.

Разрешив все конфликты в файле, сохраните окончательную версию как незавершенное изменение в целевой ветви.

Будьте осторожны во время слияния: при этом легко допустить ошибки, которые приведут к нестабильности сборки. Завершив слияние, скомпилируйте получившийся исходный код и выполните модульные тесты, чтобы убедиться в отсутствии серьезных ошибок.

Дополнительные ресурсы

- Дополнительную информацию о разрешении конфликтов вы найдете в статье «How to: Resolve Conflicts» по адресу *http://msdn2.microsoft.com/en-us/library/ms181433(VS.80).aspx*.

Как избегать конфликтов

Во избежание конфликтов, сделайте следующее:

- **Обеспечьте нормальную связь внутри команды** Работая над файлом исходного кода, проинформируйте об этом других членов команды. Также сообщите им о том, что будете исправлять. Хотя многие конфлик-

ты можно разрешить автоматически, следует избегать ситуаций, при которых два или несколько разработчиков работают в одной и той же функциональной области одного и того же файла. При этом есть большая вероятность того, что изменения будут внесены в одни и те же строки кода. Конфликты, касающиеся одинаковых строк кода, требуют разрешения вручную, что усложняет выполнение слияния.

- **Узнайте, кто еще извлек файл для редактирования** Прежде чем обновить файл, проверьте, не извлек ли его кто-либо еще. Если это так, спросите вашего коллегу, над чем он работает, и примите решение, следует ли вам подождать, пока он не завершит свои изменения, или вы можете продолжать работу одновременно с ним, поскольку работаете над разными функциями в разных частях файла.

Просмотр незавершенных изменений

1. В окне Source Control щелкните правой кнопкой решение, проект, папку или файл, для которых хотите просмотреть незавершенные изменения.
2. Выберите команду **View Pending Changes**.

Этот способ позволяет просмотреть все незавершенные изменения в выбранной области. Кроме того, узнать об отложенных изменениях можно, воспользовавшись инструментом командной строки, например:

```
tf status /format:detailed /user:*
```

Эта команда отображает подробную информацию о статусе всех незавершенных изменений, внесенных всеми пользователями. В выводимом списке наряду с изменениями указано, кто и какие файлы извлек.

Дополнительные ресурсы

- Дополнительную информацию о просмотре незавершенных изменений в своей рабочей области вы найдете в статье «How to: View and Manage All Pending Changes in Your Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181400(VS.80).aspx*.
- Дополнительную информацию о просмотре незавершенных изменений в других рабочих областях вы найдете в статье «How to: View Pending Changes in Other Workspaces» по адресу *http://msdn2.microsoft.com/en-us/library/ms181401(VS.80).aspx*.
- Дополнительную информацию о команде **tf status** вы найдете в статье «Status Command» по адресу *http://msdn2.microsoft.com/en-us/library/9s5ae285(VS.80).aspx*.

Сборки

- Как с помощью TFS производить непрерывную сборку.

Как с помощью TFS производить непрерывную сборку

Чтобы проводить сборку непрерывно, осуществляя ее при каждом возврате файла после правки, вам понадобится веб-служба, запускающая процесс сборки. Вы подписываете веб-службу на события возврата после правки, чтобы сборка запускалась каждый раз, когда происходит такое событие. Чтобы запустить сборку, вам также понадобится подходящий тип сборки, в котором определена конфигурация сборки, действия после сборки, выполняемые тесты, расположение, используемое для выгрузки и т. д.

Дополнительные ресурсы

- Загрузить веб-службу для запуска процесса сборки, разработанную в Майкрософт, можно из источника, расположенного по адресу *http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi*.

Возврат после правки и соответствующие политики

- Как работать с наборами изменений.
- Как обеспечить выполнение стандартов программирования.
- Как перекрыть политику возврата после правки.
- Как отменить возврат после правки.
- Как создать пользовательскую политику возврата после правки.

Как работать с наборами изменений

Набор изменений (changeset) представляет собой совокупность изменений, связанных с конкретным возвратом после правки. Вот список наиболее распространенных действий, применимых к наборам изменений:

- возврат набора изменений, связанного с рабочим элементом;
- задание метки набора изменений;
- просмотр свойств набора изменений;
- изменение свойств набора изменений;
- отмена набора изменений.

Возврат набора изменений, связанного с рабочим элементом

1. В Visual Studio откройте меню **View**, раскройте подменю **Other Windows** и выберите команду **Pending Changes**.
2. Введите комментарий, поясняющий характер изменений.
3. Щелкните значок **Work Items**, чтобы раскрыть список рабочих элементов, связанных с набором.

4. Обновите список рабочих элементов: выделите элемент и укажите нужное действие— **Associate** или **Resolve** (если возврат после правки подразумевает разрешение рабочего элемента).
5. Щелкните **Check In**, чтобы вернуть набор изменений на сервер управления исходным кодом.

Задание метки набора изменений

1. В окне обозревателя Source Control щелкните правой кнопкой папку с командным проектом и выберите команду **Apply Label**.
2. В раскрывающемся списке **Version** выберите вариант **Changeset**, введите номер в поле **Changeset number** и щелкните **OK**.
3. В диалоговом окне **Apply Label** введите имя метки и комментарий. Щелкните **OK**.

Просмотр свойств набора изменений

Производится с помощью команды **tf changeset**. Далее приведен пример команды, отображающей в диалоговом окне **Details for Changeset** свойства набора изменений под номером 1234:

```
tf changeset 1234
```

В этом диалоговом окне вы можете просмотреть файлы исходного кода, содержащиеся в наборе изменений, а также комментарии и примечания. Кроме того, можно просмотреть связанные с набором рабочие элементы и все предупреждения, сгенерированные политиками во время возврата набора изменений на сервер.

Изменение свойств набора изменений

Используйте команду **tf changeset**, чтобы изменить комментарии и примечания, связанные с набором изменений. Приведенная ниже команда вызывает диалоговое окно **Details for Changeset** со свойствами набора под номером 1234 и обновляет поле комментария.

```
tf changeset /comment:"Этот комментарий гораздо лучше предыдущего." 1234
```

Далее приведен пример команды, которая обновляет примечания с именами экспертов по коду и безопасности, связанных с набором изменений 1234.

```
tf changeset /notes:"Code Reviewer"="C Davis";"Security Reviewer"="F Smith"
1234
```

Отмена набора изменений

Чтобы откатить набор изменений и удалить его с сервера управления исходным кодом, используется команда **Tfpt rollback** из комплекта Team Foundation Power Tool. В следующем примере производится откат набора изменений под номером 1234.

```
TFPT rollback /changeset:1234
```

Эта команда открывает окно **Roll Back Changeset**, в котором можно выбрать файлы из набора изменений для отката.

Дополнительные ресурсы

- Дополнительную информацию о команде **tf changeset** вы найдете в статье «Changeset Command» по адресу *http://msdn2.microsoft.com/en-us/library/w51xa47k(VS.80).aspx*.
- Дополнительную информацию об извлечении наборов изменений вы найдете в статье «How to: Retrieve Old Versions of Files from Changesets» по адресу *http://msdn2.microsoft.com/en-us/library/ms181416(VS.80).aspx*.
- Загрузить инструмент Team Foundation Power Tool можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Дополнительные сведения об инструменте Team Foundation Power Tool вы найдете в статье «Power Toy: tfptexe» по адресу *http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx*.

Как обеспечить выполнение стандартов программирования

Чтобы обеспечить соответствие всего возвращаемого на сервер кода заданным стандартам программирования, используйте политики возврата после правки.

По умолчанию, доступны следующие политики:

- **Code Analysis** Требует выполнения анализа кода перед возвратом.
- **Test Policy** Требует проведения тестов перед возвратом.
- **Work Items** Требует, чтобы с возвратом был связан один или несколько рабочих элементов.

По умолчанию политика Code Analysis обеспечивает проведение проверки как управляемого, так и неуправляемого кода. В управляемом коде статически анализируется соответствие стандартным правилам проектирования, глобализации, интероперабельности, присваивания имен, производительности, безопасности и т. д. Чтобы углубить анализ кода, разработайте собственные правила.

Чтобы настроить политику анализа кода при возврате после правки, выполните следующие действия:

1. В окне Team Explorer щелкните правой кнопкой ваш командный проект, раскройте подменю **Team Project Settings** и выберите команду **Source Control**.
2. Щелкните **Check-in Policy** и **Add**.
3. В списке **Check-in Policy** выберите вариант **Code Analysis** и щелкните **OK**.

4. Укажите нужный тип анализа кода, установив соответствующий флажок. При выборе варианта **Enforce Code Analysis For Managed Code** укажите требуемые правила в списке **Rule settings for Managed Code Analysis**.
5. Дважды щелкните **OK**.

> **Важно!** Хотя описанная процедура обеспечивает применение настроенной политики при каждом возврате файла исходного кода после правки, разработчики всегда могут перекрыть политику. Чтобы отслеживать перекрытие политики, следите за соответствующими событиями.

Вы вольны создать собственные политики возврата после правки, позволяющие задавать определенные стандарты качества в зависимости от конкретного проекта, например:

- Обязательное добавление пользователями комментариев при возврате.
- Перед возвращением кода пользователи должны проводить дополнительные тесты.
- Пользователи не используют определенные директивы C#, чтобы подавить предупреждения документации XML.
- Проекты настроены так, чтобы во время компиляции генерировалась XML-документация.

Чтобы создать надстройки пользовательских политик, которые будут отображаться в диалоговом окне **Add Checkin Policy**, используйте функции расширяемости из комплекта Visual Studio Team Foundation Server Software Development Kit (SDK).

Дополнительные ресурсы

- Дополнительную информацию о создании и использовании пользовательской политики возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о настройке политики возврата после правки вы найдете в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- С примером кода, запрещающего определенные варианты программирования, можно ознакомиться по материалам статьи «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- С примером кода, заставляющего добавлять комментарии при возврате после правки, можно ознакомиться в статье «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.

- Дополнительную информацию о том, как зарегистрировать новую политику возврата после правки, вы найдете в статье «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.
- Дополнительную информацию об использовании инструментов анализа кода вы найдете в статье «Guidelines for Using Code Analysis Tools» по адресу *http://msdn2.microsoft.com/en-us/library/ms182023(VS.80).aspx*.

Как перекрыть политику возврата после правки

Чтобы перекрыть политику возврата после правки, задайте параметр **Override policy failure and continue check-in** в диалоговом окне **Policy Failure**. Перекрыть политику возврата после правки волен любой пользователь, обладающий разрешением на возврат файлов.

Чтобы проследить за перекрытием политики возврата после правки, воспользуйтесь службой событий Team Foundation Eventing Service.

Дополнительные ресурсы

- Дополнительную информацию о перекрытии политики возврата после правки вы найдете в статье «How to: Override a Check-in Policy» по адресу *http://msdn2.microsoft.com/en-us/library/ms245460(VS.80).aspx*.
- Дополнительную информацию о службе Team Foundation Eventing Service вы найдете в статье «Eventing Service» по адресу *http://msdn2.microsoft.com/en-us/library/bb130154(vs.80).aspx*.
- О том, как обнаружить перекрытие политики с помощью объектной модели TFS, не используя службу событий, можно прочитать в блоге Джеймса Мэннинга (James Manning) по адресу *http://blogs.msdn.com/jmanning/archive/2005/11/04/489193.aspx*.

Как отменить возврат после правки

Для отмены возврата файла используется команда **rollback** из комплекта Team Foundation Power Tools. Эта команда возвращает файл к его предыдущей версии. Команда **rollback** позволяет выполнить откат сразу всего набора изменений, но можно также выбирать для отката лишь некоторые файлы из набора. Это очень удобно, когда нужно отменить ошибочно возвращенное изменение файла или возвращенные изменения привели к серьезным конфликтам сборки.

Отмена возврата файла

1. Запустите приведенную ниже команду в окне командной строки. Переменная PATH должна включать путь \Program Files\Microsoft Team Founda-tion Server Power Tools.

```
TFPT rollback filename.cs
```

> **Примечание** Если вам известен номер набора изменений, содержащего правку, которую вы хотите отменить, укажите его в команде, как показано ниже:

```
TFPT rollback filename.cs /changeset:54
```

2. Команда **rollback** запросит подтверждение на обновление рабочей области. Щелкните кнопку **Yes** в информационном окне **Roll Back Changeset**. После этого в рабочую область будут переданы файлы с сервера.
3. Если вы не указали номер набора изменений в командной строке, откроется диалоговое окно **Find Changeset**. Введите критерий поиска или просто щелкните кнопку **Find**. Найдите и выделите набор изменений, содержащий правку, которую вы хотите отменить, и щелкните **Roll Back**. Откроется окно **Roll Back Changeset**.
4. Поскольку в наборе может содержаться несколько изменений, выделите файл, отмену возврата которого хотите выполнить, и щелкните **Roll Back**.

> **Примечание** Если имя файла указать в командной строке, то из всего набора будет выбран только он.

Отменяя возврат после правки командой **TFPT rollback**, учитывайте следующие соображения:

- **Расположение рабочей области** TFPT определяет расположение рабочих областей следующими способами. Если вы указали путь к файлу в качестве аргумента, для поиска рабочей области используется он. Если вы не указали путь к файлам, в качестве рабочей области используется локальная папка, если для нее есть сопоставление. Чтобы гарантировать, что инструмент будет работать в нужной рабочей области, запустите команду из локально сопоставленной папки.
- **Незавершенные изменения** Нельзя откатить набор, содержащий незавершенные изменения. При попытке сделать это вы получите сообщение об ошибке. Перед запуском команды **rollback** отложите (shelve) незавершенные изменения, которые хотите сохранить, а остальные отмените или запишите на сервер.
- **Слияния** Если вы отменяете возврат файла, который был произведен совсем недавно, вам, вероятно, не придется выполнять перенос изменений: маловероятно, что кто-нибудь уже успел обновить элемент. Если же вы хотите отменить возврат, который не является последним изменением файла, вам потребуется трехстороннее слияние. Нужно будет объединить текущую версию на сервере, версию в вашей рабочей области и версию, которую вы хотите откатить. При возникновении конфликтов в файле удаляются изменения из версии, предназначенной для отката. Любые изменения, внесенные после этой версии, сохраняются.

- **Разрешение конфликтов** Если в процессе слияния возник конфликт, на экране появится специальное окно. Чтобы все-таки выполнить слияние, выделите элемент и щелкните кнопку **Merge**. Первоначально предпринимается попытка автоматического слияния. В случае неудачи для разрешения конфликта вызывается инструмент слияния. Если вы щелкнете кнопку **Auto-Merge All**, будет произведена попытка выполнить автоматическое слияние всех элементов, находящихся в списке слияния. Инструмент слияний не вызывается.

Дополнительные ресурсы

- Загрузить инструмент TFPT можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Дополнительную информацию об инструменте TFPT вы найдете в статье «Power Toy: tfptexe» по адресу *http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx*.

Как создать пользовательскую политику возврата после правки

Для создания пользовательской политики возврата после правки используется модель надстройки, предоставленная средой политики (policy framework).

Пользовательская политика создается для реализации ваших собственных правил, например, чтобы все пользователи при возврате после правки добавляли комментарии или соответствующим образом использовали регулярные выражения.

Надстройки используются как в процессе определения политики, так и в процессе ее оценки. Надстройки устанавливаются и как автономные утилиты, и как части отдельных приложений. Они регистрируются в среде политики, чтобы их можно было загружать по мере надобности.

Надстройка политики должна предоставлять следующие интерфейсы:

- **IPolicyDefinition** Методы, используемые в процессе определения требований политики к командным проектам.
- **IPolicyEvaluation** Методы, используемые в процессе оценки соответствия требованиям политики во время возврата после правки. Принимают возвращаемое содержимое и анализируют его на предмет соответствия определенной политике.

Вы можете упаковать несколько надстроек политик в один файл сборки. Единственное требование — реализовать надстройки как отдельные классы.

Примечание Данные интерфейсы отображены в классе **PolicyBase**. В качестве альтернативы применению интерфейсов **IPolicyDefinition** и **IPolicyEvaluation** вы можете использовать производный класс из **PolicyBase**.

Дополнительные ресурсы

- Дополнительную информацию о создании и использовании пользовательской политики возврата после правки вы найдете в разделе «Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server» этой книги.
- Дополнительную информацию о настройке политики возврата после правки вы найдете в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- С примером кода, запрещающего определенные варианты программирования, можно ознакомиться по материалам статьи «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- С примером кода, заставляющего добавлять комментарии при возврате после правки, можно познакомиться в статье «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- Дополнительную информацию о том, как зарегистрировать новую политику возврата после правки, вы найдете в статье «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.
- Дополнительную информацию об использовании инструментов анализа кода вы найдете в статье «Guidelines for Using Code Analysis Tools» по адресу *http://msdn2.microsoft.com/en-us/library/ms182023(VS.80).aspx*.
- Дополнительную информацию о создании новой политики вы найдете в статье «Policy Plug-ins» по адресу *http://msdn2.microsoft.com/en-us/library/bb130343(VS.80).aspx*.

Отладка, извлечение и блокировка

- Как синхронизировать компьютер с TFS.
- Как подготовить файл к редактированию.

Как синхронизировать компьютер с TFS

Для синхронизации компьютера с сервером управления версиями используется команда **tf get**. С ее помощью вы легко синхронизируете свою работу с остальными разработчиками и всегда будете иметь дело с новейшими версиями файлов. Чтобы загрузить все файлы, а не только обновленные, запустите в окне командной строки Visual Studio 2005следующую команду:

```
tf get /all
```

При запуске этой команды перезапись всех записываемых локальных файлов, имеющихся на вашем компьютере, не производится. Если вы хотите перезаписать локальные записываемые файлы для полной синхронизации вашего компьютера с сервером, используйте ключ **/force**, как показано в примере:

```
tf get /force
```

Эта команда перезаписывает все локальные записываемые файлы за исключением тех, для которых у вас есть незавершенные правки. Если у вас имеются незавершенные правки файла, которые вам хотелось бы сохранить, перед синхронизацией возвратите их на сервер или отложите командой shelve.

Чтобы провести синхронизацию из Visual Studio, выполните следующие действия:

1. В окне Team Explorer дважды щелкните папку **Source Control**, правой кнопкой щелкните сервер или командный проект и выберите команду **Get Specific Version**.
2. Задайте параметры **Overwrite writable files that are not checked out** и **Force get of file versions already in workspace**.
3. Убедитесь, что в раскрывающемся списке **Type** выбран вариант **Latest Version** и щелкните кнопку **Get**.

Чтобы полностью синхронизировать компьютер с сервером управления версиями, не задавайте в Visual Studio параметр **Get Latest Version**. Эта команда загружает только те файлы, которых нет в вашей рабочей области, и не перезаписывает записываемые файлы, извлеченные в локальную папку. Таким образом, синхронизация компьютера с сервером фактически не выполняется.

Дополнительные ресурсы

- Дополнительную информацию о команде **get** вы найдете в статье «Get Command» по адресу *http://msdn2.microsoft.com/pt-br/library/fx7sdeyf(VS.80).aspx*.

Как подготовить файл к редактированию

Чтобы подготовить файл к редактированию, сначала следует получить его последнюю версию из системы управления исходным кодом Team Foundation Server, после чего извлечь файл для редактирования.

Подготовка файла для редактирования

1. В окне Source Control Explorer выберите файл, щелкните его правой кнопкой мыши и выберите команду **Get Latest Version**. Это приведет к загрузке последней версии файла в рабочую область на вашем компьютере. Пока она будет доступна только для чтения.

2. Щелкните файл правой кнопкой и выберите команду **Check Out for Edit**.
3. Задайте тип блокировки. Выберите **None**, чтобы разрешить другим пользователям извлекать и возвращать файл одновременно с вами.

Как правило, рекомендуется использовать именно этот тип блокировки, так как большинство возникающих при этом конфликтов может быть разрешено автоматически.

> **Примечание** Не путайте получение последней версии файла (Get Latest Version) и его извлечение для редактирования (Check Out for Edit). Это разные операции, и они должны выполняться отдельно. В этом TFS отличается от Microsoft Visual SourceSafe.

Выбирая тип блокировки, учитывайте следующие соображения:

- Тип блокировки **None** позволяет избежать задержек, связанных с невозможностью одновременной работы над одним и тем же файлом.
- Блокировать файл на время редактирования следует только в случае, если вы опасаетесь возникновения конфликтов, который приведут к необходимости трудоемкого ручного слияния.
- Выбрав тип блокировки **Check Out**, вы лишаете других пользователей возможность извлекать и возвращать файл. Это фактический запрет на редактирование файла, который может привести к замедлению разработки. При этом у вас появляется возможность применять изменения к БД управления исходным кодом, не опасаясь изменений, сделанных другими пользователями.
- Тип блокировки **Check In** позволяет другим пользователям извлекать файл для редактирования, но не разрешает возвращать его. Этот вариант также гарантирует вам бесконфликтное возвращение ваших правок.

Дополнительные ресурсы

- Дополнительную информацию о команде **checkout** вы найдете в статье «Checkout and Edit Commands» по адресу *http://msdn2.microsoft.com/pt-br/library/1yft8zkw(VS.80).aspx*.

Совместное использование кода

- Как организовать общий доступ к коду.
- Как управлять общими двоичными файлами.

Как организовать общий доступ к коду

Если у вас имеется исходный код, используемый в нескольких командных проектах, вы вольны управлять им из проекта команды-владельца или создать командный проект специально для общего исходного кода.

Для разработчиков, использующих общий исходный код, существует два

варианта действий:

- установить ссылку на код из общего расположения;
- выполнить ветвление общего кода.

Установка ссылки на код из общего расположения

В проектах, использующих общий код, можно создать сопоставление между общим расположением и рабочими областями на клиентских компьютерах. Конфигурация, объединяющая исходный код в общей папке с командными проектами, создается на стороне клиентов.

Преимущество этого способа состоит в том, что изменения, вносимые в общий исходный код, переносятся при каждом извлечении новейшего исходного кода в рабочую область. Допустим, у вас есть два командных проекта — Client и Shared Code. В проекте Shared Code расположен общий исходный код. Чтобы установить ссылки на код в общем расположении, эти проекты используют единый путь на диске клиента, как показано в примере:

- c:\TestProject\Client
- c:\TestProject\Shared Code

  В обоих проектах имеются сопоставления с этими локальными путями.

Папка системы управления исходным кодом	Локальная папка
$/Client	c:\TestProject\Client
$/Shared Code	c:\TestProject\Shared Code

Дополнительную информацию вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Ветвление общего кода

Из проектов, использующих общий код, можно выполнить ветвление из общего расположения. Конфигурация, объединяющая исходный код в общей папке с командными проектами, создается на стороне сервера.

Отличие от предыдущего варианта состоит в том, что изменения общего исходного кода переносятся в процессе слияния ветвей. Таким образом, принятие решения о переносе изменений из общего исходного кода принимается более явным образом.

Допустим, у вас есть два командных проекта — Client и Shared Code. В проекте Shared Code расположен общий исходный код. Для ветвления кода из общего расположения выполните следующие действия:

1. В окне Source Control щелкните правой кнопкой корневую папку проекта Shared Code.
2. Выберите команду **Branch**.
3. В диалоговом окне **Branch** укажите в поле **Target** корневую папку коман-

дного проекта Client. Щелкните **OK**.

4. По завершению операции ветвления не забудьте возвратить исходный код, полученный в результате ветвления.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.
- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ссылках проекта вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.

Как управлять общими двоичными файлами

Управление общими двоичными файлами очень похоже на управление общим исходным кодом: главное — решить, где вы собираетесь хранить двоичные файлы и как будет организован общий доступ к ним.

Существуют следующие варианты хранения общих двоичных файлов:

- Если владельцем общих двоичных файлов является определенная команда, их следует хранить в проекте этой команды.
- Если общие двоичные файлы конкретно никому не принадлежат, создайте командный проект специально для общих двоичных файлов.

Существуют следующие варианты использования двоичных файлов из другого проекта:

- Обновление общих двоичных файлов, как правило, происходит только на периодической основе. Если вы используете этот вариант, выполните ветвление из общего расположения в командный проект-потребитель. В случае их изменения вы можете выполнить слияние, чтобы получить их обновленную версию.
- Если вы хотите синхронизировать свои двоичные файлы с общими на постоянной основе, сопоставьте исходный код из общего расположения в локальную рабочую область на компьютерах клиентов.

Ветвление общих двоичных файлов в проект

1. В окне Source Control правой кнопкой щелкните корневую папку проекта с общими двоичными файлами.
2. Выберите команду **Branch**.
3. В диалоговом окне **Branch** укажите в поле **Target** корневую папку клиентского командного проекта. Щелкните **OK**.

4. По завершению операции ветвления не забудьте возвратить исходный код, полученный в результате ветвления.

Как при использовании рабочей области, так и при использовании ветвления следует соблюдать соглашения об именах, позволяющие точно определять расположение общих двоичных файлов в проекте, например:

- **Main**.
  - **Source** — код проекта.
  - **Lib** — общие двоичные файлы.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Working with multiple team projects in Team Build» по адресу http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx
- Вводный курс по ветвлению и слиянию вы найдете в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ссылках проекта вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.

Зависимости

- Как управлять зависимостями веб-служб.
- Как управлять зависимостями БД.

Как управлять зависимостями веб-служб

Как правило, URL веб-службы в рабочей среде отличается от ее URL в средах разработки и тестирования. Чтобы облегчить управление веб-службами, значение URL нужно указывать в пользовательском файле конфигурации, который может изменяться отдельными разработчиками и тестировщиками, не затрагивая главный конфигурационный файл App.config. Для этого следует присвоить свойству **URL Behavior** ссылки на веб-службу значение **Dynamic**. Ссылайтесь на URL веб-службы при помощи пользовательского файла конфигурации.

По умолчанию при добавлении веб-ссылки Visual Studio присваивает указанному свойству значение **Dynamic**.

Проверка значения свойства URL Behavior

1. В Solution Explorer разверните список веб-ссылок.
2. Выделите все веб-ссылки в списке.
3. Убедитесь, что свойству **URL Behavior** каждой ссылки присвоено значение **Dynamic**.

Указание URL веб-службы в пользовательском файле конфигурации

При первом добавлении веб-ссылки файл App.config выглядит примерно так:

```
<configuration>
   <configSections>
      <sectionGroup name="applicationSettings" type="System.Configuration.
ApplicationSettingsGroup, System, Version=2.0.0.0, Culture=neutral, PublicK
eyToken=b77a5c561934e089" >
         <section name=" SomeService.Properties.Settings" type="System.
Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"
/>
      </sectionGroup>
   </configSections>
   <applicationSettings>
      <YourProject.Properties.Settings>
         <setting name="SomeService_ localhost _Service"
serializeAs="String">
            <value>http://localhost/someservice/Service.asmx</value>
         </setting>
      </ YourProject.Properties.Settings>
   </applicationSettings>
</configuration>
```

В этом файле есть конфигурационный новый раздел с адресом веб-службы, заданным Visual Studio при создании этого прокси.

Создание файла User.config

1. В окне Solution Explorer щелкните правой кнопкой проект, содержащий ссылку на веб-службу, раскройте подменю **Add** и выберите команду **New Item**.
2. Выделите **Application Configuration File**, измените имя на **User.config** и щелкните **Add**.
3. Скопируйте параметр <*YourProject.Properties.Settings*> из файла App.config в файл User.config. Этот файл должен содержать только параметры, которые изменяются во время выполнения. Удалите директиву <?xml> и элемент <configuration>, если они имеются, как показано в примере:

```
<YourProject.Properties.Settings>
   <setting name="SomeService_localhost_Service" serializeAs="String">
      <value>http://localhost/someservice/Service.asmx</value>
   </setting>
</YourProject.Properties.Settings>
```

4. В окне Solution Explorer щелкните правой кнопкой файл User.config, выберите команду **Properties** и присвойте свойству **Copy to Output Directory** значение **Copy if newer**.

Каждый разработчик задает в файле User.config ссылку на нужный ему URL веб-службы.

Создание в файле App.config ссылки на файл User.config при доступе к URL веб-службы

1. В элемент <*YourProject.Properties.Settings*> главного файла конфигурации приложения добавьте атрибут **configSource="user.config"**. При достижении рабочим циклом информации, содержащейся в этом разделе, произойдет перенаправление рабочего цикла в заданный пользовательский файл конфигурации.
2. Удалите содержимое элемента <*YourProject.Properties.Settings*>.
   Теперь файл App.config должен выглядеть примерно так:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
      <sectionGroup name="applicationSettings" type="System.
Configuration.ApplicationSettingsGroup, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" >
         <section name="SomeService.Properties.Settings" type="System.
Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="fal
se" />
      </sectionGroup>
   </configSections>
   <applicationSettings>
      <YourProject.Properties.Settings configSource="user.config">
      </YourProject.Properties.Settings>
   </applicationSettings>
</configuration>
```

В предыдущем примере элемент YourProject представляет собой имя проекта, содержащего ссылку на веб-службу. Убедитесь, что элемент <*SomeService.Properties.Service*> в файле App.config пуст.

Учитывайте следующие соображения:

- Не добавляйте пользовательский файл конфигурации в систему управления исходным кодом. Для этого при первом возврате файла сбросьте флажок **User.config**. Затем щелкните файл в окне Solution Explorer правой кнопкой и выберите команду **Under Pending Changes**, чтобы не переносить файл в систему управления исходным кодом. Теперь каждый разработчик (и тестовая команда) сможет привязаться к конкретному URL с помощью собственного файла User.config.

- Система управления исходным кодом может содержать файлы User.config, например, для тестирования или производства. Этими файлами должны распоряжаться пользователи, ответственные за управление соответствующими средами. Такие файлы User.config, используемые при испытаниях и в производстве, не должны храниться в составе проектов веб-служб — они должны находиться в других областях системы управления исходным кодом.
- Глобальный файл User.config следует хранить в системе управления исходным кодом. В нем может содержаться либо только корневой элемент (не элемент <setting>), либо указание на стандартное положение веб-службы. Файл User.config нужен для работы системы конфигурирования.

Важно понимать, что при использовании этого механизма файл User.config обязательно должен быть в наличии. Кто-то из команды должен отвечать за правильную работу среды во время создания сборок для рабочих выпусков и для тестирования. При сборке соответствующий файл User.config должен быть извлечен из системы управления исходным кодом и скопирован в определенное расположение, чтобы система MSBuild могла его найти.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Web Projects and Source Control Integration in Visual Studio .NET» по адресу *http://msdn2.microsoft.com/en-US/library/aa290068(VS.71).aspx*.
- Дополнительные сведения о классе **ApplicationSettingsGroup** вы найдете в статье «ApplicationSettingsGroup Class» по адресу *http://msdn2.microsoft.com/en-us/library/system.configuration.applicationsettingsgroup.aspx*.

Как управлять зависимостями БД

Далее объясняется, как хранить и устанавливать ссылку на строку подключения БД в пользовательском файле конфигурации.

Хранение строк подключения БД в пользовательском файле конфигурации

1. В главном файле конфигурации приложения добавьте атрибут **configSource="user.config"** в элемент <connectionStrings>, как показано в примере:

```
<configuration>
   <connectionStrings configSource="user.config"/>
</configuration>
```

2. Чтобы перекрыть главный файл конфигурации приложения, создайте файл User.config (расположенный в той же папке, что и главный файл конфигурации приложения) и добавьте в него такой же элемент

<connectionStrings>. Обратите внимание, что приведенная ниже строка подключения ссылается на локальную базу данных.

```
<configuration>
   <connectionStrings>
      <add name="DBConnStr" connectionString="server=localhost;Integrate
d Security=SSPI;database=Accounts"/>
   </connectionStrings>
</configuration>
```

3. В проекте для получения строки подключения из пользовательского файла конфигурации используйте код, в котором используется свойство **ConnectionStrings** класса **System.Configuration.ConfigurationManager**. В приложении Win Form вы должны явно добавить ссылку на **System.Configuration.dll**.

```
using System.Configuration;
private string GetDBaseConnectionString()
{
   return ConfigurationManager.ConnectionStrings["DBConnStr"].
ConnectionString;
}
```

4. Убедитесь, что файл User.config устанавливается вместе с кодом приложения. Для этого в окне Solution Explorer щелкните правой кнопкой мыши файл **User.config**, выберите команду **Properties** и присвойте свойству **Copy to Output Directory** значение **Copy if newer**.

Не добавляйте пользовательский файл конфигурации в систему управления исходным кодом. При этом каждый разработчик (и тестовая команда) сможет задавать строку подключения с помощью собственного файла User.config. Система управления исходным кодом может содержать файлы User.config, например, для тестирования или производства. Этими файлами должны распоряжаться пользователи, ответственные за управление соответствующими средами. Такие файлы User.config, используемые при испытаниях и в производстве, не должны храниться в составе проектов БД — они должны находиться в других областях системы управления исходным кодом. Файл User.config нужен для работы системы конфигурирования.

> **Совет** По умолчанию во время добавления решения файл User.config автоматически добавляется в систему управления исходным кодом. Чтобы избежать этого, при первом возврате файлов после правки сбросьте флажок User.config. Чтобы гарантировать непопадание этого файла в систему управления исходным кодом, щелкните его правой кнопкой в окне Solution Explorer и выберите команду **Under Pending Changes**.

Важно понимать, что при использовании этого механизма файл User.config обязательно должен быть в наличии. Кто-то из команды должен отве-

чать за правильную работу среды во время создания сборок для рабочих выпусков и для тестирования. При сборке соответствующий файл User.config должен быть извлечен из системы управления исходным кодом и скопирован в определенное расположение, чтобы система MSBuild могла его найти.

Дополнительные ресурсы

- Дополнительную информацию об использовании файла конфигурации для определения источника данных вы найдете в статье «Walkthrough: Using a Configuration File to Define a Data Source» по адресу *http://msdn2.microsoft.com/en-us/library/ms243192(VS.80).aspx*.
- Дополнительную информацию об атрибуте **configSource** вы найдете в статье «SectionInformation.ConfigSource Property» по адресу *http://msdn2.microsoft.com/en-us/library/system.configuration.sectioninformation.configsource.aspx*.

Распределенная и удаленная разработка

- Как получить доступ к TFS через Интернет.
- Как повысить производительность TFS-прокси.

Как получить доступ к TFS через Интернет

Доступ к TFS через Интернет можно организовать одним из трех способов:

- **подключение по виртуальной частной сети (VPN)**;
- **публикация TFS через обратный прокси**, например, Microsoft Internet Security and Acceleration (ISA);
- **Расположение TFS в экстрасети**.

Используйте первый способ, если вы и так обеспечиваете поддержку удаленных пользователей при помощи VPN. Он относительно прост в реализации, обладает понятной системой безопасности, обеспечивает удаленный доступ ко всем функциям TFS и позволяет использовать для повышения производительности TFS Proxy. При таком способе доступа TFS находится во внутренней сети, а внешние пользователи получают к нему доступ по VPN. Внутренние пользователи обладают прямым доступом к TFS.

Если поддержка удаленных пользователей осуществляется без доступа к VPN или к домену, используйте сценарий с обратным прокси. Этот способ сложнее осуществить, однако он позволяет удаленным пользователям получать доступ к TFS, расположенному во внутренней сети, без использования VPN. В этой реализации TFS находится во внутренней сети, а один или несколько обратных прокси-серверов, например, ISA Server, доставляют на TFS запросы клиентов из Интернета.

Сценарий с экстрасетью подходит в ситуациях, когда вы поддерживаете достаточно большое количество удаленных пользователей, например, сайт

разработки какого-либо сообщества. При этом удаленные пользователи работают в специально предназначенной для них реализации TFS. Данный подход обеспечивает максимальное разделение между удаленными пользователями и внутренними ресурсами сети. При этом доступ к TFS имеют только внешние клиенты, а сам TFS расположен в экстрасети за пределами брандмауэра.

Если у вас есть удаленный офис с несколькими клиентами, осуществляющими доступ к Team Foundation Server через Интернет, установите в удаленном офисе Team Foundation Server Proxy. Это увеличит производительность за счет кеширования файлов исходного кода на прокси-сервере. Если вы поддерживаете одного клиента, удаленно подключающегося к TFS, настройте его на подключение непосредственно к TFS.

Дополнительные ресурсы

- Дополнительные сведения о сценариях удаленного доступа к TFS содержатся в главе 17 этой книги.
- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Как повысить производительность TFS-прокси

Установите и настройте Team Foundation Server Proxy в удаленном офисе. Это позволит повысить производительность за счет кеширования файлов системы управления исходным кодом на прокси-сервере.

Чтобы настроить и оптимизировать производительность TFS-прокси, выполните следующие действия:

1. Убедитесь, что функция кеширования включена, и проверьте счетчики производительности кеша. Чтобы иметь представление о производительности прокси, счетчики производительности (устанавливаемые по умолчанию) и журналы регистрации событий (ошибки и предупреждения) на прокси-сервере следует проверять периодически.

> **Примечание** Прокси TFS сохраняет статистику производительности кеша в XML-файле ProxyStatistics.xml. Вы можете изменить интервал, с которым происходит сохранение статистики. Файл ProxyStatistics.xml расположен в подпапке App_Data папки установки прокси.

2. Периодически запускайте запланированное задание для извлечения последних версий файлов на прокси-сервер. Это обеспечит актуальность информации в кеше и увеличит количество попаданий в кеш.
3. Если вы заранее знаете о готовящейся передаче больших файлов по медленной сети (< 3 Мбит/с), присвойте соответствующее значение пара-

метру **executionTimeout** в файле Web.config. Значение по умолчанию равно одному часу — <httpRuntime executionTimeout="3600"/>.

Дополнительные ресурсы

- Дополнительные сведения о сценариях удаленного доступа к TFS содержатся в главе 17 этой книги.
- Дополнительную информацию о Team Foundation Server Proxy вы найдете в статье «Team Foundation Server Proxy and Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.
- Дополнительную информацию о тестировании производительности прокси-сервера TFS вы найдете в статье «How to: Examine Cache Performance for Team Foundation Server Proxy» по адресу *http://msdn2.microsoft.com/en-us/library/ms252455(VS.80).aspx*.
- Дополнительные сведения о TFS Proxy вы найдете в веб-трансляции MSDN по адресу *http://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?culture=en-US&EventID=1032291120&CountryCode=US*.

Миграция

- Как осуществить перенос исходного кода с Visual SourceSafe.
- Как осуществить перенос исходного кода из других систем управления версиями.

Как осуществить перенос исходного кода из Visual SourceSafe

Для переноса исходного кода из VSS выполните следующие действия.

> **Примечание** Для выполнения этих действий нужно быть членом группы администраторов Team Foundation.

1. **Подготовка VSS.** Приготовьтесь к переходу, создав резервные копии базы данных VSS и убедившись в том, что файлы возвращены в систему. Запустите инструмент Visual SourceSafe Analyze для выявления и разрешения конфликтов целостности данных в существующей БД.
2. **Анализ проектов.** Запустите конвертер (инструмент командной строки VSSConverter.exe), передав ему с ключом **analyze** имя XML-файла, содержащего необходимые параметры, как показано в примере:

   ```
   VSSConverter analyze conversionsettings.xml
   ```

   Пример XML-файла параметров:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <SourceControlConverter>
   ```

```
   <ConverterSpecificSetting>
      <Source name="VSS">
         <VSSDatabase name="c:\VSSDatabase"></VSSDatabase>
      </Source>
      <ProjectMap>
         <Project Source="$/MyFirstProject"></Project>
         <Project Source="$/MySecondProject"></Project>
      </ProjectMap>
   </ConverterSpecificSetting>
</SourceControlConverter>
```

Файл параметров содержит имя БД VSS. В атрибуте **name** задается имя папки, содержащей .ini-файл хранилища исходного кода. Элементы <Project> определяют пути к проектам в БД VSS, которые вы собираетесь преобразовать. Для миграции всей БД VSS введите <Project Source="$/"></Project>.

Команда **analize** инструмента VssConverter.exe создает файл usermap.xml. Добавив сопоставления в этот файл, вы можете изменить имена, связанные с историей версий и другие, с регистрационных имен VSS на регистрационные имена TFS Windows.

3. **Перемещение проектов.** Выберите папки, которые хотите перенести, и запустите инструмент VSSConverter.exe с аргументом **migrate**, как показано в примере:

```
VSSConverter migrate conversionsettings.xml
```

Вы снова передаете команде XML-файл параметров настройки, но на этот раз с двумя важными дополнениями:

```
<?xml version="1.0" encoding="utf-8"?>
<SourceControlConverter>
   <ConverterSpecificSetting>
      <Source name="VSS">
         <VSSDatabase name="c:\VSSDatabase"></VSSDatabase>
      </Source>
      <ProjectMap>
         <Project Source="$/MyFirstProject" Destination="$/MyTeam_
ProjectOne"></Project>
         <Project Source="$/MySecondProject" Destination="$/MyTeam_
ProjectTwo"></Project>
      </ProjectMap>
   </ConverterSpecificSetting>
   <Settings>
      <TeamFoundationServer name="YourTFSServerName" port="PortNumber"
protocol="http"></TeamFoundationServer>
   </Settings>
</SourceControlConverter>
```

Обратите внимание на дополнительный атрибут **Destination** в элементах <Project>. Значение этого атрибута указывает на командный проект TFS (созданный вами заранее). Элемент <Settings> содержит подробности подключения уровня приложений TFS.

Дополнительные ресурсы

- Дополнительную информацию о подготовке к миграции вы найдете в статье «Walkthrough: Preparing to Migrate from Visual SourceSafe to Team Foundation» по адресу *http://msdn2.microsoft.com/en-us/library/ms181246(en-us,vs.80).aspx*.
- Дополнительную информацию о миграции вы найдете в статье «Walkthrough: Migrating from Visual SourceSafe to Team Foundation» по адресу *http://msdn2.microsoft.com/en-us/library/ms181247(VS.80).aspx*.
- Дополнительную информацию об ограничениях конвертера Visual SourceSafe вы найдете в статье «Visual SourceSafe Converter Limitations» по адресу *http://msdn2.microsoft.com/en-us/library/ms252491(VS.80).aspx*.

Как осуществить перенос исходного кода из других систем управления версиями

Вы можете вручную экспортировать файлы из прежней системы управления версиями, а затем импортировать их в Team Foundation Server. Чтобы сохранить историю и другие атрибуты системы, воспользуйтесь объектной моделью Team Foundation Server, чтобы написать собственный инструмент миграции.

В настоящий момент корпорация Microsoft ведет работу по созданию конвертера ClearCase. О выпуске конвертера будет объявлено дополнительно в блоге TFS Migration по адресу *http://blogs.msdn.com/tfs_migration*. Существует также конвертер, созданный компанией Component Software, совместимый с GNU RCS, CS-RCS, GNU CVS, Subversion (SVN) и Visual SourceSafe (VSS).

Дополнительные ресурсы

- Загрузить набор инструментов Visual Studio Team Foundation Server SDK можно из источника, расположенного по адресу *http://go.microsoft.com/fwlink/?linkid=68586*.
- Дополнительную информацию о расширяемости системы Team Foundation Server вы найдете в статье «Walkthru: The Version Control Object Model» по адресу *http://msdn2.microsoft.com/en-us/library/bb187335 (VS.80).aspx*.
- Дополнительную информацию о конвертере компании Component Software вы найдете на сайте компании по адресу *http://www.component-software.com/Products/Converter/*.

Управление проектом и рабочей областью

- Как выбрать между созданием одного командного проекта и нескольких.
- Как организовать дерево исходного кода.
- Как определять сопоставления рабочей области.
- Как изолировать изменения кода на компьютере с помощью рабочих областей.

Как выбрать между созданием одного командного проекта и нескольких

Чтобы спланировать структуру проекта, рассмотрите типичные стратегии организации проектов и выберите ту, что максимально подходит для вас, исходя из размеров предприятия, серверных ограничений и принятых правил работы. Проект олицетворяет самый крупный блок работы, который может существовать в вашей организации. Предпочтение следует отдавать одиночным проектам. Несколько проектов нужно использовать, только имея на то серьезные основания, например, изменения в команде или наличие значительных различий между выпусками, когда вы не хотите переносить в новый выпуск нежелательные рабочие элементы (ошибки и т. п.). Основанием для создания нескольких проектов может также стать изменение в шаблоне процесса.

Основным преимуществом одного проекта перед несколькими проектами является упрощение процедуры переноса требований, функций, сценариев и ошибок из выпуска в выпуск.

Основные вопросы, которые вам предстоит ответить, таковы:

- Хотите ли вы от выпуска к выпуску сохранять рабочие элементы и другие ресурсы? Если да, храните все выпуски в одном проекте.
- Хотите ли вы при переходе на новый выпуск заново создавать новую структуру рабочих элементов и процессов? Если да, создавайте новый проект для каждого нового выпуска.

  Далее описаны типичные структуры проектов.

Один проект на все приложение

Используется один проект, содержащий все версии приложения. Внутри проекта для отдельных выпусков создаются ветви.

- **Почему следует использовать эту структуру?**
  - Она упрощает перенос кода, рабочих элементов и прочих ресурсов.
- **Почему не следует использовать эту структуру?**
  - В параллельно разрабатываемых выпусках приходится использовать общие политики возврата после правки, схему рабочих элементов и руководство по процессам.

- Сложно составлять отчеты об ошибках: стандартные отчеты составляются для всего проекта, поэтому вам придется производить фильтрацию по выпуску.
- Если у вас сотни приложений, каждое из которых находится в собственном проекте, вы можете превысить предел масштабируемости TFS.
- По мере роста количества выпусков у вас накапливается солидный «багаж». Лучший способ избавиться от него — создать новый проект и перенести в него нужный вам в будущем код.

Один проект на один выпуск

Создается новый проект для каждой версии вашего приложения.

- **Почему следует использовать эту структуру?**
  - Она удобна, если вы хотите начинать проект заново после каждого выпуска.
  - Старый проект можно использовать для сопровождения выпуска, передав отдельной группе.
  - Легко перемещать исходный код из одного проекта в другой.
- **Почему не следует использовать эту структуру?**
  - Достаточно сложно перемещать рабочие элементы и ресурсы TFS из одного проекта в другой. Рабочие элементы можно копировать в другой проект только по одному. Если вам захочется переместить наборы элементов, придется делать это вручную или написать собственную утилиту.
  - Если вы управляете сотнями проектов, то рискуете превысить предел масштабируемости TFS.

Выбирая стратегию, имейте в виду следующее:

- Выбирая структуру, думайте о дне завтрашнем: реструктуризация существующих командных проектов — трудное занятие.
- Имеются способы организовать общий доступ к исходному коду из нескольких командных проектов:
  - ветвлением исходного кода из одного проекта в другой;
  - сопоставлением исходного кода из другого проекта в вашу рабочую область.
- Система Team Foundation Server способна вместить около 500 проектов Microsoft Solution Framework (MSF), основанных на шаблоне процесса Agile Software Development (MSF Agile), или 250 проектов на шаблоне процесса MSF CMMI. Создавая собственный процесс или настраивая существующий, помните, что на масштабируемость сервера оказывает огромное влияние схема рабочего элемента. Чем сложнее схема, там меньше проектов сможет поддерживать сервер.

Дополнительные ресурсы

- Дополнительную информацию о стратегии выбора проекта вы найдете в блоге Эрика Ли (Eric Lee) «When to use Team Projects» по адресу *http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx*.

Как организовать дерево исходного кода

Структура дерева исходного кода состоит из комбинации структуры папок, файлов и ветвей. Внутри главной ветви свою работоспособность в самых различных по размеру командах доказала следующая структура папок и файлов:

- **Main** — контейнер для всех объектов, необходимых для передачи проекта заказчику.
  - **Source** — контейнер для всех объектов, необходимых для выполнения сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, используемого совместно с другими проектами.
    - **Unit Tests** — контейнер для модульных тестов.
    - **Lib** — контейнер для двоичных зависимостей.
  - **Docs** — контейнер для документации, поставляющейся с проектом.
  - **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
  - **Tests** — контейнер, содержащий результаты испытаний, проводимых тестовой командой.

При ветвлении папки Main структура папок и файлов будет скопирована в новую ветвь, например:

- **Development** — ветвь разработки.
  - **Source** — контейнер для всех объектов, необходимых для выполнения сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, используемого совместно с другими проектами.
    - **Unit Tests** — контейнер для модульных тестов.
    - **Lib** — контейнер для двоичных зависимостей.
- **Main** — ветвь интеграции.
  - **Source** — контейнер для всех объектов, необходимых для выполнения сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, используемого сов-

местно с другими проектами.

- ▲ **Unit Tests** — контейнер для модульных тестов.
- ▲ **Lib** — контейнер для двоичных зависимостей.

- ❑ **Docs** — контейнер для документации, поставляющейся с проектом.
- ❑ **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
- ❑ **Tests** — контейнер, содержащий результаты испытаний, проводимых тестовой командой.

Как определять сопоставления рабочей области

Для отображения файлов и папок системы управления исходным кодом, расположенных на сервере, определите на вашем локальном диске сопоставление рабочей области.

Создание сопоставления рабочей области для проекта, которого еще нет на жестком диске

1. В окне Source Control Explorer выделите корневую папку исходного кода.
2. Щелкните ее правой кнопкой и выберите команду **Get Latest Version**.
3. Выберите локальную папку, в которую хотите отобразить рабочую область.

Изменение сопоставления рабочей области проекта, уже имеющегося на жестком диске

1. Выберите в меню **File** команды **Source Control** и **Workspaces**.
2. В диалоговом окне **Manage Workspaces** добавьте, удалите или отредактируйте существующую рабочую область.

Просмотр существующего сопоставления рабочей области

1. В окне Source Control Explorer выделите папку с исходным кодом.
2. Щелкните ее правой кнопкой и выберите команду **Properties**. В разделе **Local Name** будет отображено сопоставление рабочей области на локальном диске.

Создавая сопоставления рабочей области, используйте следующие рекомендации:

- **Сопоставляйте рабочие области на корневом уровне командного проекта** В новых командных проектах сопоставляйте корень проекта (**$/**MyTeamProject) с папкой на локальном диске, имеющей то же имя, например, C:\TeamProjects. Вся структура локальной папки создается автоматически и будет в точности повторять структуру в системе управления исходным кодом.
- **Используйте уникальный путь к локальной папке на совместно исполь-**

зуемых компьютерах Два пользователя одного и того же компьютера не могут использовать одно сопоставление рабочей области. Допустим, вы и ваш коллега не можете сопоставить один командный проект ($/MyTeamProject) с одной и той же папкой на локальном компьютере. Создавайте сопоставления в папке Мои документы (хотя это удлиняет путь) или разработайте соглашение об именах для папок на локальном компьютере (например, C:\TeamProjects\User1, C:\TeampProjects\User2 и т. д.).

- **Подумайте, нужно ли вам все дерево** Чтобы увеличить производительность и сократить занимаемый объем диска, сопоставляйте только те файлы, которые требуются для проекта разработки. Как правило, вам требуются файлы и проекты, связанные с решением, над которым вы работаете.
- **Не используйте сопоставление рабочей области для поддержки зависимостей между различными проектами** Как правило, следует избегать зависимостей, пересекающих командные проекты. Старайтесь объединить все связанные и зависимые решения и проекты в одном командном проекте. Это позволит реже придется прибегать к настройке сборочного сценария. Если у вас есть зависимость, для ее определения используйте ссылки на проект или перенесите зависимость из общего проекта в свой проект. Следует избегать файловых ссылок, потому что ими сложнее управлять. Исключение составляют случаи, когда параллельно ведется разработка зависимого проекта, и вам нужно получать изменения в реальном времени. В этом случае вы можете воспользоваться сопоставлением рабочей области. Если зависимый код приводит к появлению большого количества серьезных ошибок, используйте ветвление.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию о редактировании рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Как изолировать изменения кода на компьютере с помощью рабочих областей

Разработчик волен создать две рабочие области: одна из них содержит ссылки на файлы и папки, над которыми работает остальная команда, а вторая — файлы и папки, которые он хочет изолировать. Изолирование может понадобиться, если вы хотите работать с файлами, которые параллельно изменяются где-то еще. Например, им можно пользоваться для работы над рискованными изменениями или при выполнении обзора кода.

Создание второй рабочей области

1. В окне Source Control Explorer щелкните раскрывающийся список **Workspace** и выберите команду **Workspaces**.
2. В диалоговом окне **Manage Workspaces** щелкните кнопку **Add**.
3. В диалоговом окне **Add Workspace** введите имя новой рабочей области, например, *ИзолированнаяРабота*. Добавьте комментарий, напоминающий о цели создания рабочей области.
4. В списке **Working folders** задайте статус рабочего места **Active**, определите папку в системе управления исходным кодом, которая будет включена в рабочую область. Это может быть корневая папка командного проекта или любая вложенная папка. Задайте путь на локальном компьютере, в котором будут находиться файлы рабочей области.
5. Щелкните **OK** и **Close**, чтобы создать изолированную рабочую область.

Извлечение актуального набора исходного кода для работы в изолированной рабочей области

1. В окне Source Control Explorer убедитесь, что в раскрывающееся списке **Workspace** выбрано имя изолированной рабочей области.
2. Выберите корневую папку командного проекта (или вложенную папку, если нужна только часть дерева исходного кода), щелкните ее правой кнопкой и выберите команду **Get Latest Version**.

При этом будет выполнено копирование структуры папок и актуального набора файлов с сервера управления исходным кодом в папку на локальном компьютере, отображенную в новой рабочей области.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию о редактировании рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Безопасность

- Как защитить канал между рабочей станцией разработчика и TFS.

Как защитить канал между рабочей станцией разработчика и TFS

Для повышения безопасности канала между рабочей станцией разработчика и TFS используйте протоколы HTTPS и SSL. Настройте TFS так, чтобы ис-

пользовались подключения только по этим протоколам, но не по HTTP. Для этого сначала следует разрешить протоколы HTTPS и SSL, а затем предпринять дополнительные шаги, чтобы сделать эти протоколы обязательными.

HTTPS и SSL шифруют сетевой трафик между TFS и клиентами Team Foundation, которым необходим доступ к веб-ресурсам Team Foundation Server, включая порталы проектов, отчеты и рабочие элементы.

Дополнительные ресурсы

- Дополнительную информацию вы найдете в статье «Securing Team Foundation Server with HTTPS and Secure Sockets Layer (SSL)» по адресу *http://msdn2.microsoft.com/en-us/library/aa395265(VS.80).aspx*.
- Дополнительную информацию о том, как установить протокол SSL, вы найдете в статье «Walkthrough: Setting up Team Foundation Server with Secure Sockets Layer (SSL)» по адресу *http://msdn2.microsoft.com/en-us/library/ms242875(VS.80).aspx*.
- Дополнительную информацию о настройке TFS на использование только HTTPS и SSL вы найдете в статье «How to: Configure Team Foundation Server for HTTPS and SSL Only» по адресу *http://msdn2.microsoft.com/en-us/library/aa395285(VS.80).aspx*.

Отложенные правки

- Как использовать отложенные правки для создания резервной копии незавершенной работы.
- Как с помощью отложенных правок передать код другому члену команды.

Как использовать отложенные правки для создания резервной копии незавершенной работы

Чтобы создать резервную копию незавершенных изменений на сервере, создайте отложенные редакции файлов, которые еще не готовы возвратить в систему. При этом исходный код выгружается на сервер, но возврат неоконченной работы, которая может привести к нестабильности сборки, не производится.

Выгрузка отложенных правок на сервер

1. Просмотрите незавершенные изменения: в Solution Explorer щелкните правой кнопкой решение и выберите команду **View Pending Changes**.
2. Выберите файлы, которые хотите выгрузить на сервер, и щелкните **Shelve**.
3. Введите имя набора отложенных правок и комментарий о его предназначении. Щелкните **Shelve**.

Восстановление работы

1. В меню **File** раскройте подменю **Source Control** и выберите команду **Unshelve**).
2. Выберите нужный набор изменений и щелкните **Unshelve**.

Система Team Foundation Server восстановит все отложенные правки в целевую рабочую область в виде незавершенных изменений, если они не вступают в конфликт с уже имеющимися в рабочей области незавершенными изменениями.

Дополнительные ресурсы

- Дополнительную информацию о резервном копировании незавершенных изменений вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Как с помощью отложенных правок передать код другому члену команды

Чтобы отложить редакцию исходного кода для передачи другому члену команды, выполните операцию **Get Latest**, синхронизовав свою рабочую область с последней версией на сервере. Затем произведите сборку приложения, чтобы убедиться в его компилируемости. Выгрузите исходный код в качестве отложенной правки при помощи обозревателя Source Control. Члену команды, которому предназначается код, остается только загрузить его с помощью команды **Unshelve**.

Удобно использовать отложенные правки, когда у вас есть неоконченная работа, завершать которую предстоит другому члену команды. Создание отложенной правки облегчает передачу изменений. Синхронизируя последнюю версию кода, вы получаете возможность получить изменения файлов, сделанные за пределами вашей рабочей области.

Выгрузка набора отложенных правок

1. Щелкните правой кнопкой окно Source Control Explorer и выберите команду **Shelve Pending Changes**.
2. В диалоговом окне **Shelve - Source Files** в поле **Shelve name** введите имя набора отложенных правок, например, **shelvetest**.
3. В поле **Comment** введите комментарий и щелкните **Shelve**.

Файлы и папки копируются на сервер управления исходным кодом, откуда их могут извлечь другие члены команды.

При извлечении набора отложенных правок другим членом команды TFS восстанавливает все отложенные правки в целевую рабочую область в качестве незавершенных изменений, если эти правки не вступают в конфликт с незавершенными изменениями, уже имеющимися в рабочей области.

Извлечение набора отложенных правок

1. В меню **File** Visual Studio 2005 раскройте подменю **Source Control** и выберите команду **Unshelve**.
2. В поле **Owner name** введите имя создателя набора отложенных правок (например, ADVENTUREWORKS\JuanGo или просто juango) и щелкните **Find**.
3. В панели **Results** выделите набор отложенных правок, который хотите извлечь в свою рабочую область, и щелкните **Details**.
4. Если вы хотите удалить набор отложенных правок с сервера управления исходным кодом TFS, сбросьте флажок **Preserve shelveset on server**.
5. При необходимости сбросьте флажок **Restore work items and check-in notes**, если не хотите вместе с набором правок восстанавливать рабочие элементы и заметки о возврате после правки.
6. В открывшемся диалоговом окне **Details**, выберите набор отложенных редакций или отдельные элементы, которые хотите извлечь в свою рабочую область, и щелкните **Unshelve**.

Дополнительные ресурсы

- Дополнительную информацию о резервном копировании незавершенных изменений вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Дополнительные ресурсы по управлению исходным кодом

- Дополнительную информацию о системе управления исходным кодом Team Foundation Server вы найдете в статье «Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181237(VS.80).aspx*.

Вопросы и ответы

В этой части

- Вопросы и ответы: система управления исходным кодом и версиями TFS.

Вопросы и ответы: система управления исходным кодом и версиями TFS

В этом разделе

Доступ к системе управления версиями

- Что такое MSSCCI Provider и когда он используется?
- Какие еще интегрированные среды разработки поддерживаются TFS?
- Когда следует использовать Team Foundation Server Power Tool?
- Каковы наиболее распространенные сценарии расширения системы управления версиями?
- Как работать с системой управления версиями из командной строки?

Администрирование

- Как предоставить разрешение на доступ к файлу в папке, имеющей наследуемые разрешения?
- Что делать, если разработчик покидает проект?
- Как управлять стажерами и другими разработчиками, которым я не доверяю возвращать правки на сервер?
- Как следует изменять разрешения после отправки приложения заказчику?

Ветвление, метки и слияние

- Когда использовать метки?
- В чем отличие меток TFS от меток VSS?
- Что такое ветвление?
- Когда следует выполнять ветвление?
- Когда не следует выполнять ветвление?

- Как использовать ветвление для выпуска приложения?
- Как использовать ветвление для поддержки приложения?
- Как при помощи ветвления сократить количество конфликтов между командами?
- Как при помощи ветвления сократить количество конфликтов в компонентах?
- Как наиболее эффективно провести ветвление и слияние?
- В чем разница между ветвлением и метками?
- Что представляет собой модель ветвления в пространстве путей?
- Что такое модель распространения TFS?
- Как выполнить слияние двух ветвей?
- Можно ли выполнить слияние между командными проектами?
- Что такое слияние без основы?
- Что такое модель продвижения кода?
- В чем отличие между логическим и физическим представлениями ветвей?
- Как часто следует выполнять слияния?

Возврат после правки и политики возврата

- Что такое набор изменений?
- Что такое политика возврата после правки?
- Как перекрыть политику возврата?
- Как гарантировать соблюдение политики?
- Как пользоваться проверкой возврата после правки?
- Приведет ли к нарушению синхронизации переименование или удаление файлов на диске?
- Как работает система автоматического разрешения конфликтов?
- Как разрешать конфликты вручную?
- Как избежать конфликтов?

Извлечение, получение и блокировка

- Как узнать, кто изменял файл последним?
- Как работает команда get?
- В чем разница между совместным и исключающим извлечением?
- Когда следует использовать команду lock?
- Какие типы блокировок поддерживает TFS?

Распределенная и удаленная разработка

- Как работать автономно?

- Как оптимизировать среду распределенной командной разработки?
- Что такое TFS-прокси?
- Как оптимизировать производительность TFS-прокси?

Переход с других версий

- Чем система управления версиями TFS отличается от VSS?
- Чем модель извлечения TFS отличается от аналогичной модели VSS?
- Как осуществить перенос исходного кода из VSS в TFS?
- Как осуществить перенос исходного кода из других систем управления версиями?

Управление проектом и рабочей областью

- Как упорядочить командные проекты?
- Как управлять зависимостями между проектами?
- Что такое рабочая область?
- Как изолировать работу разработчика при помощи рабочей области?
- Как эффективно организовать сопоставление рабочей области?
- Когда следует создавать новый проект, а когда — новую ветвь?
- Как управлять исходным кодом, используемым в нескольких проектах?
- Как управлять двоичными файлами, используемыми в нескольких проектах?
- Как организовать дерево исходного кода?

Отложенные правки

- Что такое отложенные правки?
- Что такое набор отложенных правок?
- Как используются отложенные правки?
- Как использовать отложенные правки для архивации?
- Для чего нужно извлекать набор отложенных правок?

Доступ к системе управления версиями

- Что такое MSSCCI Provider и когда он используется?
- Какие еще интегрированные среды разработки поддерживаются TFS?
- Когда следует использовать Team Foundation Server Power Tool?
- Каковы наиболее распространенные сценарии расширения системы управления версиями?
- Как работать с системой управления версиями из командной строки?

Что такое MSSCCI Provider и когда он используется?

Поставщик интерфейса системы управления исходным кодом Microsoft® (Microsoft Source Code Control Interface, MSSCCI) используется для интегрированного управления версиями при работе с продуктами, не поддерживающими Microsoft Visual Studio® Team Explorer. Например, если вы работаете в Visual Studio 6.0, то можете использовать клиент MSSCCI или командную строку для взаимодействия с системой Microsoft Visual Studio Team System (VSTS) Team Foundation Version Control.

Ниже приведен список клиентов, способных работать напрямую с Team Foundation Version Control при помощи провайдера MSSCCI:

- Microsoft Visual Studio .NET 2003.
- Microsoft Visual C++® 6 Service Pack 6 (SP6).
- Microsoft Visual Basic® 6.0 SP6.
- Microsoft Visual FoxPro® 9.0 SP1.
- Microsoft Access 2003 SP2.
- Microsoft SQL Server™ Management Studio.
- Sparx Systems Enterprise Architect 6.1.
- Sybase PowerBuilder 105.
- Toad for SQL Server 2.0.

Работа MSSCCI в Visual Studio 2005 имеет следующие отличия от системы Team Foundation Version Control:

- в процессе отладки выполняется операция GetLatest;
- при извлечении файла применяется исключающая блокировка возврата после правки;
- команды **Open from Source Control** и **Save to Source Control** ведут себя так же, как и в Microsoft Visual SourceSafe®.

Дополнительные ресурсы

- Дополнительную информацию о MSSCCI вы найдете на сайте Microsoft MSDN® в статье «The Microsoft Source-Code Control Interface» по адресу *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcug98/html/_asug_the_microsoft_source_code_control_interface.asp*.
- Дополнительную информацию о провайдере MSSCCI вы найдете в статье «Update on the TFS MSSCCI Provider» по адресу *http://blogs.msdn.com/bharry/archive/2006/03/24/559876.aspx*.
- Надстройка MSSCCI разработана в TFS Power Tool, но официально Microsoft не поддерживается. Вы можете загрузить ее из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyId=87E1FFBD-A484-4C3A-8776-D560AB1E6198&displaylang=en*.

Какие еще интегрированные среды разработки поддерживаются TFS?

Систему Team Foundation Server можно использовать из любой версии Visual Studio 2005, где установлен Team Explorer. Кроме того, вы вольны запускать Team Explorer параллельно с любыми другими (не Visual Studio 2005) интегрированными средами разработки, чтобы работать с командными проектами и управлять рабочими элементами.

Для приведенных ниже клиентов имеются интеграционные решения, поставляемые сторонними производителями:

- Eclipse;
- клиент Linux ;
- клиент Apple Macintosh ;
- веб-клиент Hypertext Markup Language (HTML).

Чтобы получить доступ к системе управления версиями TFS из среды разработки Eclipse, а также из клиентов Linux или Macintosh, установите пакет клиентских приложений Teamprise (*http://www.teamprise.com/*).

Чтобы получить доступ к системе управления версиями TFS только для чтения через Интернет, воспользуйтесь приложением Team System Web Access (*http://msdn2.microsoft.com/en-us/teamsystem/bb676728.aspx*).

Дополнительные ресурсы

- Дополнительную информацию об использовании Team Explorer вы найдете в статье «Working with Older Visual Studio Projects or Other Code Projects» по адресу *http://msdn2.microsoft.com/en-us/library/ms242912(vs.80).aspx*.
- Дополнительные сведения о Teamprise вы найдете по адресу *http://www.teamprise.com*.
- Дополнительные сведения о Team System Web Access вы найдете по адресу *http://msdn2.microsoft.com/en-us/teamsystem/bb676728.aspx*.

Когда следует использовать Team Foundation Server Power Tool?

Функциональные возможности Team Foundation Power Tool (TFPT) недоступны из пользовательского интерфейса Visual Studio 2005. Например, TFPT используется для работы в автономном режиме, а также для выполнения операций отката — отмены возврата набора изменений после правки. Инструмент TFPT может применяться для выполнения следующих операций:

- **Извлечение отложенных правок** Операция извлечения отложенных правок (unshelve), поддерживаемая TFPT, не допускает объединения отложенных и локальных правок. Допустим, вы при помощи TFPT извлекаете изменение из набора, а у соответствующего элемента локальной

рабочей области имеется также и ожидающее изменение, TFPT может объединить изменения путем трехстороннего слияния, при условии что и отложенное, и ожидающее изменения являются правками кода.

- **Откат** Возможность отмены возврата набора изменений после правки не поддерживается TFS напрямую. Однако с помощью команды rollback инструмента TFPT вы можете попытаться отменить любые изменения из конкретного набора. Отменить можно не все изменения, но в большинстве сценариев откат работает.
- **Автономная работа** Онлайн-инструмент TFPT позволяет на протяжении некоторого времени работать без подключения к серверу. Он располагает функциональными возможностями, информирующими сервер об изменениях, внесенных в локальную рабочую область.
- **Получение набора изменений** Команда GetCS позволяет получить все объекты, включенные в набор изменений, версию которого вы указали. Это полезно, если ваш коллега вернул в систему нужное вам изменение, но вы не хотите полностью обновлять рабочую область до последней версии.
- **Удаление ожидающих правок** Команда UU (Undo Unchanged) удаляет ожидающие правки из файлов, которые на самом деле не редактировались. Это полезно, когда вы извлекаете для правки большое количество файлов, но реально вносите изменения только в некоторые из них. Команда UU позволяет отказаться от правки тех файлов, которые фактически не были изменены. Команда сравнивает хеш файлов из локальной рабочей области с хешем файлов, находящихся на сервере, и выясняет, был ли файл действительно изменен.

Каждая из этих команд запускается из командной строки при помощи Tftp.exe.

Дополнительные ресурсы

- Загрузить инструмент TFPT можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyID=7324c3db-658d-441b-8522-689c557d0a79&DisplayLang=en*.
- Форум, посвященный обсуждению TFPT, находится по адресу *http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1*.

Каковы наиболее распространенные сценарии расширения системы управления версиями?

Наиболее распространенный сценарий расширения функциональности системы управления исходным кодом состоит в настройке политики возврата кода после правки с целью соблюдения стандартов. Чтобы создать надстройки пользовательских политик, перечисляемые в диалоговом окне Add Checkin Policy, применяются функции расширения из комплекта Visual Studio Team Foundation Server Software Development Kit. Инструментарий TFS SDK мож-

но загрузить по адресу *http://go.microsoft.com/fwlink/?linkid=68586*.

Чтобы клиенты помимо Visual Studio 2005 могли работать с системой управления версиями Team Foundation Version Control, можно также написать интеграционный слой, хотя к этому способу прибегают нечасто.

Дополнительные ресурсы

- Подробнее о том, как настроить политику возврата после правки, читайте в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- Чтобы посмотреть пример кода, который запрещает возврат правок с определенными вариантами кодирования, читайте статью «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- Чтобы посмотреть пример кода, который обязывает вводить комментарии при возврате после правки, читайте статью «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- Чтобы узнать, как зарегистрировать новую политику возврата после правки, читайте статью «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.
- Загрузить TFS SDK можно из источника, расположенного по адресу *http://go.microsoft.com/fwlink/?linkid=68586*.
- Дополнительную информацию о расширении Team Foundation Version Control вы найдете в статье «Walkthru: The Version Control Object Model» по адресу *http://msdn2.microsoft.com/en-us/library/bb187335(VS.80).aspx*.

Как работать с системой управления версиями из командной строки?

В систему Team Foundation Server включен инструмент командной строки Tf.exe, который позволяет выполнять ряд действий по управлению исходным кодом. В частности, из командной строки можно настраивать выполнение задач по расписанию при помощи планировщика задач Microsoft Windows® Task Scheduler.

Если путь и другие переменные среды заданы правильно, запускайте инструмент из окна командной строки Visual Studio 2005. Чтобы задать путь и переменные, запустите пакетный файл Vsvars32, который обычно располагается в папке *Диск*:\Program Files\Microsoft Visual Studio 8\Common7\Tools. В командной строке поддерживается большинство команд системы управления исходным кодом, включая Checkin, Checkout, Get, History, Shelve, Branch, Merge, Label, Status, Undelete и Undo.

Ниже приведены наиболее распространенные операции, выполняемые из командной строки:

- Синхронизация файлов локального компьютера с файлами сервера: **tf get**.
- Добавление файлов на сервер: **tf add**.
- Извлечение файла для редактирования: **tf checkout**.
- Возврат ожидающих изменений: **tf checkin**.
- Извлечение определенного набора изменений с сервера: **tf get /version**.

Некоторые операции можно выполнить только из командной строки:

- Удаление рабочей области другого пользователя: **tf workspace /delete**.
- Удаление файлов, возвращенных другим пользователем: **tf undo**.
- Снятие блокировки, установленной другим пользователем: **tf lock**.
- Определение области действия меток: **tf label**.
- Слияние без основы: **tf merge**.

Дополнительные ресурсы

- Дополнительную информацию о работе с командами Tf.exe вы найдете в статье «MSDN: Walkthrough: Working with Team Foundation Source Control from Command Line» по адресу *http://msdn2.microsoft.com/en-us/library/zthc5x3f.aspx*.
- Дополнительную информацию о командах, доступных только из командной строки, вы найдете в статье «Operations Available Only From the Command-Line (Team Foundation Source Control)» по адресу *http://msdn2.microsoft.com/en-us/library/ms194957(VS.80).aspx*.

Администрирование

- Как предоставить разрешение на доступ к файлу в папке, имеющей наследуемые разрешения?
- Что делать, если разработчик покидает проект?
- Как управлять стажерами и другими разработчиками, которым я не доверяю возвращать правки на сервер?
- Как следует изменять разрешения после отправки приложения заказчику?

Как предоставить разрешение на доступ к файлу в папке, имеющей наследуемые разрешения?

Чтобы задать разрешения, щелкните правой кнопкой папку или файл в окне Source Control Explorer, выберите команду **Properties** и перейдите на вкладку **Security**. Далее выберите пользователя или группу, для которых хотите изменить разрешения, и отредактируйте разрешения в списке **Permissions**.

Задавать разрешения можно также из командной строки с помощью подкоманды **Permission**, входящей в состав утилиты TF.

Система Team Foundation Source Control позволяет управлять доступом для групп Windows Groups, Windows Users и Team Foundation Groups. Разрешения могут наследоваться из папки более высокого уровня или задаваться явно.

Разрешения имеют вид разрешения (Allow) или запрета (Deny). Запрет всегда перекрывает разрешение, даже если Deny является наследуемой формой, а форма Allow объявлена явно. Фактические разрешения на доступ к объекту для пользователей или групп определяются комбинацией наследуемых и явных разрешений. Поскольку Deny всегда перекрывает Allow, можно не отключать наследование, чтобы, например, запретить возврат после правки.

Будьте внимательны при отключении наследования. Сначала следует установить необходимые явные разрешения и убедиться в наличии необходимых разрешений у вашей учетной записи. Если отключить наследование без предварительной подготовки, вы рискуете закрыть самому себе доступ к файлу или папке. Для исправления ситуации потребуется вмешательство администратора TFS, являющегося также администратором компьютера уровня приложения. Конструкцией системы предусмотрено, что администраторы локального уровня приложения не могут полностью закрыть себе доступ.

Дополнительные ресурсы

- Дополнительную информацию о разрешениях вы найдете в статье «Team Foundation Server Default Groups, Permissions, and Roles» on MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms253077.aspx*.
- Дополнительную информацию о разрешениях вы найдете в статье «Source Control Security Rights and Permissions» on MSDN по адресу *http://msdn2.microsoft.com/en-us/library/ms181761.aspx*.

Что делать, если разработчик покидает проект?

Когда ваш проект покидает разработчик, обязательно удалите его рабочую область. Это действие одновременно удалит все ожидающие изменения разработчика и отменит все установленные им блокировки.

> **Примечание** Если в командном проекте включена исключающая блокировка, вы не сможете отменить блокировки, не отменив изменения.

Чтобы узнать, какие файлы заблокированы пользователем, запустите следующую команду:

```
tf workspaces /owner:domain\devuser /computer:* /server:servername
```

Для удаления рабочей области и снятия блокировок выполните команду

```
Tf workspace /delete workspacename;domain\devuser /s:servername
```

Дополнительные ресурсы

- Дополнительную информацию о том, что следует предпринять после ухода разработчика из проекта, вы найдете в статье «How to: Clean Up Files When Users Leave» по адресу *http://msdn2.microsoft.com/en-us/library/ms194958(VS.80).aspx*.
- Дополнительную информацию о команде Workspace вы найдете в статье «Workspace Command» по адресу *http://msdn2.microsoft.com/en-us/library/y901w7se(VS.80).aspx*.

Как управлять стажерами и другими разработчиками, которым я не доверяю возвращать правки на сервер?

Если в вашей команде присутствуют разработчики, которым вы пока не доверяете, например, новички или стажеры, откажите им в разрешении на возврат кода в дерево после правки. Прежде чем отключить наследование, убедитесь, что задали необходимые разрешения (включая разрешения для собственной учетной записи). Вместо возврата напрямую, неопытные разработчики смогут создавать ожидающие изменения, а затем сохранять их на сервере в качестве отложенных правок. Затем более опытный разработчик извлечет правки, просмотрит их и при необходимости выполнит возврат.

Дополнительные ресурсы

- Дополнительную информацию об удалении разрешений вы найдете в статье «How to: Remove Access to Source Control Files» по адресу *http://msdn2.microsoft.com/en-us/library/ms400718(VS.80).aspx*.

Как следует изменять разрешения после отправки приложения заказчику?

Когда ветвь переведена в режим обслуживания, например, после отправки ПО заказчику, отключите наследуемые разрешения, чтобы заблокировать дерево. Затем вы сможете предоставить отдельным пользователям разрешения на внесение ожидающих изменений и возврат кода после правки на случай потребности в заплатках.

Дополнительные ресурсы

- Дополнительную информацию об удалении разрешений вы найдете в статье «How to: Remove Access to Source Control Files» по адресу *http://msdn2.microsoft.com/en-us/library/ms400718(VS.80).aspx*.

Ветвление, метки и слияние

- Когда использовать метки?
- В чем отличие меток TFS от меток VSS?

- Что такое ветвление?
- Когда следует выполнять ветвление?
- Когда не следует выполнять ветвление?
- Как использовать ветвление для выпуска приложения?
- Как использовать ветвление для поддержки приложения?
- Как при помощи ветвления сократить количество конфликтов между командами?
- Как при помощи ветвления сократить количество конфликтов в компонентах?
- Как наиболее эффективно провести ветвление и слияние?
- В чем разница между ветвлением и метками?
- Что представляет собой модель ветвления в пространстве путей?
- Что такое модель распространения TFS?
- Как выполнить слияние двух ветвей?
- Можно ли выполнить слияние между командными проектами?
- Что такое слияние без основы?
- Что такое модель продвижения кода?
- В чем отличие между логическим и физическим представлениями ветвей?
- Как часто следует выполнять слияния?

Когда использовать метки?

Используйте метки для группировки файлов и папок с целью выполнения над ними каких-либо действий — ветвления, слияния, сравнивания или извлечения. Метка закрепляет за объектом маркер, к которому можно обратиться позднее при выполнении перечисленных операций.

Система Team Foundation Build автоматически помечает версии файлов, связанные каждой создаваемой сборкой.

Примечание Если вы пока не уверены в целесообразности ветвления, пометьте набор файлов, а позже при необходимости создайте на основе этой метки ветвь.

Дополнительные ресурсы

- Дополнительную информацию о метках вы найдете в статье «Working with labels» по адресу *http://msdn2.microsoft.com/en-us/library/ms181439(VS.80).aspx*.
- Дополнительную информацию о применении меток вы найдете в статье «How to: Apply Labels» по адресу *http://msdn2.microsoft.com/en-us/library/ms181440(VS.80).aspx*.

В чем отличие меток TFS от меток VSS?

Метки Team Foundation Server отличаются от меток VSS довольно сильно. Поскольку метки VSS являются метками времени и присваиваются, как правило, всему дереву VSS или его части, система отображает метки в хронологической последовательности вместе с историей файлов. Все, что стоит в списке до помеченного файла, включается в метку, а все, что стоит после — не включается.

Метки TFS не являются метками времени. Они связывают между собой конкретные версии набора файлов исходного кода. Типичный сценарий использования меток — отметка ежедневных сборок. Это позволяет без труда извлечь набор файлов исходного кода, соответствующий определенной сборке, например, при применении исправления.

Дополнительные ресурсы

- Сравнительную характеристику меток TFS и VSS вы найдете в статье «Comparing SourceSafe Labels to Team Foundation Server Labels» по адресу *http://blogs.vertigosoftware.com/teamsystem/archive/2006/05/03/Comparing_SourceSafe_Labels_to_Team_Foundation_Server_Labels.aspx*.

Что такое ветвление?

Ветвление (branching) направлено на разделение набора файлов по различным вариантам разработки. Система Team Foundation Server поддерживает ветвление и сложное слияние, позволяющее объединять файлы из различных ветвей. Например, ветвление применяется для изоляции крупных выпусков приложения. Выделив в отдельную ветвь готовый выпуск приложения, вы значительно облегчите его сопровождение. Слияние позволяет выборочно вносить исправления в обе ветви. Ветвление также позволяет изолировать работу команд, создающих различные компоненты ПО одновременно разрабатывать несколько версий одного и того же ПО.

При ветвлении TFS не создает отдельные копии содержимого файла, поэтому само ветвление не требует значительного дополнительного пространства в БД системы управления исходным кодом. Ветвь содержит указатель в БД на исходный код и список отличий текущего содержимого от базовой версии.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.

- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Когда следует выполнять ветвление?

Ветви позволяют нескольким разработчикам изолированно работать над одними и теми же файлами. Поскольку ветвление увеличивает накладные расходы и требует усилий по разрешению конфликтов, не следует выполнять ветвление, пока оно вам действительно не понадобится. При необходимости вы можете пометить сборку и выполнить ветвление позднее.

Для принятия решения по поводу создания ветви ответьте на один вопрос — что больше: затраты на разрешение конфликтов слияния в реальном времени или накладные расходы на разрешение конфликтов слияния между ветвями?

Общие основания для создания ветвей таковы:

- **Выпуск** Изоляция нескольких параллельных выпусков.
- **Обслуживание** Сопровождение ранее выпущенной сборки.
- **Компонент** Изоляция работы над экспериментальными или рискованными компонентами, которые могут внести неустойчивость в остальной проект.
- **Команда** Изоляция подгрупп, чтобы оградить их от нежелательного влияния изменений друг друга. Изолированные группы могут стремиться к различным контрольным вехам. Ветви команд схожи с ветвями компонентов.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Когда не следует выполнять ветвление?

Не выполняйте ветвление, пока членам вашей команды разработчиков действительно не придется параллельно работать с одним и тем же набором файлов. Если сомневаетесь, для начала пометьте сборку, чтобы позднее при необходимости создать из помеченной сборки ветвь. Слияние ветвей — ресурсоемкая операция, особенно, когда между ветвями имеются серьезные различия.

Проведением слияния и разрешением возникающих при этом конфликтов должен заниматься квалифицированный разработчик, а то и не один. Полученный в результате код должен всесторонне тестироваться, так как зачастую некорректное разрешение конфликта при слиянии выводит из равновесия всю сборку.

Особую сложность представляет слияние поперек иерархии ветвей, которое требует ручного разрешения многих конфликтов, которые в других обстоятельствах разрешиться автоматически.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как использовать ветвление для выпуска приложения?

Создайте ветвь Release, когда будете готовы зафиксировать сборку перед выпуском приложения. Далее приводится пример структуры ветвей после создания ветви выпуска:

- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.
- **Release 1** — Ветвь выпуска.
  - **Source**.

  Учитывайте следующие рекомендации по работе с ветвью выпуска:
- **Когда выполнять ветвление** Подготовившись к выпуску, соберите все в главную ветвь (Main), а затем создайте ветвь выпуска (Release), целью которой будет стабилизация приложения перед выпуском.

- **Когда не следует выполнять ветвление** Если каждому выпуску соответствует отдельный проект TFS, используйте для продолжения разработки новый проект, не выполняя ветвление текущего проекта.
- **Разрешения на доступ к ветви:**
  - **Перед выпуском** — разрешения на чтение и запись для всех разработчиков.
  - **После выпуска** — разрешения на чтение и запись разработчикам, принимающим участие в работе над исправлениями, всем остальным — разрешения только на чтение.
- **Частота сборок в ветви** По мере необходимости.
- **Тесты в ветви** Прекращаются после выпуска.

Ветвь Release следует использовать только для внесения целевых исправлений и изменений, требуемых для стабилизации сборки перед выпуском. Разработка следующих версий приложения может параллельно продолжаться в ветвях сборки Development или Integration. Благодаря этому в версию, готовящуюся к выпуску, можно будет до даты выпуска вносить все возникающие стабилизирующие изменения. После создания окончательной сборки выпуска вы можете провести слияние изменений из ветви Release в ветви сборки Development или Integration.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как использовать ветвление для поддержки приложения?

Для поддержки ранее выпущенных сборок используйте ветви сопровождения. Далее приводится пример структуры ветвей после создания ветви сопровождения:

- **Main** — главная ветвь сборки.
  - **Source**.

- **Другие папки ресурсов**.

- **Releases** — контейнер для ветвей выпусков.
  - **Release 1** — ветвь сопровождения.
    - **Source**.

Учитывайте следующие рекомендации по работе с ветвью сопровождения:

- **Когда выполнять ветвление** После выпуска. Проводите обслуживание выпуска в ветви, находящейся в папке Releases.
- **Когда не следует выполнять ветвление** Если сопровождение выпуска не планируется, помечайте сборки старых выпусков с помощью меток и продолжайте работать в основной ветви.
- **Разрешения на доступ к ветви:**
  - разрешения на чтение и запись для разработчиков, принимающих участие в работе над исправлениями;
  - всем остальным — разрешения только на чтение.
- **Частота сборок в ветви** По мере необходимости.
- **Тесты в ветви** Прекращаются после выпуска.

Ветви сопровождения используются для поддержки предыдущих версий приложения. Вы можете перенести изменения в главную ветвь сборки или оставить их исключительно в ветви сопровождения.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как при помощи ветвления сократить количество конфликтов между командами?

Используйте ветви команды для совершенствования интеграции и стабилизации критических изменений между командами. Далее приводится пример структуры ветвей после создания ветвей команд:

- **Development** — Контейнер для ветвей команд.
  - **Team 1** — ветвь команды.
    - **Source**.
  - **Team 2** — ветвь команды.
    - **Source**.
- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.

Учитывайте следующие рекомендации по работе с ветвями команд:

- **Когда выполнять ветвление** Если разные команды работают с одними и теми же файлами или компонентами, создайте ветви для их изоляции друг от друга. Внутри ветвей команд можно создавать ветви компонентов.
- **Когда не следует выполнять ветвление** Если дерево исходного кода упорядочено по компонентам и вы уверены, что в командах не возникнет критических изменений интерфейса или большого количества конфликтов, вероятно, у вас нет необходимости создавать ветви команд.
- **Разрешения на доступ к ветви:**
  - разрешения на чтение и запись для разработчиков из данной команды;
  - всем остальным — разрешения только на чтение.
- **Частота сборок в ветви** Непрерывная интеграция.
- **Тесты в ветви** Испытания компонентов и быстрое тестирование качества кода.

Командные ветви применяются, чтобы позволить командам решать задачи разработки параллельно. Смысл ветвления состоит в изолировании команд от критических изменений, вносимых другими командами. Благодаря этому команды могут двигаться в направлении различных контрольных целей. Вся активная разработка проводится в этом случае в ветвях команд с последующим слиянием в ветвь Main.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как при помощи ветвления сократить количество конфликтов в компонентах?

Используйте ветви компонентов для совершенствования интеграции и стабилизации критических изменений в компонентах. Далее приводится пример структуры ветвей после создания ветвей компонентов:

- **Development** — контейнер для ветвей компонентов.
  - **Feature A** — ветвь компонента.
    - **Source**.
  - **Feature B** — ветвь компонента.
    - **Source**.
  - **Feature C** — ветвь компонента.
    - **Source**.
- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.

Учитывайте следующие рекомендации по работе с ветвями компонентов:

- **Когда выполнять ветвление** Команды, занятые разработкой компонентов, часто работают с перекрывающимися файлами исходного кода, провоцируя ошибки сборки и конфликты при возврате после правки. Если у вас возникают подобные проблемы, рассмотрите возможность ветвления по компонентам, чтобы изолировать их разработку. Ветвление можно выполнить как в папке Main, так и в папках команд.
- **Когда не следует выполнять ветвление** Если вы используете только непрерывную интеграцию, при этом ежедневные сборки достаточно стабильны, вам, возможно, нет смысла нести дополнительные расходы, связанные с ветвями компонентов.
- **Разрешения на доступ к ветви:**
  - разрешения на чтение и запись для разработчиков, работающих над данным компонентов;
  - всем остальным — разрешения только на чтение.
- **Частота сборок в ветви** Непрерывная интеграция.
- **Тесты в ветви** Испытания компонентов и быстрое тестирование качества кода.

Использование ветвей позволяет вести параллельную разработку каждого компонента. Вся активная разработка ведется в ветвях компонентов, а последующая интеграция кода осуществляется в ветви Main.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

Как наиболее эффективно провести ветвление и слияние?

Выполняя ветвление, руководствуйтесь следующими рекомендациями:

- Структурируйте деревья ветвей так, чтобы последующие слияния выполнялись вдоль иерархии (вверх и вниз по дереву ветви), а не поперек. Ветвление поперек иерархии требует последующего слияния без основы, при котором большее количество конфликтов придется разрешать вручную.
- Иерархия ветвей основана родительских и дочерних ветвях и может отличаться от физической структуры исходного кода на диске. Планируя слияния, опирайтесь на логическую, а не на физическую структуру ветвей.
- Не выполняйте ветвление слишком глубоко — это приводит к задержкам при слиянии. Глубокая структура ветвей может привести к тому, что для распространения изменений из дочерней ветви в главную ветвь потребуется много времени. Все это негативно сказывается на графике выполнения проекта и увеличивает время, затрачиваемое на исправление ошибок.
- Выполняйте ветвление на высоком уровне. Включайте в него файлы конфигурации и исходного кода.
- Развивайте структуру ветвей согласно меняющимся нуждам.
- Не выполняйте ветвление, если команде разработчиков не требуется параллельно работать над одним и тем же набором файлов. Если сомневаетесь, для начала пометьте сборку, чтобы позднее при необходимости создать из нее ветвь. Слияние ветвей — затратная операция, особенно, когда между ветвями существуют серьезные отличия.
- Для выполнения слияния и устранения конфликтов требуется один или несколько разработчиков. Полученный в результате код должен быть тщательно протестирован, ибо нередко слияние выполняется некорректно и дестабилизирует сборку.

- Особую сложность представляет слияние поперек иерархии ветвей. Оно требует ручного разрешения многих конфликтов, которые при других обстоятельствах могли бы быть разрешены автоматически.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о методах ветвления и слияния в Visual Studio 2005 читайте в статье «Branching and Merging Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181423(VS.80).aspx*.

В чем разница между ветвлением и метками?

Ветви предназначены для разделения набора файлов по различным путям разработки, чтобы над разными вариантами разработки можно было работать параллельно. Например, ветви используются для поддержки и обслуживания ранее выпущенных версий приложения или для изоляции рискованных изменений кода с целью минимизации их влияния. Если же вам нужно изолировать одного разработчика, используйте рабочие области, чтобы избежать сложностей с ветвлением и слиянием.

Метки используются для маркировки файла или набора файлов, чтобы позднее файл или набор можно было без труда извлечь. Метки используются, например, для маркировки ежедневных сборок. Если со сборкой возникли проблемы, вы легко извлечете ее.

Дополнительные ресурсы

- Дополнительную информацию о команде **branch** вы найдете в статье «Branch Command» по адресу *http://msdn2.microsoft.com/en-us/library/d73s8b27(VS.80).aspx*.
- Дополнительную информацию о команде **label** вы найдете в статье «Label Command» по адресу *http://msdn2.microsoft.com/en-us/library/9ew32kd1(VS.80).aspx*.

Что представляет собой модель ветвления в пространстве путей?

Ветви TFS создаются на базе путей хранилища, напоминая операцию копирования файлов. Этим они отличаются от механизма «pin-and-share», ис-

пользуемого в Visual SourceSafe. Подход на базе пространства путей облегчает сопровождение старых версий ПО, поскольку упрощает перенос изменений из одной ветви в другую или внесение изменений сразу в обе ветви.

При ветвлении TFS не создает отдельные копии содержимого файла, поэтому само ветвление не требует значительного дополнительного пространства в БД системы управления исходным кодом. Ветвь содержит указатель в БД на исходный код и список отличий текущего содержимого от базовой версии.

Дополнительные ресурсы

- Вводный курс по ветвлению и слиянию можно прочитать в статье «Branching and Merging Primer» по адресу *http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx*.

Что такое модель распространения TFS?

Моделью распространения (promotion model) называется способ передачи изменения из одной ветви в другую. Система Visual SourceSafe также позволяла выполнять ветвление и слияние, но на практике клиенты использовали в качестве простейшего средства распространения механизм «pin-and-share». В TFS изменения распространяются путем слияния файлов из разных ветвей.

Дополнительные ресурсы

- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Подробнее о различиях между Visual SourceSafe и Team Foundation Server Version Control читайте в статье «Visual Source Safe vs. Team Foundation Server» по адресу *http://msmvps.com/blogs/vstsblog/archive/2005/07/02/56401.aspx*.

Как выполнить слияние двух ветвей?

Для выполнения слияния можно использовать соответствующую функцию Source Control Explorer или инструмент командной строки **merge**. Слияние может быть основано на наборе изменений, метке, дате или версии.

Если ваша структура ветвей охватывает более одного уровня, имейте в виду, что единовременно допускается выполнять слияние только одного уровня. Рассмотрим в качестве примера следующую структуру дерева исходного кода:

- **Development** — контейнер для изолированных ветвей разработки.
  - **Feature A**.

    - **Source**.
  - **Feature B**.
    - **Source**.
- **Main** — главная ветвь сборки.
  - **Source**.
  - **Другие папки ресурсов**.
- **Releases** — контейнер для ветвей выпусков.
  - **Release 1**.
    - **Source**.
  - **Release 2** — заблокированный текущий выпуск.
    - **Source**.

Допустим, изменен исходный код в ветви **Release 2**. В конце рабочего дня вам необходимо выполнить слияние изменений в одну из ветвей компонентов, например, для переноса исправлений в актуальный код компонента. Нелегко будет выполнить слияние напрямую из ветви **Release 2** в ветвь **Feature B**. Вместо этого выполните перенос из **Release 2** в ее логическую родительскую ветвь **Main**, а затем из **Main** в ее логическую дочернюю ветвь **Feature B**. Если вам нужно выполнить слияние между ветвями, не имеющими прямых отношений по вертикали, следует произвести слияние без основы.

Дополнительные ресурсы

- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о слиянии без основы вы найдете в описании параметра **/baseless** в статье «Merge Command» по адресу *http://msdn2.microsoft.com/en-us/library/bd6dxhfy(VS.80).aspx*.

Можно ли выполнить слияние между командными проектами?

Да, причем, в слиянии между командными проектами нет ничего экзотического. Здесь вполне применимы обычные слияния. Мастер создания командных проектов Team Project Creation Wizard позволяет создать командный проект как ветвь другого проекта.

Дополнительные ресурсы

- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

Что такое слияние без основы?

Команда **branch** создает связь между двумя папками, которая используется для выполнения слияний между ветвями. У ветвей, не являющихся прямыми родителями или потомками по отношению к целевой папке слияния, нет такой связи. Согласно определению из MSDN, *слияние без основы* (baseless merge) представляет собой слияние ветвей в отсутствии базовой версии. Это позволяет производить слияние файлов и папок, не имеющих связей типа «ветвь-слияние». Слияние без основы происходит с гораздо большим количеством конфликтов, чем обычное слияние. Однако после слияния без основы необходимые связи устанавливаются, дальнейшие слияния уже будут иметь основу, и потому количество конфликтов сократится.

Слияния без основы можно выполнить только из командной строки. Даже после проведения первого слияния без основы, когда связи между ветвями уже будут установлены, последующие слияния по-прежнему должны будут выполняться из командной строки.

Дополнительные ресурсы

- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о слиянии без основы вы найдете в описании параметра **/baseless** в статье «Merge Command» по адресу *http://msdn2.microsoft.com/en-us/library/bd6dxhfy(VS.80).aspx*.

Что такое модель продвижения кода?

Модель *продвижения кода* (code promotion) — это стратегия использования ветвей для продвижения кода по этапам стабилизации. Например, активная разработка осуществляется в ветви Development, в ветви Main производятся сборка и тестирование, ветвь Production используется вашей командой для стабилизации итогового выпуска.

Дополнительные ресурсы

- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.

В чем отличие между логическим и физическим представлениями ветвей?

Физическое представление — это иерархия построения дерева исходного кода, например:
$/MyTeamProject

- **Development**.
  - Source.
- **Main**.
  - Source.
- **Releases**.
  - **Release1**.
    - Source.
  - **Release2**.
    - Source.

Исходный код содержится непосредственно в ветвях Development, Main, Release1 и Release2. Папка Releases содержит несколько ветвей.

Логическое представление — это иерархия родительских и дочерних ветвей в той последовательности, в которой они были созданы, например:

- **Main**.
  - **Development**.
  - **Release1**.
  - **Release2**.

Важность логической структуры заключается в том, что именно она определяет наиболее простой способ слияния — вверх и вниз по логической иерархии. Для выполнения слияния поперек иерархии вы должны произвести слияние без основы, что влечет за собой использование командной строки и гораздо большее количество конфликтов.

Дополнительные ресурсы

- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о слиянии без основы вы найдете в описании параметра **/baseless** в статье «Merge Command» по адресу *http://msdn2.microsoft.com/en-us/library/bd6dxhfy(VS.80).aspx*.

Как часто следует выполнять слияния?

Частота слияний зависит от структуры ветвей. Общий пример структуры ветвей приводится ниже:

- **Development** — контейнер изолированных ветвей разработки.
  - **Feature A**.

- ▲ **Source**.
- ❑ **Feature B**.
  - ▲ **Source**.
- ■ **Main** — главная ветвь сборки.
  - ❑ **Source**.
  - ❑ **Другие папки ресурсов**.
- ■ **Releases** — контейнер для ветвей выпусков.
  - ❑ **Release 1**.
    - ▲ **Source**.
  - ❑ **Release 2** — заблокированный текущий выпуск.
    - ▲ **Source**.

В данном случае, активная разработка происходит в ветвях Feature. Выбирая частоту слияний, вы должны ответить на два вопроса:

1. Как часто команде, занимающейся разработкой компонента, следует переносить изменения из ветви Main в ветвь Feature?
2. Как часто команде, занимающейся разработкой компонента, следует переносить изменения в ветвь Main?

Ниже приводится пример проверенной временем стратегии, которая позволяет сократить число некорректных изменений и время переноса:

1. Команда, занимающаяся разработкой компонента, переносит изменения в ветвь Main при завершении работы над компонентом.
2. Команда, занимающаяся разработкой компонента, регулярно переносит изменения из ветви Main, чтобы в финале обеспечить совместимость всех компонентов. В идеале следует выполнять слияние один раз в день и не реже, чем раз в два дня.

Дополнительные ресурсы

- Дополнительную информацию о слиянии вы найдете в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.

Возврат после правки и политики возврата

- Что такое набор изменений?
- Что такое политика возврата после правки?
- Как перекрыть политику возврата?
- Как гарантировать соблюдение политики?
- Как пользоваться проверкой возврата после правки?
- Приведет ли к нарушению синхронизации переименование или удаление файлов на диске?

- Как работает система автоматического разрешения конфликтов?
- Как разрешать конфликты вручную?
- Как избежать конфликтов?

Что такое набор изменений?

Набор изменений (changeset) представляет собой совокупность всех изменений, связанных с определенной операцией возврата после правки. Все изменения из набора поштучно применяются к TFS во время возврата набора. Наборы изменений обозначаются уникальными последовательно возрастающими номерами.

Позже вы можете извлечь набор изменений для просмотра измененных файлов, комментариев и связанных рабочих элементов. Вы также вольны извлечь версии файлов, связанных с набором, чтобы просмотреть их, протестировать или выполнить сборку с использованием старой версии.

Дополнительные ресурсы

- Дополнительную информацию о наборах изменений вы найдете в статье «How to: Retrieve Old Versions of Files from Changesets» по адресу *http://msdn2.microsoft.com/en-us/library/ms181416(VS.80).aspx*.

Что такое политика возврата кода после правки?

Политики возврата после правки применяются с целью гарантированного соблюдения стандартов программирования. Например, вы можете потребовать от каждого разработчика перед возвратом кода провести его статический анализ.

По умолчанию в VSTS имеются следующие политики возврата после правки:

- **Code Analysis** Требует проведения анализа кода перед возвратом.
- **Test Policy** Требует выполнения определенных тестов перед возвратом.
- **Work Items** Требует связать с возвратом один или несколько рабочих элементов.

Вы вольны также создать собственную политику возврата для выполнения проверок, не предусмотренных по умолчанию. В частности, можно запретить определенные варианты кодирования, например, вызовы запрещенных функций API, запретить возврат кода без комментариев, а также проверять код на соответствие правилам программирования, принятым в вашей организации.

Дополнительные ресурсы

- О настройке политики возврата читайте в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.

- Пример кода с запретом на возврат определенных фрагментов вы найдете в статье «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- Пример кода с запретом на возврат кода без комментариев вы найдете в статье «Sample Checkin Policy: Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.
- О том, как зарегистрировать созданную вами политику, читайте в статье «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Как перекрыть политику возврата?

Для перекрытия политики возврата кода после правки нужно установить флажок **Override policy failure and continue check-in**. Перекрыть политику возврата может любой член команды, у которого есть разрешение на возврат кода после правки.

Чтобы своевременно узнавать о случаях перекрытия политики членами команды, воспользуйтесь службой событий Team Foundation Eventing Service, чтобы отслеживать события возврата после правки.

Дополнительные ресурсы

- Дополнительную информацию о перекрытии политики возврата после правки вы найдете в статье «How to: Override a Check-in Policy» по адресу *http://msdn2.microsoft.com/en-us/library/ms245460(VS.80).aspx*.
- Дополнительную информацию о службе событий Team Foundation Eventing Service вы найдете в статье «Eventing Service» по адресу *http://msdn2.microsoft.com/en-us/library/bb130154(vs.80).aspx*.

Как гарантировать соблюдение политики?

Система управления версиями не препятствует перекрытию политики, но позволяет фиксировать этот факт:

- при помощи службы событий Team Foundation Eventing Service из Team Foundation Core Services API, отслеживая события возврата;
- написав метод **Notify** для анализа свойств набора изменений, который будет реагировать на факт перекрытия.

Конечно, вы всегда можете с целью обнаружения перекрытия политики просмотреть историю набора изменений вручную.

Дополнительные ресурсы

- О том, как получать по электронной почте уведомление о нарушении политики, вы узнаете из статьи по адресу *http://blogs.infosupport.com/marcelv/archive/2005/10/18/1635.aspx*.

Как пользоваться проверкой возврата после правки?

Политики возврата после правки могут также применяться в качестве системы проверки возвращаемого кода. В комплект TFS входят политики, проверяющие перед возвратом истинность следующих утверждений:

- с изменением связан рабочий элемент;
- пройдены все модульные тесты;
- успешно проведен статический анализ.

Вы вольны установить собственные требования к возврату, создавая новые надстройки политик для контроля за соответствием кода стандартам программирования вашей компании, для запуска проверочных тестов или для проверки любых других вас требований.

Дополнительные ресурсы

- Дополнительную информацию о создании и настройке политик возврата после правки вы найдете в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.

Приведет ли к нарушению синхронизации переименование или удаление файлов на диске?

Да. Если вы случайно удалили файлы или папки, сервер TFS будет по-прежнему считать, что на локальном компьютере присутствуют их последние версии. Это означает, что команда **get latest version** не вернет удаленные файлы обратно на диск. Восстановить утраченные файлы и папки поможет команда **force get**.

Если вам нужно переименовать или удалить файлы, делайте это средствами Source Explorer. При этом локальные изменения будут синхронизированы с сервером.

Как работает система автоматического разрешения конфликтов?

В результате выполнения операций **change**, **merge** или **get** может возникнуть конфликт. Одним из способов его разрешения является автоматическое разрешение конфликтов Visual Studio. Оно сработает только в том случае, если файлы не являются двоичными и не имеют перекрывающихся изменений (изменений, примененных к одной и той же строке кода). В этом случае в новую версию переносятся изменения из обеих версий файла.

Если автоматическое разрешение конфликтов не срабатывает, вы можете на выбор принять изменения только из одной версии файла или провести слияние вручную при помощи графических инструментов сравнения, поставляющихся с VSTS.

Дополнительные ресурсы

- Дополнительные сведения о разрешении конфликтов вы найдете в статье «How to: Resolve Conflicts» по адресу *http://msdn2.microsoft.com/en-us/library/ms181433(VS.80).aspx*.

Как разрешать конфликты вручную?

Для разрешения конфликтов слияния используйте инструментарий слияния Visual Studio. Конфликт, обнаруженный во время слияния, можно разрешить как автоматически, так и вручную. При ручном разрешении конфликта вы вольны сохранить изменения из исходного файла, из конечного файла или разрешить конфликты при помощи инструмента слияния.

Разрешать конфликты приходится во время слияния между ветвями, при извлечении файлов в рабочую область или при возврате новых версий файлов. Существует три типа конфликтов:

- **Версии** Файл изменялся несколькими различными способами, например, в результате редактирования, переименования, удаления или отмены удаления.
- **Коллизия имен файлов** Два или несколько элементов пытаются занять одно положение.
- **Локальная перезапись** Возникает только при выполнении операции **get**, когда происходит попытка перезаписать редактируемый файл.

Большинство конфликтов разрешается автоматически. Единственный вид конфликтов, возникающий при ручном слиянии, — конфликт версий. Чаще всего ручное слияние необходимо в следующих ситуациях:

- В двух ветвях выполнено редактирование файла, затронувшее одинаковые строки кода.
- Производится слияние без основы, при котором TFS не известны связи файлов ветвей.

Инструмент слияния отображает подробности для каждого конфликта и позволяет выбрать изменение, которое следует после слияния сохранить в конечном файле. Вы вольны сохранить изменения источника или целевого файла, объединить изменения или вручную модифицировать окончательную версию, непосредственно отредактировав файл. После разрешения всех конфликтов в файле сохраните окончательную версию как ожидающее изменение, которое предстоит вернуть в целевую ветвь.

Соблюдайте осторожность во время слияния: при этом легко допустить ошибки, которые станут причиной нестабильности сборки. Завершив слияние, скомпилируйте получившийся код и выполните модульные тесты, чтобы своевременно обнаружить серьезные ошибки.

Дополнительные ресурсы

- Дополнительную информацию о разрешении конфликтов вы найдете в статье «How to: Resolve Conflicts» по адресу *http://msdn2.microsoft.com/en-us/library/ms181433(VS.80).aspx*.

Как избежать конфликтов?

Большинство конфликтов разрешается автоматически при помощи системы управления версиями Team Foundation. Ситуации, в которых вам, скорее всего, предстоит выполнять слияние вручную (и которых по этой причине следует избегать), связаны с перекрывающимися изменениями в одних и тех же строках кода или одновременной работой трех и более разработчиков над одним и тем же файлом.

Просмотр ожидающих изменений

Чтобы просмотреть ожидающие изменения в выбранной области, выполните следующие действия:

1. В окне Source Control Explorer щелкните правой кнопкой решение, проект, файл или папку, ожидающие изменения для которых хотите просмотреть.
2. Выберите команду **View Pending Changes**.

Если вы извлекаете файл для редактирования при помощи командной строки, система проинформирует вас о том, какие еще пользователи извлекли этот файл и каковы внесенные ими ожидающие изменения. В приведенном ниже примере для правки извлекается файл Math.cs. Система TFS уведомляет вас, что над этим файлом уже работают два других пользователя (Джеймс и Салли). В уведомлении указан тип ожидающих изменений в рабочих областях обоих пользователей (Edit и Rename), и подтверждается тот факт, что локальная версия файла (в текущей рабочей области) не основана на последней версии из хранилища.

```
c:\dev\projects\calc\src>tf edit math.cs
ExplorerScc.cs
$/MathProject/dev/calc/src/math.cs:
   opened for edit in Workspace21;contoso\james
   opened for rename in WS24;contoso\sally
   newer version exists in the repository
```

Регулярно общайтесь с другими разработчиками, указывая на характер изменений в конкретном файле, чтобы избежать перекрывающихся изменений, слияние которых придется проводить вручную. Согласованность действий застрахует вас от ситуаций, когда два разработчика одновременно работают над одним фрагментом кода. С другой стороны, нет ничего страшного в том, что два или несколько человек редактируют *различные* части одного файла.

Дополнительные ресурсы

- Дополнительную информацию о просмотре ожидающих изменений в собственной рабочей области вы найдете в статье «How to: View and Manage All Pending Changes in Your Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181400(VS.80).aspx*.
- Дополнительную информацию о просмотре ожидающих изменений в чужой рабочей области вы найдете в статье «How to: View Pending Changes in Other Workspaces» по адресу *http://msdn2.microsoft.com/en-us/library/ms181401(VS.80).aspx*.

Извлечение, получение и блокировка

- Как узнать, кто изменял файл последним?
- Как работает команда get?
- В чем разница между совместным и исключающим извлечением?
- Когда следует использовать команду lock?
- Какие типы блокировок поддерживает TFS?

Как узнать, кто изменял файл последним?

Установить, кто последним изменил файл, позволяет его история. Щелкните файл правой кнопкой в окне Solution Explorer и выберите команду **View History**. Откроется окно **History** со списком последних изменений и именами пользователей, которые их внесли. Последний разработчик, изменявший данный файл, отображается в верхней части списка. Просмотреть историю файла можно также из командной строки. Для этого используется команда **history** утилиты tf.exe.

Дополнительные ресурсы

- Дополнительную информацию об истории файла вы найдете в статье «How to: View Historical Data» по адресу *http://msdn2.microsoft.com/en-us/library/ms181415(VS.80).aspx*.

Как работает команда get?

Команда **get** служит для синхронизации файлов на вашем компьютере с файлами на сервере Team Foundation Server. Получив команду **get**, Visual Studio выполняет следующие действия:

1. Сервер устанавливает, какие файлы устарели, на основе информации о том, какие версии имеются на вашем жестком диске.
2. Сервер сообщает клиенту о том, какие обновления требуются в рабочей области.

3. Клиент выполняет загрузку, перемещение и удаление файлов, а затем сообщает серверу о выполненных действиях.

Здесь предполагается, что у вас для повышения производительности установлен TFS-прокси. Операция **get** не помечает файлы для редактирования и по умолчанию не переписывает файлы, извлеченные вами для редактирования.

Дополнительные ресурсы

- Дополнительную информацию о команде get вы найдете в статье «Get Command» по адресу *http://msdn2.microsoft.com/pt-br/library/fx7sdeyf(VS.80).aspx*.

В чем разница между совместным и исключающим извлечением?

Система управления исходным кодом Team Foundation Server поддерживает как общее, так и исключающее извлечение.

Исключающее извлечение (exclusive checkout) не позволяет другому пользователю извлекать файл для правки, пока вы не вернете файл в систему управления исходным кодом. Это может привести к замедлению процесса разработки.

По умолчанию система TFS позволяет извлекать для правки один и тот же объект одновременно нескольким пользователям. Модель *совместного извлечения* (shared checkout) позволяет нескольким разработчикам трудиться в своих рабочих областях над копиями одного и того же файла. Система TFS «знает», какая версия находится в рабочей области данного разработчика, и этот разработчик должен будет разрешить конфликты до возврата объекта.

В большинстве сред коллективной разработки вероятность того, что вы в своей рабочей области внесете изменение, конфликтующее с ожидающим изменением в рабочей области другого разработчика, невелика. Большинство конфликтов в рабочих областях разрешаются системой TFS автоматически. Если конфликт не удается разрешить автоматически, воспользуйтесь командой **resolve**, которая позволит решить, чье изменение следует сохранить (ваше или другого разработчика).

Дополнительные ресурсы

- Дополнительную информацию о блокировках вы найдете в статье «How to: Lock and Unlock Folders or Files» по адресу *http://msdn2.microsoft.com/en-us/library/ms181420(VS.80).aspx*.

Когда следует использовать команду lock?

Используйте команду **lock** (блокировка), чтобы запретить другим пользователям извлекать или возвращать изменения, пока вы не снимете блокировку с файла, над которым они работают. Файл следует блокировать, только если

вы опасаетесь возникновения конфликта, требующего выполнения сложного ручного слияния. Большинство конфликтов может быть разрешено автоматически, поэтому не увлекайтесь блокировкой.

Дополнительные ресурсы

- Дополнительную информацию о типах блокировок вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181419(VS.80).aspx*.

Какие типы блокировок поддерживает TFS?

В системе Team Foundation Server предусмотрены блокировки двух типов: блокировка возврата после правки и блокировка извлечения кода для редактирования. Когда вы применяете блокировку возврата, пользователи по-прежнему могут вносить локальные изменения в объект, расположенный в других рабочих областях. Однако эти изменения нельзя будет вернуть на сервер, пока вы явно не снимете блокировку или не вернете файл после правки.

Блокировка извлечения является более жесткой, чем блокировка возврата. Применив блокировку извлечения к файлу или папке системы управления исходным кодом, вы запрещаете пользователям как извлечение файла для редактирования, так и возврат ранее сделанных ожидающих изменений. Нельзя применить блокировку извлечения к объекту, у которого в данный момент имеются ожидающие изменения.

Дополнительные ресурсы

- Дополнительную информацию о типах блокировок вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181419(VS.80).aspx*.

Распределенная и удаленная разработка

- Как работать автономно?
- Как оптимизировать среду распределенной командной разработки?
- Что такое TFS-прокси?
- Как оптимизировать производительность TFS-прокси?

Как работать автономно?

Встроенной поддержки автономной работы в TFS нет. Если вы все-таки хотите работать автономно, вам следует выполнить описанные ниже действия, причем, строго в заданной последовательности:

1. Вручную снимите флажки «только для чтения».
2. Отредактируйте файлы.
3. При необходимости добавляйте или удаляйте файлы.

4. Запустите команду **online** из инструментария Team Foundation Power Tool (TFPT).

Примечание Во время автономной работы файлы нельзя переименовывать.

---

1. Снятие флагов «только для чтения».

   По умолчанию все файлы в рабочей области, не извлеченные для правки, доступны только для чтения. При отсутствии подключения к серверу, вы должны вручную снять флажки «только для чтения» с файлов, прежде чем сможете редактировать или удалять их. Щелкните правой кнопкой файл в проводнике Windows, выберите команду **Свойства (Properties)**, сбросьте флажок **Только чтение (Read-only)** и щелкните **OK**. То же самое можно выполнить с помощью команды DOS **attrib -r**.

2. Редактирование файлов.

   Теперь вы можете редактировать любые файлы, с которых сняли метку «только для чтения».

3. Добавление или удаление файлов.

   Вы можете добавлять новые файлы, а также удалять файлы, с которых сняли метку «только для чтения». Не переименовывайте файлы, потому что инструмент TFTP online не в состоянии отличить операцию переименования от операции удаления в сочетании с операцией добавления.

Примечание Команда **Tfpt online** не ищет удаленные файлы, если явно не задать соответствующий параметр, поскольку это продолжительная операция.

---

4. Запуск команды TFPT online.

   Вернувшись в оперативный режим, введите в командной строке **TFTP online**. Эта команда проверит вашу рабочую область на предмет записываемых файлов и установит, какие изменения следует перевести в разряд ожидающих. Если вы удалили какие-либо файлы, используйте командный параметр **/deletes** для дополнительного поиска в вашей рабочей области удаленных файлов. На экране появится окно **Online**, в котором вы сможете выбрать, какие изменения следует сделать ожидающими.

Дополнительные ресурсы

- Загрузить Team Foundation Power Tools можно из источника, расположенного по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en*.
- Дополнительную информацию о Team Foundation Power Tools вы найдете в статье «Power Tools: tfpt.exe» по адресу *http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx*.

Как оптимизировать среду распределенной командной разработки?

Доступ к TFS через Интернет можно организовать одним из следующих трех способов:

- подключение по виртуальной частной сети (VPN);
- публикация TFS через обратный прокси, например, Microsoft Internet Security and Acceleration (ISA);
- расположение TFS в экстрасети («сценарий с размещением»).

Если вы обеспечиваете поддержку удаленных пользователей при помощи VPN, используйте первый способ. Он прост в реализации, обеспечивает уверенную безопасность, открывает удаленный доступ ко всем компонентам TFS и позволяет использовать TFS-прокси для повышения производительности. При таком способе доступа TFS находится в интрасети. Внешние пользователи подключаются к нему посредством VPN, внутренние — непосредственно.

Если вы не предоставили удаленным пользователям доступ VPN или доступ к домену, используйте вариант с обратным прокси. Это решение сложнее в настройке, но оно позволяет удаленным пользователям обращаться к TFS в интрасети без использования VPN. TFS находится в интрасети. Клиентские запросы из Интернета направляются к нему одним или несколькими обратными прокси, например, ISA Server.

Если вы поддерживаете работу группы удаленных пользователей, специально для которых установили TFS, используйте вариант с экстрасетью. Он позволит максимально отделить удаленных пользователей от ресурсов внутренней сети. При этом доступ к TFS имеют только внешние клиенты, а сам TFS расположен в экстрасети, за пределами действия брандмауэра.

Если удаленным доступом к Team Foundation Server через Интернет пользуется несколько клиентов, установите в удаленном офисе Team Foundation Server Proxy. Производительность возрастет за счет кеширования файлов исходного кода на прокси-сервере.

Если вы поддерживаете удаленное подключение к TFS для одного клиента, настройте его на подключение непосредственно к TFS.

Дополнительные ресурсы

- Дополнительную информацию о TFS Proxy вы найдете в статье MSDN «Team Foundation Server Proxy and Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.
- Подробнее о TFS Proxy читайте в онлайн-конференции MSDN по адресу *http://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?culture=en-US&EventID=1032291120&CountryCode=US*.

Что такое TFS-прокси?

Клиент-серверные коммуникации в TFS реализованы при помощи протокола HTTP. Чтобы повысить производительность и избежать лишних обращений к серверу, в удаленных филиалах, отделенных от БД TFS глобальной сетью, устанавливается прокси-сервер системы управления версиями TFS. Он кеширует копии файлов системы управления исходным кодом в удаленном расположении. Если файл отсутствует в локальном кеше, прокси перед возвращением файла клиенту загружает его из TFS в локальный кеш.

Хотя чаще всего прокси используется для обеспечения удаленного доступа, вы вольны использовать его и просто для разгрузки основного сервера. В частности, если сервер перегружен большим количеством одновременных локальных запросов на получение новейшего кода, вы можете переложить эту работу на прокси. Поскольку каждый уровень приложений включает в себя прокси-сервер, нет никакого смысла ставить прокси перед этим уровнем.

Дополнительные ресурсы

- Дополнительную информацию о TFS Proxy вы найдете в статье MSDN «Team Foundation Server Proxy and Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms252490(VS.80).aspx*.

Как оптимизировать производительность TFS-прокси?

Существуют следующие способы оптимизации производительности прокси-сервера:

- Убедитесь, что включена функция кеширования, и периодически проверяйте счетчики производительности кеша и журналы регистрации событий (ошибки и предупреждения) на прокси-сервере, чтобы иметь представление о его производительности. Помните, что прокси-сервер TFS сохраняет статистику производительности кеша в XML-файле ProxyStatistics.xml. Интервал, с которым происходит сохранение статистики, поддается настройке. Файл ProxyStatistics.xml расположен в подпапке App_Data папки установки прокси
- Периодически запускайте запланированное задание для копирования новейших файлов на прокси-сервер. Это обеспечит наличие актуальных версий в кеше прокси. Как следствие, увеличится количество удачных обращений к кешу.
- Если вам известно, что предстоит загрузка больших файлов по сети с малой пропускной способностью (< 3 Мб/с), измените значение параметра **executionTimeout** в файле Web.config. Значение по умолчанию равно одному часу <httpRuntime executionTimeout="3600"/>.

Дополнительные ресурсы

- Дополнительную информацию о проверке производительности прокси-сервера вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms252455(VS.80).aspx*.

Переход с других версий

- Чем система управления версиями TFS отличается от VSS?
- Чем модель извлечения TFS отличается от аналогичной модели VSS?
- Как осуществить перенос исходного кода из VSS в TFS?
- Как осуществить перенос исходного кода из других систем управления версиями?

Чем система управления версиями TFS отличается от VSS?

Система Microsoft Visual SourceSafe предназначена для поддержки индивидуальных разработчиков и небольших групп. Система управления версиями TFS создана для профессиональных разработчиков и крупных команд, в которых количество пользователей на сервер достигает 2500. Среди крупных усовершенствований системы управления версиями TFS перечислим глубокую интеграцию с Visual Studio 2005, а также поддержку распределенной разработки, которая позволяет эффективно осуществлять доступ к файлам через глобальные сети. Основные различия между двумя системами таковы:

- **Изменяемость кода** В системе Visual SourceSafe отсутствует возможность оценки изменяемости кода. Система управления версиями TFS предоставит вам соответствующие показатели для каждого файла, возвращаемого в хранилище.
- **Извлечение** Система Visual SourceSafe получает из БД новейший или помеченный файл и перезаписывает локальную версию. Система управления версиями TFS открывает запись в версию, находящуюся в вашей рабочей области, но не перезаписывает ее версией из БД.
- **Блокировка файлов** Система Visual SourceSafe автоматически применяет к файлу, извлеченному для правки, исключающую блокировку. Система управления версиями TFS позволяет нескольким разработчикам параллельно выполнять редактирование одного файла. При возникновении конфликтов, они, как правило, разрешаются автоматически.
- **Ветвление** В системе Visual SourceSafe используется метод ветвления «pin-and-share». В системе управления версиями TFS применяется метод ветвления в пространстве путей. Благодаря этому облегчается сопровождение старых версий ПО, поскольку данный метод облегчает перенос изменений из одной ветви в другую или внесение изменений в обе ветви сразу.

- **Совместное использование** Система управления версиями TFS не поддерживает совместное использование. Перенос общего файла приведет к его копированию в целевую папку.
- **Интеграция операций** Система Visual SourceSafe — изолированная система управления версиями. Система управления версиями TFS интегрирована с системой отслеживания рабочих элементов и системой подготовки отчетов TFS.

Дополнительные ресурсы

- Дополнительную информацию о различиях систем Visual SourceSafe и TFS вы найдете в статье «Visual Source Safe vs. Team Foundation Server» по адресу *http://msmvps.com/blogs/vstsblog/archive/2005/07/02/56401.aspx*.

Чем модель извлечения TFS отличается от аналогичной модели VSS?

При извлечении файла для правки из системы управления версиями TFS файл в вашей рабочей области открывается для записи, но его синхронизация с сервером БД не выполняется. Чтобы получить с сервера БД последнюю версию файла, следует запустить команду **get**. При извлечении файла в VSS файл копируется с сервера БД в вашу рабочую область и открывается для записи.

Система Visual SourceSafe по умолчанию использует исключительную блокировку, в то время как в TFS по умолчанию применяется совместное извлечение. Оно позволяет нескольким разработчикам одновременно вносить изменения в один и тот же файл. В отличие от Visual SourceSafe, в TFS большинство конфликтов разрешается автоматически.

Дополнительные ресурсы

- Дополнительную информацию о различиях моделей извлечения систем VSS и TFS вы найдете в статье «Team Foundation vs. SourceSafe | Checking Out» по адресу *http://blogs.msdn.com/korbyp/archive/2004/07/28/199720.aspx*.

Как осуществить перенос исходного кода из VSS в TFS?

В комплекте Team Foundation Server поставляется инструмент VSS Converter, позволяющий переносить файлы, папки, историю версий, метки и пользовательскую информацию из БД Visual SourceSafe в систему управления исходным кодом Team Foundation Server.

Следует знать, что конвертер имеет некоторые ограничения, например:

- Не сохраняется история общего доступа к файлам, поскольку система Team Foundation Server не поддерживает общий доступ. Перемещение совместно используемого файла сводится к копированию этого файла в конечную папку.

- Не сохраняется история ветвления.
- Система Team Foundation Server не поддерживает закрепление (pinning). Закрепленные файлы переносятся с созданием двух меток.
- Во время переноса не сохраняются временные метки, связанные с действиями.

Дополнительные ресурсы

- Дополнительную информацию о подготовке к переходу вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181246(en-us,vs.80).aspx*.
- Дополнительную информацию о том, как выполнять переход, вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181247(VS.80).aspx*.
- Дополнительные сведения об ограничениях VSS Converter вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms252491(VS.80).aspx*.
- Дополнительную информацию об инструменте VSS Analize вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ysxsfw4x.aspx*.
- Дополнительные сведения о команде **migrate** утилиты командной строки **VSS Converter** вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms400685(VS.80).aspx*.

Как осуществить перенос исходного кода из других систем управления версиями?

Вы можете вручную выполнить экспорт файлов из предыдущей системы управления версиями, а затем импортировать их в систему управления версиями TFS. Если вам нужно сохранить историю файлов или другие атрибуты предыдущей системы, воспользуйтесь объектной моделью TFS, чтобы написать собственную утилиту перехода.

В данный момент корпорация Microsoft ведет работу по созданию конвертера ClearCase. О выпуске конвертера будет объявлено дополнительно в блоге TFS Migration, расположенном по адресу *http://blogs.msdn.com/tfs_migration*.

Существует конвертер, созданный компанией Component Software, который совместим с системами GNU RCS, CS-RCS, GNU CVS, Subversion (SVN) и Visual SourceSafe (VSS).

Дополнительные ресурсы

- Дополнительную информацию о расширении системы управления версиями TFS вы найдете в статье «Walkthru: The Version Control Object Model» по адресу *http://msdn2.microsoft.com/en-us/library/bb187335 (VS.80).aspx*.
- Дополнительную информацию о конвертере компании Component Software вы найдете на сайте компании по адресу *http://www.component-software.com/Products/Converter/*.

Управление проектом и рабочей областью

- Как упорядочить командные проекты?
- Как управлять зависимостями между проектами?
- Что такое рабочая область?
- Как изолировать работу разработчика при помощи рабочей области?
- Как эффективно организовать сопоставление рабочей области?
- Когда следует создавать новый проект, а когда — новую ветвь?
- Как управлять исходным кодом, используемым в нескольких проектах?
- Как управлять двоичными файлами, используемыми в нескольких проектах?
- Как организовать дерево исходного кода?

Как упорядочить командные проекты?

Командные проекты призваны представить самую крупную единицу работы, существующую в вашей организации. Чаще всего, это программный продукт, находящийся в разработке.

Выберите одну из следующих стратегий управления командными проектами:

- один проект на приложение;
- один проект на каждый выпуск.

Создание одного проекта на каждое приложение

Если вы намерены переносить из версии в версию не только исходный код, но и рабочие элементы, а также другие объекты TFS, используйте один командный проект на все приложение. Все папки TFS при этом будут автоматически переноситься на следующий выпуск. Когда все будет готово к выпуску новой версии, вы сможете создать для ее изоляции ветвь внутри проекта.

Причины, по которым не следует использовать один проект на каждое приложение:

- В параллельных выпусках будут использоваться одни и те же схемы рабочих элементов, политики возврата после правки и руководства по процессам.
- Будет затруднено составление отчетов. Поскольку отчеты составляются для всего проекта, вам придется использовать фильтрацию по выпуску.
- Если количество разрабатываемых приложений исчисляется сотнями, наличие собственного проекта для каждого из них нанесет ущерб производительности TFS. К тому же, вы рискуете выйти за пределы масштабируемости.
- После нескольких выпусков в проекте набирается огромный «багаж».

Самый простой способ избавиться от него — создать новый проект с переносом в него только нужного кода.

Создание одного проекта на каждый выпуск

Если вы готовы с каждым выпуском начинать все заново, не перенося рабочие элементы и другие папки TFS, создавайте отдельный проект на каждый выпуск. Это позволит вам изменять схемы рабочих элементов, последовательность операций, политики возврата после правки и прочие элементы, не затрагивая при этом предыдущий выпуск. Это особенно полезно, если он будет сопровождаться отдельной командой, например, группой поддержки, порядок работы которой может отличаться от основной команды разработчиков.

Причины, по которым не следует использовать один проект на каждый выпуск:

- Переместить код из одного проекта в другой достаточно просто, а вот перемещать рабочие элементы и прочие папки TFS из одного проекта в другой нелегко. Рабочие элементы можно копировать в другой проект только по одному. Если вы хотите копировать наборы рабочих элементов, вам придется написать собственную утилиту.
- Если количество приложений и выпусков исчисляется сотнями и каждый из них находится в собственном проекте, вы столкнетесь с проблемой производительности системы TFS и выйдете за пределы масштабируемости.

  Выбирая стратегию, имейте в виду следующее:
- Организовать общий доступ к исходному коду из нескольких командных проектов можно несколькими способами:
  - ветвлением исходного кода из одного проекта в другой;
  - сопоставлением исходного кода из другого проекта в вашей рабочей области.
- Система Team Foundation Server способна вместить около 500 проектов на основе шаблона процесса MSF Agile или до 250 проектов на основе шаблона MSF CMMI. Если вы создаете собственный процесс или настраиваете существующий, помните, на масштабируемость сервера наибольшее влияние оказывает схема рабочих элементов. Чем сложнее схема, там меньше проектов сможет поддерживать сервер.

Дополнительные ресурсы

- Дополнительную информацию об использовании командных проектов вы найдете в статье «When to use Team Projects» по адресу *http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx*.

Как управлять зависимостями между проектами?

Если вам нужно установить ссылку на внешний файл сборки, который был создан вне вашего решения Visual Studio, например, на библиотеку DLL стороннего производителя, у вас есть три варианта действий:

- Вы можете хранить внешний файл сборки как двоичный ресурс в составе вашего проекта. Этот вариант подходит для случаев, когда вы хотите сохранить зависимость для других проектов, которым может понадобиться ссылка на нее, или когда ваш проект — единственный, которому нужна эта зависимость.
- Вы можете сослаться на сборку, находящуюся на сервере сборок, при помощи буквы виртуального диска или UNC-пути.
- Вы можете хранить набор внешних файлов сборок в папке общего командного проекта. Этот способ удобен при наличии нескольких проектов, ссылающихся на общую зависимость, когда ваш проект не является ее очевидным собственником.
  - Если в случае изменения двоичного файла вы желаете получать обновление немедленно, создайте сопоставление рабочей области этого командного проекта с вашим локальным компьютером. Внутри локальной структуры создайте файловую ссылку на сборку.
  - Если вы хотите управлять расписанием сборки данной зависимости, создайте ветвь из общего проекта в ваш проект, чтобы переносить изменения к себе в любое удобное для вас время.

В целом, следует избегать зависимостей, переходящих из проекта в проект. Старайтесь объединять все связанные и зависимые решения и проекты в одном командном проекте. Это сократит трудозатраты на разработку сценария сборки. Если у вас есть зависимость, для ее определения используйте ссылки на проект. Избегайте файловых ссылок, ибо ими намного труднее управлять.

> **Примечание** Ссылки между различными проектами поддерживаются внутри решения. Имейте в виду, что файл проекта (csproj, vjsproj, vcproj, vbproj и т. д.) может включаться в несколько решений (sln-файлов). Иными словами, один проект может использоваться несколькими решениями.

Дополнительные ресурсы

- Дополнительную информацию о ссылках на проект вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/*

library/ms181425(VS.80).aspx.

- Дополнительную информацию о рабочих областях вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181383(VS.80).aspx*.

Что такое рабочая область?

Рабочей областью (workspace) называется клиентская копия файлов и папок с сервера управления версиями TFS. Локальные ожидающие изменения изолируются внутри рабочей области, пока вы не вернете их на сервер в виде единого блока (набора изменений). Одна рабочая область может содержать ссылки на несколько командных проектов. Для изоляции файлов или версий вы вольны использовать несколько рабочих областей.

> **Примечание** Рабочая область может создаваться как для компьютера, так и для учетной записи пользователя. Поэтому, на компьютере могут присутствовать различные определения рабочих областей. В частности, можно завести несколько рабочих областей на одном компьютере, создав их для различных учетных записей на этом компьютере.

Дополнительные ресурсы

- Дополнительную информацию о рабочих областях вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181383(VS.80).aspx*.

Как изолировать работу разработчика при помощи рабочей области?

Вы, как разработчик, можете создать две рабочих области: в одной разместить файлы и папки, над которыми работает вся команда, а в другой — файлы и папки, которые вы хотите изолировать. Изоляция файлов может понадобиться, чтобы какое-то время работать над ними параллельно с основной разработкой. В частности, этот способ может использоваться для работы над рискованными ожидающими изменениями или при рецензировании кода.

Дополнительные ресурсы

- Дополнительную информацию о рабочих областях вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181383(VS.80).aspx*.

Как эффективно организовать сопоставление рабочей области?

Сопоставление рабочей области — это средство отображения находящихся на сервере файлов и папок на ваш локальный диск.

Создавая сопоставления рабочей области, используйте следующие рекомендации:

- **Сопоставляйте рабочие области на корневом уровне командного проекта** В новых командных проектах сопоставляйте корень проекта ($/MyTeamProject) с папкой на локальном диске, имеющей то же имя, например, C:\TeamProjects. Вся структура локальной папки создается автоматически и будет в точности повторять структуру в системе управления исходным кодом.
- **Используйте уникальный путь к локальной папке на совместно используемых компьютерах** Два пользователя одного и того же компьютера не могут использовать одно сопоставление рабочей области. Допустим, вы и ваш коллега не можете сопоставить один командный проект ($/MyTeamProject) с одной и той же папкой на локальном компьютере. Создавайте сопоставления в папке Мои документы (хотя это удлиняет путь) или разработайте соглашение об именах для папок на локальном компьютере (например, C:\TeamProjects\User1, C:\TeampProjects\User2 и т. д.).
- **Подумайте, нужно ли вам все дерево** Чтобы увеличить производительность и сократить занимаемый объем диска, сопоставляйте только те файлы, которые требуются для проекта разработки. Как правило, вам требуются файлы и проекты, связанные с решением, над которым вы работаете.
- **Не используйте сопоставление рабочей области для поддержки зависимостей между различными проектами** Как правило, следует избегать зависимостей, пересекающих командные проекты. Старайтесь объединить все связанные и зависимые решения и проекты в одном командном проекте. Это позволит реже прибегать к настройке сборочного сценария. Если у вас есть зависимость, для ее определения используйте ссылки на проект или перенесите зависимость из общего проекта в свой проект. Следует избегать файловых ссылок, потому что ими сложнее управлять. Исключение составляют случаи, когда параллельно ведется разработка зависимого проекта, и вам нужно получать изменения в реальном времени. В этом случае вы можете воспользоваться сопоставлением рабочей области. Если зависимый код приводит к появлению большого количества серьезных ошибок, используйте ветвление.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию об изменении рабочей области вы найдете в статье «How to: Edit a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms245466(VS.80).aspx*.

Когда следует создавать новый проект, а когда — новую ветвь?

Создавайте ветвь, если хотите изолировать код проекта, над которым работают разработчики, сохранив при этом общий доступ к другим ресурсам TFS, например, рабочим элементам, руководствам процесса и отчетам.

Создавайте новый командный проект, если намерены использовать новые ресурсы TFS (рабочие элементы, руководство процесса и отчеты). Также следует создавать новый командный проект, если вы собираетесь перенести код, не перенося другие ресурсы TFS.

Дополнительные ресурсы

- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о добавлении нового проекта вы найдете в статье «How to: Add a Project or Solution to Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181373(VS.80).aspx*.

Как управлять исходным кодом, используемым в нескольких проектах?

Управление общим исходным кодом включает в себя принятие двух решений:

1. Где хранить исходный код?
2. Как ваша команда будет получать доступ к нему?

Возможны следующие варианты хранения исходного кода:

- Если исходный код явно принадлежит той или иной команде, храните его в проекте этой команды.
- Если у совместно используемого исходного кода нет конкретного владельца, создайте командный проект специально для его хранения.

Если вы хотите использовать исходный код в другом проекте, выберите один из следующих вариантов:

- Если вам нужно поддерживать постоянную синхронизацию с совместно используемым кодом, создайте сопоставление между местом его хранения и локальной рабочей областью на клиентских компьютерах.
- Если синхронизация требуется вам лишь периодически, выполните ветвление из общего расположения в целевой проект. Каждый раз при слиянии из общего расположения в целевой проект вы будете копировать новую версию исходного кода.

Используя как сопоставление рабочей области, так и ветвление, придерживайтесь соглашений об именах, чтобы четко представить место совместно используемого кода в вашем проекте, например:

- **Main** — контейнер всех ресурсов, необходимых для выпуска ПО.
  - **Source** — контейнер для всего, что нужно для сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, заимствованного из других проектов.
    - **Unit Tests** — контейнер для модульных тестов.
    - **Lib** — контейнер для двоичных зависимостей.
  - **Docs** — контейнер для документации, поставляющейся с продуктом.
  - **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
  - **Builds** — контейнер для сценариев сборки.
  - **Tests** — контейнер, содержащий результаты тестирования.

Дополнительные ресурсы

- Дополнительную информацию о ссылках на проект вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о рабочих областях вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181383(VS.80).aspx*.

Как управлять двоичными файлами, используемыми в нескольких проектах?

Управление общими двоичными файлами сходно с управлением общим исходным кодом. Вы должны решить, где хранить двоичные файлы и каким образом ваша команда будет осуществлять доступ к ним.

Возможны следующие варианты хранения двоичных файлов:

- Если общие двоичные файлы явно принадлежат той или иной команде, храните их в проекте этой команды.
- Если у общих двоичных файлов нет конкретного владельца, создайте командный проект специально для них.

Варианты использования общих двоичных файлов в другом проекте:

- Как правило, совместно используемые двоичные файлы обновляются только периодически. Если в вашем случае это справедливо, выполните ветвление из общего расположения в расположение целевого проекта. При изменении двоичных файлов вы запустите слияние и получите последнюю версию.

- Чтобы поддерживать постоянную синхронизацию с общими двоичными файлами, сопоставьте исходный код из общего расположения с локальной рабочей областью на клиентских компьютерах.

Используя как сопоставление рабочей области, так и ветвление, придерживайтесь соглашений об именах, чтобы четко представить место совместно используемых двоичных файлов в вашем проекте, например:

- **Main** — контейнер всех ресурсов, необходимых для выпуска ПО.
  - **Source** — контейнер для всего, что нужно для сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, заимствованного из других проектов.
    - **Unit Tests** — контейнер для модульных тестов.
    - **Lib** — контейнер для двоичных зависимостей.
  - **Docs** — контейнер для документации, поставляющейся с продуктом.
  - **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
  - **Builds** — контейнер для сценариев сборки.
  - **Tests** — контейнер, содержащий результаты тестирования.

Дополнительные ресурсы

- Дополнительную информацию о ссылках на проект вы найдете в статье «Project References» по адресу *http://msdn2.microsoft.com/en-us/library/ez524kew(VS.80).aspx*.
- Дополнительную информацию о ветвлении вы найдете в статье «How to: Branch Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181425(VS.80).aspx*.
- Дополнительную информацию о рабочих областях вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181383(VS.80).aspx*.

Как организовать дерево исходного кода?

Структура дерева исходного кода представляет собой сочетание структуры папок, структуры файлов и структуры ветвей. Внутри главной ветви свою работоспособность в различных по размеру командах доказала следующая структура папок и файлов:

- **Main** — контейнер всех ресурсов, необходимых для выпуска ПО.
  - **Source** — контейнер для всего, что нужно для сборки.
    - **Code** — контейнер для исходного кода.
    - **Shared Code** — контейнер для исходного кода, заимствованного из других проектов.
    - **Unit Tests** — контейнер для модульных тестов.

        - **Lib** — контейнер для двоичных зависимостей.
    - **Docs** — контейнер для документации, поставляющейся с продуктом.
    - **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
    - **Builds** — контейнер для сценариев сборки.
    - **Tests** — контейнер, содержащий результаты тестирования.

Все ветви, создаваемые из Main, повлекут за собой копирование структуры папок и файлов Main в новую ветвь, например:

- **Development** — ветвь разработки.
    - **Source** — контейнер для всего, что нужно для сборки.
        - **Code** — контейнер для исходного кода.
        - **Shared Code** — контейнер для исходного кода, заимствованного из других проектов.
        - **Unit Tests** — контейнер для модульных тестов.
        - **Lib** — контейнер для двоичных зависимостей.
- **Main** — ветвь интеграции.
    - **Source** — контейнер для всего, что нужно для сборки.
        - **Code** — контейнер для исходного кода.
        - **Shared Code** — контейнер для исходного кода, заимствованного из других проектов.
        - **Unit Tests** — контейнер для модульных тестов.
        - **Lib** — контейнер для двоичных зависимостей.
    - **Docs** — контейнер для документации, поставляющейся с продуктом.
    - **Installer** — контейнер для исходного кода и двоичных файлов программы установки.
    - **Builds** — контейнер для сценариев сборки.
    - **Tests** — контейнер, содержащий результаты тестирования.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочей области вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.

Отложенные правки

- Что такое отложенные правки?
- Что такое набор отложенных правок?
- Как используются отложенные правки?

- Как использовать отложенные правки для архивации?
- Для чего нужно извлекать набор отложенных правок?

Что такое отложенные правки?

Отложенные правки (shelving) или наборы отложенных правок (shelveset) позволяют сохранить ожидающие изменения в системе управления исходным кодом, не выполняя возврат измененных файлов. Таким образом, вы получаете все преимущества от хранения (архивации) файлов на сервере и вместе с тем знаете, что незавершенность этих файлов никак не отразится на регулярных сборках.

Дополнительные ресурсы

- Дополнительную информацию о сохранении отложенных правок на сервере вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Что такое набор отложенных правок?

Набором отложенных правок (shelveset) называется набор файлов, которые сохранены на сервере, но еще не готовы для возврата в систему управления исходным кодом. Файлы откладывают на сервер, чтобы сохранить ожидающие изменения из рабочей области или предоставить общий доступ к ним. Вы также можете использовать файлы отложенных правок для передачи неоконченной работы другому разработчику.

Дополнительные ресурсы

- Дополнительную информацию о сохранении отложенных правок на сервере вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Как используются отложенные правки?

Существует несколько общих сценариев использования отложенных правок:

- Вы находитесь в процессе подготовки изменений исходного кода, но получили новое, более срочное задание (например, срочное исправление ошибки). Вам придется вновь вернуться к стабильной версии кода, но вы не хотите терять изменения. Отложите правки кода, с тем чтобы позднее без труда извлечь их.
- Вы не завершили работу к концу дня и хотите сохранить ее на сервере. Сохраняя отложенные правки, вы переносите их на Team Foundation Server, откуда вы или кто-то другой сможете извлечь их на следующий день.
- Вы хотите обсудить незавершенный код с удаленным членом команды. Вместо отправки кода по электронной почте отложите правки на сервер, а затем предложить коллеге извлечь эти файлы.

- Вы хотите передать частично сделанную работу другому члену команды для ее завершения.

Дополнительные ресурсы

- Дополнительную информацию о сохранении отложенных правок на сервере вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Как использовать отложенные правки для архивации?

Если ваша работа над одним или несколькими файлами исходного кода к концу рабочего дня не завершена, сохраните правки кода на сервере. Это обеспечит архивацию исходного кода на сервере без выполнения возврата неоконченной работы.

Дополнительные ресурсы

- Дополнительную информацию о сохранении отложенных правок на сервере вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Для чего нужно извлекать набор отложенных правок?

Типичные причины извлечения набора отложенных правок таковы:

- для продолжения работы над набором файлов, сохраненным ранее в виде набора отложенных правок;
- для интеграции сохраненных отложенных правок в текущую работу;
- для рецензирования чьего-либо кода;
- для принятия на себя частично завершенной работы другого разработчика.

Дополнительные ресурсы

- Дополнительную информацию о сохранении отложенных правок на сервере вы найдете в статье «How to: Shelve and Unshelve Pending Changes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181404(VS.80).aspx*.

Дополнительные ресурсы по системе управления исходным кодом

- Дополнительные сведения о системе управления исходным кодом TFS вы найдете в статье «Team Foundation Source Control» по адресу *http://msdn2.microsoft.com/en-us/library/ms181237(VS.80).aspx*.

Практикум

В этой части

- Как добавить нового разработчика в проект Visual Studio 2005.
- Как автоматически выполнять анализ кода припомощи Team Build.
- Как создать собственный отчет в Visual Studio 2005 Team Foundation Server.
- Как создать отчет о развитии рисков в Visual Studio 2005 Team Foundation Server.
- Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server.
- Как создать дерево кода в Team Foundation Server.
- Как настроить шаблон процесса в Visual Studio Team Foundation Server.
- Как настроить отчет в Visual Studio Team Foundation Server.
- Как управлять проектами в Visual Studio Team Foundation Server.
- Как перенести исходный код из Visual Source Safe в Team Foundation Server.
- Как настроить непрерывную интеграцию в Visual Studio Team Foundation Server.
- Как настроить плановую сборку в Visual Studio Team Foundation Server.
- Как структурировать приложения ASP.NET в Visual Studio Team Foundation Server.
- Как структурировать приложения Windows в Visual Studio Team Foundation Server.
- Как структурировать папки управления исходным кодом в Visual Studio Team Foundation Server.

Как добавить нового разработчика в проект Visual Studio 2005 Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System.
- Microsoft SQL Server™ Reporting Services.

Описание

В этой статье подробно разбирается процесс введения нового разработчика в командный проект Team Foundation Server.

Содержание

Задачи

- Предоставить разработчику доступ к командному проекту.

- Предоставить разработчику разрешения Read и Contributor на сайте SharePoint.
- Предоставить разработчику возможность просматривать и подписываться на отчеты.

Обзор

Вводя в проект нового разработчика, необходимо предоставить ему соответствующие права доступа к проекту и сайту SharePoint проекта. Учетная запись нового разработчика должна также иметь права доступа к SQL Server Reporting Services, чтобы разработчик мог просматривать отчеты, представленные на портале проекта.

Порядок операций

- Шаг 1 — предоставление доступа к командному проекту.
- Шаг 2 — предоставление доступа к сайту проекта.
- Шаг 3 — предоставление доступа к службам SQL Server Reporting Services.

Шаг 1 — предоставление доступа к командному проекту

На этом шаге новому члену команды предоставляется доступ к Team Foundation Server. Чтобы открыть ему доступ к командному проекту, выполните следующие действия:

1. Зарегистрируйтесь в Visual Studio под учетной записью, входящей в группу Team Foundation Administrators.
2. Добавьте необходимый проект в **Team Explorer** (если его там еще нет).
3. В **Team Explorer** щелкните правой кнопкой мыши командный проект, выберите **Team Project Settings** и щелкните **Group Membership**.
4. Выберите **Project\Contributors**, щелкните **Properties** и добавьте учетную запись нового разработчика в эту группу.

Обратите внимание, что членство в группе Contributors обеспечивает типовой набор разрешений, необходимых разработчику, включая возможность добавлять, изменять, удалять элементы командного проекта и выполнять сборки.

Шаг 2 — предоставление доступа к сайту проекта

На этом шаге новый член группы получает доступ к сайту проекта SharePoint. Выполните следующие действия:

1. Войдите на сайт проекта под учетной записью, входящей в группу SharePoint Administrator.

   Помните, что сайт проекта YourProject по умолчанию располагается по адресу *http://server/sites/YourProject/default.aspx*.
2. Щелкните **Site Settings**.
3. Щелкните **Manage Users** под надписью **Administration**.
4. Щелкните **Add Users**.
5. Введите регистрационное имя учетной записи нового разработчика в формате **домен\учетнаязаписьпользователя**, выберите **Contributor** и щелкните **Next**.
6. Введите адрес электронной почты разработчика в поле адреса и, если хотите, приветственное сообщение для разработчика на сайте.
7. Щелкните **Finish**.

Членство в группе Contributors обеспечивает разработчику возможность просматривать и добавлять содержимое в существующие библиотеки документов и списки. Членство в группе Reader обеспечивает доступ к сайту только для чтения. Иногда этого вполне достаточно, все зависит от потребностей нового разработчика.

Шаг 3 — предоставление доступа к службам SQL Server Report Services

На этом шаге новому члену команды предоставляется доступ к SQL Report Services. Выполните следующие действия:

1. Зарегистрируйтесь на административном веб-сайте SQL Server Reporting Services под учетной записью администратора. Сайт находится по адресу *http://server/reports*.
2. Щелкните имя своего командного проекта.
3. Перейдите на вкладку **Properties**.
4. Перейдите на вкладку **Security**.
5. Щелкните **New Role Assignment**.
6. Введите Windows-имя разработчика, выберите **Browser** и щелкните **OK**.

Членство в группе Browser позволяет разработчику просматривать и подписываться на отчеты.

Дополнительные ресурсы

- Назначение прав доступа в SQL Server Reporting Services подробно рассматривается в статье «Setting Permissions in Reporting Services» по адресу *http://msdn2.microsoft.com/en-us/library/aa337491.aspx*.

- Подробную информацию о ролях системы безопасности уровня приложений вы найдете в статье «Securing Reporting Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms157198.aspx*.
- Справочник администратора SharePoint приводится в статье «Managing Site and Group Permissions» по адресу *http://www.microsoft.com/resources/documentation/wss/2/all/adminguide/en-us/stsf03.mspx?mfr=true*.
- Дополнительную информацию об управлении правами доступа в TFS вы найдете в статье «Managing Permissions» по адресу *http://msdn2.microsoft.com/en-us/library/ms253094(VS.80).aspx*.

Как автоматически выполнять анализ кода при помощи Team Build

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System.

Описание

В этой статье подробно разбирается процесс настройки сценария сборки и включения в него этапа анализа кода. Это обеспечит автоматическое выполнение анализа кода в составе сценария сборки проекта и включение отчета о результатах анализа в результаты сборки.

Содержание

Задача

- Выполнить анализ кода в составе сборки с целью контроля ее качества.

Обзор

Visual Studio Team System Team Build позволяет определять для проекта типы сборки, благодаря которым сервер способен компилировать приложение и предоставлять его для доступа на общем сетевом ресурсе. Если в сценарий сборки включен анализ кода, он будет автоматически выполняться при каждой сборке, а отчет о его результатах будет выкладываться на странице отчетов о результатах сборки. В этой статье подробно разбирается процесс настройки сценария сборки и включения в него этапа анализа кода.

Порядок операций

- Шаг 1 — тестирование сборки.
- Шаг 2 — включение анализа кода в сборку.
- Шаг 3 — тестирование анализа кода.

Прежде всего

Прежде чем включить анализ кода в сценарий сборки, необходимо убедиться в следующем:

- Ваш пользовательский идентификатор Team Foundation обладает разрешением на администрирование сборки. Уточните у администратора, обладаете ли вы таким разрешением.
- Сценарий сборки для вашего проекта уже существует. Проверить это можно, посмотрев на содержимое узла Team Build в Visual Studio Team Explorer.

Шаг 1 — тестирование сборки

Перед включением анализа кода протестируйте сценарий сборки, чтобы убедиться в отсутствии проблем. Это делается следующим образом:

1. В Visual Studio откройте **Team Explorer**.
2. Разверните узел своего командного проекта.
3. Разверните узел **Team Builds**.
4. Щелкните правой кнопкой мыши существующий сценарий сборки и выберите **Build Team Project**.
5. Убедитесь в успешном выполнении сборки. Если сборка выполнена со сбоем или вовсе не завершена, исправьте ошибки, прежде чем переходить к следующему этапу.

Шаг 2 — включение анализа кода в сборку

Убедившись, что сборка выполняется правильно, можно включить в нее анализ кода. Выполните следующие действия:

1. Откройте **Source Control Explorer**.
2. Разверните папку своего командного проекта.
3. Разверните папку **TeamBuildTypes**.
4. Выберите папку типа сборки, в который хотите включить анализ кода.
5. Извлеките версию файла TFSBuild.proj из системы управления исходным кодом для редактирования. Возможно, сначала вам понадобиться выполнить операцию **Get Latest Version**.
6. Откройте файл TFSBuild.Proj, дважды щелкнув его в **Source Control Explorer**.
7. Если требуется выполнять анализ кода для всех проектов независимо от их настроек, присвойте тегу <RunCodeAnalysis> значение Always.
8. Если вы хотите выполнять анализ кода в зависимости от настроек проекта, присвойте тегу <RunCodeAnalysis> значение Default.
9. При использовании индивидуальных настроек для каждого проекта анализ кода для проекта включается следующим образом:
   а. Откройте решение в Visual Studio.
   б. В Solution Explorer щелкните проект правой кнопкой мыши.
   в. Выберите команду **Properties.**
   г. Щелкните **Code Analysis.**
   д. Установите флажок **Enable Code Analysis**.
   е. Извлеките файл проекта .csproj из системы управления исходным кодом для редактирования.
   ж. Сохраните файл, щелкнув значок **Save** на панели инструментов при открытом окне свойств.
   з. Верните файл .csproj в систему управления исходным кодом.
10. Сохраните TFSBuild.proj и верните его в систему управления исходным кодом.

Шаг 3 — тестирование анализа кода

Включив анализ кода в сценарий сборки, протестируйте его, чтобы убедиться, что все работает нормально. Выполните следующие действия:

1. В Team Explorer щелкните правой кнопкой мыши тип сборки и выберите команду **Build Team Project**.
2. По завершении сборки щелкните ссылку на ее журнал.

3. Просмотрите предупреждения анализа кода, приведенные в конце журнала сборки. Их идентификаторы начинаются с «CA», как в следующих примерах:
   - MSBUILD : warning : CA2209 : Microsoft.Usage : No valid permission requests were found for assembly 'HelloWorldTest'. You should always specify the minimum security permissions using SecurityAction. RequestMinimum.
   - MSBUILD : warning : CA2210 : Microsoft.Design : Sign 'HelloWorldTest' with a strong name key.
   - MSBUILD : warning : CA1014 : Microsoft.Design : 'HelloWorldTest' should be marked with CLSCompliantAttribute and its value should be true.

Дополнительные ресурсы

- Дополнительную информацию об инструментах анализа кода вы найдете в статье «Guidelines for Using Code Analysis Tools» по адресу *http://msdn2.microsoft.com/en-us/library/ms182023(VS.80).aspx*.
- Типы сборки подробно рассматриваются в статье «Overview of Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms181710(VS.80).aspx*.

Как создать собственный отчет в Visual Studio 2005 Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System.
- Microsoft SQL Server™ Reporting Services.

Описание

В статье подробно разбирается процесс создания нового отчета с последующей публикацией на портале отчетов команды.

Содержание

Задачи

- Научиться создавать проект отчетов в Visual Studio.
- Научиться создавать новый отчет в проекте отчетов.
- Научиться публиковать новый отчет на сервере отчетов.

Обзор

Отчеты, поставляемые с VSTS, основаны на использовании SQL Server Reporting Services. С помощью конструктора отчетов Visual Studio (Business Intelligence Development Studio) из комплекта клиентских инструментов SQL Server 2005 можно редактировать готовые отчеты или создавать собственные. Создание собственного отчета в Visual Studio начинается с создания проекта отчетов. Затем создаются источники данных для подключения к реляционной базе данных TFS и базе данных Online Analytical Processing (OLAP).

Порядок операций

- Шаг 1 — создание нового проекта отчетов.
- Шаг 2 — создание источников данных.
- Шаг 3 — создание нового отчета в проекте.
- Шаг 4 — изменение отчета.
- Шаг 5 — развертывание отчета на Team Foundation Server.
- Шаг 6 — тестирование отчета.

Прежде всего

Прежде чем приступать к настройке отчета для Team Foundation Server, убедитесь в следующем:

- На компьютере, который будет использоваться для настройки отчета, должна быть установлена среда Business Intelligence Development Studio. Чтобы проверить ее наличие, при создании нового проекта посмотрите, имеется ли в Visual Studio тип Business Intelligence Project.
- Ваша учетная запись должна быть членом роли безопасности Microsoft Analysis Server TfsWarehouseDataReaders на сервере уровня данных.
- Ваша учетная запись должна обладать правами администратора БД TFSWarehouse уровня данных.
- Ваша учетная запись должна быть членом роли Publisher в SQL Server Reporting Services на сервере уровня приложений.

Шаг 1 — создание нового проекта отчетов

Чтобы добавить в проект новый отчет и настроить его, начните с создания проекта отчетов. Выполните следующие действия:

1. В Visual Studio откройте меню **File**, выберите команду **New** и щелкните **Project**.
2. Выберите тип **Business Intelligence Project**.
3. Выберите шаблон **Report Server Project**.
4. Задайте имя и расположение проекта. Затем щелкните **OK**.

Шаг 2 — создание источников данных

Чтобы редактировать и публиковать настроенный отчет, необходимо добавить источники данных для хранилища данных Team Foundation Server и OLAP-куб. После добавления этих источников данных в проект Visual Studio, отчет может закачивать данные с сервера.

Создание источника данных хранилища

1. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
2. На вкладке **General** введите **TfsReportDS** в текстовое поле **Name**.
3. В списке **Type** выберите **Microsoft SQL Server**.
4. Щелкните **Edit**.
5. Введите имя сервера уровня данных.
6. Выберите базу данных **TFSWarehouse**.
7. Дважды щелкните **OK**, чтобы добавить источник данных.

Создание источника данных OLAP

1. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
2. На вкладке **General** введите **TfsOlapReportDS** в поле **Name**.
3. В списке **Type** выберите **Microsoft SQL Server Analysis Services**.
4. Щелкните **Edit**.
5. Введите имя сервера уровня данных.
6. Выберите базу данных **TFSWarehouse**.
7. Дважды щелкните **OK**, чтобы добавить источник данных.

Шаг 3 — создание нового отчета в проекте

Когда в проект добавлены источники данных, можно создавать новый отчет. Выполните следующие действия:

1. В **Solution Explorer** щелкните правой кнопкой **Reports** и выберите **Add** и **New Item**.
2. Выберите шаблон **Report**.
3. Задайте имя отчета и щелкните **OK**.

Шаг 4 — изменение отчета

Добавив отчет в проект, отредактируйте его:

1. Если **Report Designer** не открывается автоматически, откройте отчет для редактирования, щелкнув его дважды в **Solution Explorer**.
2. Выберите в раскрывающемся списке **Dataset** вариант **New Dataset**.
3. Присвойте набору данных имя, например **TestDataSet**.
4. Выберите **TFSOlapReportDS (shared)**.
5. Щелкните **OK**.
6. Щелкните многоточие рядом с кнопкой **Build** (под раскрывающимся списком **Dataset**) и выберите **Team System**.

Теперь можно редактировать отчет, перетаскивая меры и измерения из дерева **Dataset** на панели **Query Pane** и **Filter Pane**. Компоновка отчета меняется на вкладке **Layout**. Увидеть, как будет выглядеть отчет, можно на вкладке **Preview**.

Шаг 5 — развертывание отчета на Team Foundation Server

После редактирования отчета его можно развернуть на портале отчетов командного проекта:

1. В Solution Explorer щелкните правой кнопкой проект отчетов и выберите **Properties**.
2. Убедитесь, что атрибуту **OverwriteDataSources** присвоено значение **false**.
3. Измените значение **TargetDataSourceFolder** согласно имени своего командного проекта, например:

```
TargetDataSourceFolder = TestProject
```

4. Измените значение **TargetReportFolder** согласно имени своего командного проекта, например:

```
TargetReportFolder = TestProject
```

5. Присвойте параметру **TargetServerURL** значение *http://<имя сервера уровня данных>/reportserver*, например:

```
TargetServerURL = http://tfsrtm/reportserver
```

6. Щелкните **OK.**
7. В Solution Explorer щелкните правой кнопкой файл .rdl и выберите **Deploy**.
8. Посмотрите на **Output Pane**, чтобы убедиться в успешности операции.

Шаг 6 — тестирование отчета

Опубликовав отчет на сервере отчетов своего командного проекта, протестируйте его, чтобы убедиться в успешности развертывания:

1. В **Team Explorer** разверните узел своего командного проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите созданный отчет.
3. Убедитесь, что он выглядит так, как ожидалось.

Дополнительные ресурсы

- Дополнительную информацию по работе с проектами отчетов вы найдете в статье «Reporting Services Tutorials» по адресу *http://msdn2.microsoft.com/en-us/library/ms170246.aspx*.
- О редактировании отчетов читайте в статье «How to: Edit Reports in Report Designer» по адресу *http://msdn2.microsoft.com/en-us/library/ms244655(VS.80).aspx*.
- О ролях системы безопасности уровня данных читайте в статье «Securing Access Through Analysis Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms174839.aspx*.
- О ролях системы безопасности уровня приложений читайте в статье «Securing Reporting Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms157198.aspx*.

Как создать отчет о развитии рисков в Visual Studio 2005 Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).
- Microsoft SQL Server™ Reporting Services.

Описание

В этой статье подробно разбирается процесс создания нового отчета, представляющего динамику развития рисков, и его публикации на портале отчетов группы в TFS.

Содержание

Задачи

- Создать проект отчетов в Visual Studio.
- Создать новый отчет о динамике рисков в проекте отчетов.
- Опубликовать отчет на сервере отчетов.

Обзор

Отчеты, поставляемые с VSTS, основаны на использовании SQL Server Reporting Services. С помощью конструктора отчетов Visual Studio (Business Intelligence Development Studio) из комплекта клиентских инструментов SQL Server 2005 можно редактировать готовые отчеты или создавать собственные. Создание собственного отчета в Visual Studio начинается с создания проекта отчетов. Затем создаются источники данных для подключения к реляционной базе данных TFS и базе данных Online Analytical Processing (OLAP). В этой статье рассказано, как создать с нуля простой отчет о динамике рисков (Risk over Time), который показывает количество рабочих элементов Risk за данный период времени.

Порядок операций

- Шаг 1 — создание нового проекта отчетов.
- Шаг 2 — создание источников данных.
- Шаг 3 — создание нового отчета в проекте.
- Шаг 4 — изменение отчета.
- Шаг 5 — развертывание отчета на Team Foundation Server.
- Шаг 6 — тестирование отчета.

Прежде всего

Прежде чем приступать к настройке отчета для Team Foundation Server, убедитесь в следующем:

- На компьютере, который будет использоваться для настройки отчета, должна быть установлена среда Business Intelligence Development Studio. Чтобы проверить ее наличие, при создании нового проекта посмотрите, имеется ли в Visual Studio тип Business Intelligence Project.
- Ваша учетная запись должна быть членом роли безопасности Microsoft Analysis Server TfsWarehouseDataReaders на сервере уровня данных.
- Ваша учетная запись должна обладать правами администратора БД TFSWarehouse уровня данных.
- Ваша учетная запись должна быть членом роли Publisher в SQL Server Reporting Services на сервере уровня приложений.

- Проект должен содержать рабочие элементы Risk, иначе в отчете нечего будет показывать.

Шаг 1 — создание нового проекта отчетов

На данном начальном этапе создается новый проект создания отчетов, благодаря которому вы получаете возможность добавлять новый отчет в проект и затем настраивать его. Чтобы создать новый проект создания отчетов в Visual Studio:

1. В меню **File** выберите **New** и затем щелкните **Project**.
2. Выберите тип **Business Intelligence Project**.
3. Выберите шаблон **Report Server Project**.
4. Задайте **Name** и **Location** для проекта и щелкните **OK**.

Шаг 2 — создание источников данных

Чтобы редактировать и публиковать настроенный отчет, необходимо добавить источники данных для хранилища данных Team Foundation Server и OLAP-куб. После добавления этих источников данных в проект Visual Studio, отчет может закачивать данные с сервера.

Создание источника данных хранилища

1. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
2. На вкладке **General** введите **TfsReportDS** в текстовое поле **Name**.
3. В списке **Type** выберите **Microsoft SQL Server**.
4. Щелкните **Edit**.
5. Введите имя сервера уровня данных.
6. Выберите базу данных **TFSWarehouse**.
7. Дважды щелкните **OK**, чтобы добавить источник данных.

Создание источника данных OLAP

1. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
2. На вкладке **General** введите **TfsOlapReportDS** в поле **Name**.
3. В списке **Type** выберите **Microsoft SQL Server Analysis Services**.
4. Щелкните **Edit**.
5. Введите имя сервера уровня данных.
6. Выберите базу данных **TFSWarehouse**.
7. Дважды щелкните **OK**, чтобы добавить источник данных.

Шаг 3 — создание нового отчета в проекте

Когда в проект добавлены источники данных, можно создавать новый отчет. Выполните следующие действия:

1. В **Solution Explorer** щелкните правой кнопкой **Reports** и выберите **Add** и **New Item**.
2. Выберите шаблон **Report**.
3. Задайте имя отчета и щелкните **OK**.

Шаг 4 — изменение отчета

Когда отчет добавлен в проект, его можно редактировать:

1. Если **Report Designer** не открывается автоматически, откройте отчет для редактирования, щелкнув его дважды в **Solution Explorer**.
2. Выберите в раскрывающемся списке **Dataset** вариант **New Dataset**.
3. Присвойте имя набору данных, например, **TestDataSet**.
4. Выберите **TFSOlapReportDS (shared)** и щелкните **OK.**
5. Щелкните многоточие рядом с **Build** (под раскрывающимся списком **Dataset**) и выберите **Team System**.
6. В дереве **Dataset Tree** разверните узел **Measures**.
7. В дереве **Dataset Tree** разверните узел **Current Work Item**.
8. Перетащите **Current Work Item Count** в главное окно запроса.
9. В дереве **Dataset Tree** сверните узел **Measures**.
10. Перейдите к узлу **Team Project** и перетащите его в панель **Dimensions Grid**.
11. В панели **Dimensions Grid** щелкните ячейку **Filter Expression** и выберите имя вашего командного проекта. После этого в отчет будут включаться только данные, касающиеся этого проекта.
12. Разверните измерение **Work Item** в дереве **Dataset Tree**.
13. Перетащите **WorkItem.WorkItemType** из дерева **Dataset Tree** в панель **Dimensions Grid**. Если вместо **WorkItem.WorkItemType** отображается **System_WorkItemType**, это означает, что отчет все равно сработает, но вам необходимо установить SQL Server Service Pack 2.
14. Перетащите **WorkItem.WorkItemType** из дерева **Dataset Tree** в главное окно запроса и разместите перед столбцом **work item count**. Если вместо **WorkItem.WorkItemType** отображается **System_WorkItemType**, это означает, что отчет все равно сработает, но вам необходимо установить SQL Server Service Pack 2.
15. В **Dimensions Grid** щелкните ячейку **Filter Expression** и выберите тип **Risk**. Это обеспечит включение в отчет только рабочих элементов Risk.

16. В **Dataset Tree** разверните измерение **Date**.
17. Перетащите значение измерения **Date** в главное окно запроса. Разместите его перед столбом **work item type**.
18. Перейдите на вкладку **Layout**.
19. Откройте окно **Toolbox**.
20. Перетащите элемент **Chart** из **Toolbox** на сетку.
21. Настройте размеры диаграммы.
22. Щелкните диаграмму правой кнопкой и выберите **Chart Type**, **Line** и **Smooth Line**.
23. Откройте панель **Datasets Pane**.
24. Разверните свой набор данных, например, **TestDataSet**.
25. Выделите мышью диаграмму. В ней появятся области для перетаскивания **Data**, **Series** и **Category**.
26. Перетащите **Current_Work_Item_Count** в поле **Drop Data Fields Here**.
27. Перетащите **Work_Item_Type** в поле **Drop Series Fields Here**.
28. Перетащите **Date** в поле **Drop Category Fields Here**.
29. Щелкните график правой кнопкой мыши и выберите **Properties**.
30. Введите название графика и щелкните **OK**.
31. Перейдите на вкладку **Preview**, чтобы посмотреть, как будет выглядеть отчет.

Шаг 5 — развертывание отчета на Team Foundation Server

После редактирования отчета Risk over Time его можно развернуть на портале отчетов командного проекта:

1. В Solution Explorer щелкните правой кнопкой проект отчетов и выберите **Properties**.
2. Убедитесь, что атрибуту **OverwriteDataSources** присвоено значение **false**.
3. Измените значение **TargetDataSourceFolder** согласно имени своего командного проекта, например:

```
TargetDataSourceFolder = TestProject
```

4. Измените значение **TargetReportFolder** согласно имени своего командного проекта, например:

```
TargetReportFolder = TestProject
```

5. Присвойте параметру **TargetServerURL** значение *http://<имя сервера уровня данных>/reportserver*, например:

```
TargetServerURL = http://tfsrtm/reportserver
```

6. Щелкните **OK.**
7. В Solution Explorer щелкните правой кнопкой файл .rdl и выберите **Deploy**.
8. Посмотрите на **Output Pane**, чтобы убедиться в успешности операции.

Шаг 6 — тестирование отчета

Опубликовав отчет на сервере отчетов своего командного проекта, протестируйте его, чтобы убедиться в успешности развертывания:

1. В **Team Explorer** разверните узел своего командного проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите созданный отчет.
3. Убедитесь, что он выглядит так, как ожидалось.

Дополнительные ресурсы

- Дополнительную информацию по работе с проектами отчетов вы найдете в статье «Reporting Services Tutorials» по адресу *http://msdn2.microsoft.com/en-us/library/ms170246.aspx*.
- О редактировании отчетов читайте в статье «How to: Edit Reports in Report Designer» по адресу *http://msdn2.microsoft.com/en-us/library/ms244655(VS.80).aspx*.
- О ролях системы безопасности уровня данных читайте в статье «Securing Access Through Analysis Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms174839.aspx*.
- О ролях системы безопасности уровня приложений читайте в статье «Securing Reporting Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms157198.aspx*.

Как создать пользовательскую политику возврата после правки в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System.

Описание

В этой статье подробно разбирается процесс создания, регистрации и применения пользовательских политик возврата правок в TFS. Политика обеспечивает выполнение определенных правил при каждом возврате изменений в систему управления исходным кодом. Благодаря этому исходный код гарантированно будет соответствовать заданному набору критериев. В качестве примера в этой статье используется политика, обеспечивающая ввод комментариев ко всем возвратам изменений. Для реализации пользовательской политики возврата изменений создается класс, производный от **PolicyBase**, и реализуются интерфейсы **IPolicyDefinition** и **IPolicyEvaluation**. Сборка политики регистрируется в реестре Microsoft Windows® и применяется к командному проекту.

Содержание

- Порядок операций.
- Шаг 1 — создание и сборка класса пользовательской политики.
- Шаг 2 — регистрация класса пользовательской политики в реестре Windows.
- Шаг 3 — применение пользовательской политики.
- Шаг 4 — проверка работоспособности пользовательской политики.
- Дополнительные ресурсы.

Задачи

- Разобраться с тем, что такое пользовательская политика возврата изменений.
- Научиться создавать, регистрировать и применять специальные политики возврата изменений.

Обзор

Политики возврата после правки (check-in policies) обеспечивают реализацию ограничений при возврате файлов в систему управления исходным кодом. В Team Foundation Server включено несколько стандартных политик, в том числе, политики для проверки выполнения и успешного прохождения модульных тестов, политики для выполнения статического анализа кода, гарантирующие соответствие кода стандартам и рекомендациям написания .NET-кода, и политики, обеспечивающие связывание рабочих элементов с возвратами правок. Еще несколько политик возврата после правки включено в комплект Microsoft Visual Studio 2005 Team Foundation Power Tool. В этой статье рассказывается, как создавать, регистрировать и применять нестандартную политику. В качестве примера приводится политика, вынуждающая разработчиков сопровождать возвращаемые изменения комментариями.

Прежде всего

Чтобы вы могли создавать политику возврата после правки, разрешение **Manipulate** у вас должно иметь значение **Allow**.

Порядок операций

- Шаг 1 — создание и сборка класса пользовательской политики.
- Шаг 2 — регистрация класса пользовательской политики в реестре Windows.
- Шаг 3 — применение пользовательской политики.
- Шаг 4 — проверка работоспособности пользовательской политики.

Шаг 1 — создание и сборка класса пользовательской политики

На начальном этапе в пространстве имен **Microsoft.TeamFoundation.VersionControl.Client** путем наследования от базового класса **PolicyBase** создается класс пользовательской политики. Наследование обеспечивает реализацию создаваемым классом интерфейсов **IPolicyDefinition** и **IPolicyEvaluation**. Ниже приведен пример фрагмента кода политики, которая требует обязательного ввода комментариев при каждом возврате правок.

1. В Visual Studio создайте новый проект библиотеки классов Visual C#®.
2. Добавьте ссылку на файл сборки **System.Windows.Forms.dll**. Эта сборка используется для отображения информационных окон.
3. Добавьте ссылку на файл сборки **Microsoft.TeamFoundation.Version Control.Client.dll.** По умолчанию он находится в папке \Program Files\ Visual Studio 2005 Team Foundation Server\Tools.
4. Замените шаблонную реализацию класса следующим исходным кодом. Обратите внимание, что класс наследуется от базового класса **PolicyBase** и помечен как сериализуемый.

```
using System;
using System.Windows.Forms;
using Microsoft.TeamFoundation.VersionControl.Client;

[Serializable]
public class CheckForCommentsPolicy : PolicyBase
{
   public override string Description
   {
   get { return “Напоминает пользователям, что они должны сопровождать
возврат изменений содержательными комментариями”;
   }

   // Эта строка хранится с описанием политики на сервере системы управ-
ления исходным кодом.
   // Если у пользователя не установлена надстройка политики, отобража-
ется эта
   // строка. С ее помощью вы объясняете пользователю, как установить
   // надстройку политики.
   public override string InstallationInstructions
   {
      get { return “Инструкции по установке этой политики см. в
InstallInstructions.txt.”; }
   }

   // Эта строка определяет тип политики. Она отображается в списке по-
литик, когда вы
```

```
   // добавляете новую политику в Team Project.
   public override string Type
   {
      get { return “Политика для проверки наличия комментариев к возвра-
ту изменений”; }
   }

   // Эта строка является описанием типа политики. Она отображается,
когда вы
   // выбираете политику в диалоговом окне Add Check-in Policy.
   public override string TypeDescription
   {
      get { return “Эта политика подскажет пользователю, на каких усло-
виях ему будет разрешено возвратить изменения.”; }
   }

   // Этот метод вызывается инфраструктурой написания политик при создании
   // новой политики возврата изменений или редактировании существующей.
   // Он может использоваться для отображения интерфейса для данного
типа политики,
   // позволяя пользователю менять параметры политики.
   public override bool Edit(IPolicyEditArgs args)
   {
      // Не требует специальной конфигурации
      return true;
   }

   // Этот метод является фактической реализацией политики. Он вызывается
   // инфраструктурой создания политик, когда требуется применить политику.
   // В приведеннм примере метод вызывается при возникновении различных
асинхронных
   // событий, которые могли привести к недействительности текущего списка
   // нарушений политики.
   public override PolicyFailure[] Evaluate()
   {
      string proposedComment = PendingCheckin.PendingChanges.Comment;
      if (String.IsNullOrEmpty(proposedComment))
      {
         return new PolicyFailure[] {
            new PolicyFailure(“Пожалуйста, сопроводите возвращаемые из-
менения комментариями”, this) };
      }
      else
      {
         return new PolicyFailure[0];
      }
   }
```

```
    // Этот метод вызывается двойным щелчком нарушения политики в интерфейсе.
    // В данном случае на экран выводится сообщение, предлагающее пользователю
    // ввести комментарий.
    public override void Activate(PolicyFailure failure)
    {
        MessageBox.Show("Пожалуйста, сопроводите возвращаемые изменения
комментариями.", "Как устранить нарушение политики");
    }

    // Этот метод вызывается, если пользователь нажимает кнопку F1 при
активном
    // нарушении политики в интерфейсе. В данном примере на экран выво-
дится сообщение.
    public override void DisplayHelp(PolicyFailure failure)
    {
        MessageBox.Show("Данная политика напоминает о необходимости сопро-
вождения возвращаемых изменений комментариями.", "Prompt Policy Help");
    }
}
```

Шаг 2 — регистрация класса пользовательской политики в реестре Windows

На этом шаге вы добавите запись в реестр Windows, благодаря чему ваша политика будет отображаться в диалоговом окне **Add Check-in Policy**. Учтите, что сборка политики должна быть установлена на все компьютеры, где предполагается ее использовать. К ним относятся компьютер администратора командного проекта, который должен связать политику с проектом, и все компьютеры членов команды, на которых эта политика фактически реализуется.

> **Важно!** Политика проверяется на клиенте, когда разработчик возвращает файл в систему управления исходным кодом.

1. Запустите Regedit.exe и найдите раздел HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\TeamFoundation\SourceControl\Checkin Policies. Список зарегистрированных политик выводится в правой панели.
2. Щелкните правой кнопкой правую панель, выберите команду **Создать (New)** и щелкните **Строковый параметр (String Value)**.
3. Введите имя DLL-библиотеки своей пользовательской политики без расширения, например, **CheckForCommentsPolicy**, как в рассмотренном выше примере.

> **Важно!** Строка должна точно соответствовать имени DLL-файла — без расширения DLL.

4. Щелкните дважды новой строковый параметр и задайте в качестве значения полный путь и имя файла .dll.

Шаг 3 — применение пользовательской политики

Теперь добавим пользовательскую политику в командный проект. Это гарантирует, что политика будет выполняться при каждом возврате файла.

1. В **Team Explorer** щелкните правой кнопкой свой командный проект, выберите **Team Project Settings** и щелкните **Source Control**.
2. Перейдите на вкладку **Check-in Policy** и щелкните **Add**.
3. Выберите политику **Check for Comments Policy** и дважды щелкните **OK**.

Теперь политика будет применяться при каждом возврате файла в этом командном проекте.

Шаг 4 — проверка работоспособности пользовательской политики

Попробуйте возвратить в систему файл исходного кода, чтобы убедиться, что пользовательская политика работает правильно.

1. Внесите изменения в файл и попытайтесь вернуть его без предоставления комментария к возвращаемым правкам.
2. Убедитесь, что возврат изменений приостановлен, потому что не выполнено требование политики возврата.
3. Добавьте комментарии и завершите возврат правок. С комментарием возврат должен пройти успешно, без вывода на экран уведомления о нарушении политики.

Дополнительные ресурсы

- Подробнее о том, как настроить политику возврата после правки, читайте в статье «Walkthrough: Customizing Check-in Policies and Notes» по адресу *http://msdn2.microsoft.com/en-us/library/ms181281(VS.80).aspx*.
- Чтобы посмотреть пример кода, который запрещает возврат правок с определенными вариантами кодирования, читайте статью «Checkin Policy to Disallow Certain Patterns» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx*.
- Чтобы посмотреть пример кода, который обязывает вводить комментарии при возврате после правки, читайте статью «Sample Checkin Policy:

Make Sure the Comment Isn't Empty» по адресу *http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx*.

- Чтобы узнать, как зарегистрировать новую политику возврата после правки, читайте статью «I've Made a New Check-In Policy! How Do I Add It?» по адресу *http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx*.

Как создать дерево кода в Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System.

Описание

В этой статье подробно разбирается процесс создания в TFS дерева исходного кода. Цель статьи — познакомить вас со всеми этапами создания собственного дерева исходного кода.

Содержание

Задачи

- Научиться создавать командный проект.
- Научиться создавать сопоставление рабочей области.
- Научиться создавать дерево исходного кода в системе Team Foundation Server.

Обзор

Чтобы добавить решение в систему управления исходным кодом, достаточно просто щелкнуть его правой кнопкой в Solution Explorer и выбрать команду **Add Solution To Source Control**. Однако такой вариант не позволяет явно настроить структуру дерева исходного кода в системе управления исходным кодом. Явно описывая структуру каталогов, вы можете организовать исходный код под папками верхнего уровня и использовать отдельные папки верхнего уровня для размещения основного исходного кода и его ответвлений, например, ветвей, используемых при разработке или для обслуживания готовых выпусков.

В данной статье описаны шаги, необходимые для явного создания структуры дерева исходного кода.

Порядок операций

- Шаг 1 — создание командного проекта.
- Шаг 2 — создание сопоставления рабочей области.
- Шаг 3 — создание структуры каталогов в системе управления исходным кодом.
- Шаг 4 — добавление исходного кода в дерево.

Шаг 1 — создание командного проекта

Для начала мы создадим новый командный проект с настройками по умолчанию.

1. В **Team Explorer** щелкните правой кнопкой свой сервер TFS и выберите команду **New Team Project**.
2. В диалоговом окне **New Team Project** ведите имя проекта, например **MyTeamProject1**, и щелкните **Next**.
3. На странице **Select a Process Template** оставьте значение по умолчанию — **MSF for Agile Software Development - v4.0** — и щелкните **Next**.
4. На странице **Specify the Settings for the Project Portal** оставьте предлагаемое имя портала проекта (**MyTeamProject1**), введите описание портала проекта и щелкните **Next**.
5. На странице **Specify Source Control Settings** оставьте значение по умолчанию **Create an empty source control folder**, чтобы создать пустую папку системы управления исходным кодом, и щелкните **Next**.
6. Щелкните **Finish**, чтобы создать проект.

На сервере TFS будут созданы новый командный проект с использованием выбранного шаблона процесса и пустая папка для него.

Шаг 2 — создание сопоставления рабочей области

Теперь нужно создать сопоставление рабочей области для описания соответствия между структурой каталогов на сервере TFS и на клиенте. Это необходимо для создания структуры дерева исходного кода. Сначала дерево каталогов создается в вашей рабочей области, затем вы должны передать его на сервер TFS.

Сопоставление рабочей области можно создать двумя способами:

- явно задать сопоставление рабочей области;
- выполнить для своего командного проекта операцию **get**.

Явное задание сопоставления рабочей области

1. В меню **File** Visual Studio выберите команду **Source Control** и щелкните **Workspaces**.
2. В диалоговом окне **Manage Workspaces** выберите имя своего компьютера и щелкните **Edit**.
3. В диалоговом окне **Edit Workspace** в списке **Working folders** щелкните **Click here to enter a new working folder**.
4. Щелкните многоточие, выберите свой командный проект (например, **MyTeamProject1**) и щелкните **OK**.
5. Щелкните ячейку локальной папки, чтобы появилась еще одна кнопка с многоточием.
6. Щелкните многоточие под **Local Folder** и выберите локальную папку на компьютере, где вы хотите разместить рабочую область командного проекта, например, C:\DevProjects\MyTeamProject1.
7. Дважды щелкните **OK**, чтобы закрыть диалоговое окно **Edit Workspace**.
8. Щелкните **OK** в информационном сообщении Microsoft Visual Studio об изменении одной или нескольких рабочих папок.
9. Щелкните **Close**, чтобы закрыть диалоговое окно **Manage Workspaces**.

Выполнение операции Get для командного проекта

1. В **Team Explorer** разверните узел командного проекта **MyTeamProject1**.
2. Щелкните дважды **Source Control**.
3. В **Source Control Explorer** щелкните правой кнопкой мыши корневую папку **MyTeamProject1** и выберите команду **Get Latest Version**.
4. В диалоговом окне **Browse For Folder** выберите нужный локальный путь (например, C:\DevProjects\MyTeamProject1) и щелкните **OK**. Корневая папка командного проекта с TFS будет сопоставлена с локальной папкой на вашем компьютере.

Шаг 3 — создание структуры каталогов в системе управления исходным кодом

На этом этапе исходя из стратегии и требований проекта создается структура каталогов системы управления исходным кодом на сервере. Обычно за основу берется структура /Main/Source, которая позволяет впоследствии создавать на одном уровне с **Main** ветви **Development** и **Releases**. В папке **Releases** размещаются ветви кода выпущенных версий ПО, для которых вы обеспечиваете поддержку. Папка **Development** содержит изолированные ветви разработки.

```
/Main
   /Source
      /MyApp1                    → Содержит MyApp1.sln
         /Source                 → Папка-контейнер
            /ClassLibrary1       → Содержит ClassLibrary1.csproj
            /MyApp1Web           → Содержит Default.aspx
         /UnitTests              → Содержит проекты модульных тестов
            /ClassLibrary1Tests  → Проект тестирования для ClassLibrary1
            /MyApp1WebTests      → Проект тестирования для MyApp1Web
   /Build                        → Содержит результат сборки (двоичные файлы)
   /Docs                         → Содержит проектную документацию и пр.
   /TestCases                    → Содержит документацию по тестированию
/Development
   /FeatureBranch1
      /Source
         /MyApp1
            /Source
               /MyApp1Web
               /ClassLibrary1
            /UnitTests
               /ClassLibrary1Tests
               /MyApp1WebTests
   /FeatureBranch2

/Releases
/Release1
         /MyApp1
            /Source
               /ClassLibrary1
               /MyApp1Web
            /UnitTests
               /ClassLibrary1Tests
               /MyApp1WebTests
/Release 1.1
/Release 1.2
```

Создание структуры каталогов на сервере:

1. В **Team Explorer** разверните узел командного проекта **MyTeamProject1**.
2. Дважды щелкните **Source Control**.
3. В **Source Control Explorer** выберите корневой узел, щелкните правой кнопкой мыши панель **Local Path** и выберите команду **New Folder**.
4. Введите имя **Main** и нажмите Enter.
5. В папке **Main** создайте папку **Source**.
6. Повторите предыдущие шаги, чтобы создать другие корневые папки, например, **Development** и **Releases**.
7. Создав структуру дерева каталогов, щелкните правой кнопкой мыши корневой узел **MyTeamProject1** в **Source Control Explorer** и выберите команду **Check-in Pending Changes.**
8. В диалоговом окне **Check In - Source Files — Workspace** выберите папки, которые необходимо возвратить в систему управления исходным кодом, добавьте комментарий и щелкните **Check In**. Структура каталогов будет создана локально и добавлена в систему управления исходным кодом TFS.

Шаг 4 — добавление исходного кода в дерево

На этом этапе исходный код копируется с локального диска в дерево каталогов на сервере. В этом примере вы создадите новое веб-приложение и проект библиотеки классов, а затем добавите их в систему управления исходным кодом.

Создание нового файла решения Visual Studio

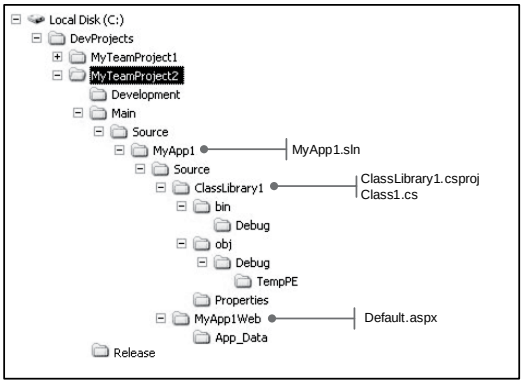
1. Выберите в меню **File** команду **New** и щелкните **Project**.
2. Разверните **Other Project Types** и выберите **Visual Studio Solutions**.
3. На панели **Templates** выберите **Blank Solution**.
4. Введите **MyApp1** в поле **Name** и **C:\DevProjects\MyTeamProject1\Main\Source** в поле **Location**.
5. Щелкните **OK**.Visual Studio создаст новое решение и поместит файл решения (.sln) в папку C:\DevProjects\ MyTeamProject1\Main\Source\MyApp1.

Добавление в решение нового веб-сайта

1. В Solution Explorer щелкните решение правой кнопкой, выберите **Add** и щелкните **New Web Site**.
2. В списке **Templates** выберите **ASP.NET Web Site**, задайте **File System** в качестве **Location** и **C:\DevProjects\MyTeamProject1\Main\Source\MyApp1\Source\MyApp1Web** в качестве пути.
3. Щелкните **OK**. Visual Studio создаст веб-сайт.

Добавление в решение нового проекта библиотеки классов

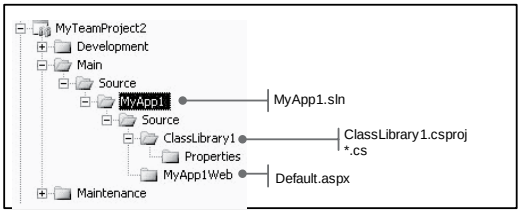
1. В Solution Explorer щелкните решение правой кнопкой, выберите **Add** и щелкните **New Project**.
2. В списке **Project types** выберите **Visual C#**, а в списке **Templates** выберите **Class Library**.
3. Не меняйте предлагаемое по умолчанию имя **ClassLibrary1** и задайте в поле **Location** путь **C:\DevProjects\MyTeamProject1\Main\Source\MyApp1\Source**.
4. Щелкните **OK**. Visual Studio создаст структуру нового проекта. Теперь локальная структура каталогов должна выглядеть следующим образом:



Добавление решения в систему управления исходным кодом

В Solution Explorer щелкните решение правой кнопкой мыши и выберите **Add Solution to Source Control**. Ваше решение и два проекта будут добавлены в Team Foundation Source Control.

Теперь дерево исходного кода в системе управления исходным кодом должно выглядеть так:



Дополнительные ресурсы

- Подробнее о создании рабочей области читайте в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.

Как настроить шаблон процесса в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).

Описание

В этой статье подробно разбирается изменение шаблона процесса, направленное на приведение его в соответствие с потребностями команды, а также рассматриваются используемые для этого инструменты. В шаблонах процесса описаны исходные настройки процесса для командных проектов. Настраивая шаблон процесса, можно задать:

- параметры безопасности для управления доступом к командному проекту;
- шаблоны, доступные на портале проекта Microsoft Office SharePoint®;
- наличие комментариев к изменениям, возвращаемым в систему управления исходным кодом;
- типы и запросы рабочих элементов;
- отчеты для контроля за разработкой проекта и его состоянием;
- итерации и области, используемые для организации проекта.

Последовательно пройдите все этапы, измените шаблон, а затем протестируйте внесенные изменения.

Содержание

- Шаг 1 — установка редактора процесса.
- Шаг 2 — выбор шаблона процесса.
- Шаг 3 — загрузка шаблона процесса.
- Шаг 4 — открытие шаблона в редакторе процесса.
- Шаг 5 — изменение типов рабочих элементов.
- Шаг 6 — изменение стандартных рабочих элементов.
- Шаг 7 — изменение и управление запросами.
- Шаг 8 — изменение областей и итераций.
- Шаг 9 — изменение групп и прав доступа.
- Шаг 10 — изменение настроек системы управления исходным кодом.
- Шаг 11 — изменение портала проекта.
- Шаг 12 — изменение отчетов.
- Шаг 13 — выгрузка измененного шаблона процесса на сервер.
- Дополнительные ресурсы.

Задачи

- Разобраться, какие параметры шаблона процесса можно изменять.
- Настроить шаблон процесса, используя Process Editor.

Обзор

Visual Studio Team System (VSTS) и TFS предоставляют собой интегрированную среду с поддержкой большинства этапов процесса разработки ПО. Методология поддержки рабочего цикла командных проектов реализована в TFS при помощи шаблонов процессов. *Шаблон процесса* (process template) — это набор XML-файлов со спецификациями процессов и артефактов, составляющих конкретную методологию. Редактирование шаблона процесса заключается в изменении стандартных типов рабочих элементов, настроек безопасности, параметров системы управления исходным кодом и отчетов.

Изменить шаблон можно, вручную редактируя XML-файлы, однако проще делать это при помощи инструмента Process Editor из комплекта Team Foundation Server Power Tool. Его интерфейс существенно упрощает процесс настройки. К тому же, редактирование XML-файлов вручную чревато большим количеством ошибок.

Чтобы настроить шаблон процесса, необходимо скачать его с сервера, а затем с помощью Process Editor внести необходимые изменения.

Порядок операций

- Шаг 1 — установка редактора процессов.
- Шаг 2 — выбор шаблона процесса.
- Шаг 3 — загрузка шаблона процесса.
- Шаг 4 — открытие шаблона в редакторе процесса.
- Шаг 5 — изменение типов рабочих элементов.
- Шаг 6 — изменение стандартных рабочих элементов.
- Шаг 7 — изменение и управление запросами.
- Шаг 8 — изменение областей и итераций.
- Шаг 9 — изменение групп и прав доступа.
- Шаг 10 — изменение настроек системы управления исходным кодом.
- Шаг 11 — изменение портала проекта.
- Шаг 12 — изменение отчетов.
- Шаг 13 — выгрузка измененного шаблона процесса на сервер.

Шаг 1 — установка редактора процессов

На начальном этапе выполняется установка инструмента Process Editor — удобного средства просмотра и настройки шаблонов процессов. Process Editor — это часть Team Foundation Server Power Tool, представляющая собой набор расширений, инструментов и утилит командной строки.

1. Прежде чем запускать программу установки Power Tool, установите DSL Tools для Visual Studio 2005. Это необходимое условие для установки Team System Process Editor. Если не установить DSL Tools, программа установки Power Tool установит все, кроме Process Editor. Скачать DSL Tools можно по адресу *http://go.microsoft.com/fwlink/?LinkId=82410*.
2. Скачайте Power Tool по адресу *http://www.microsoft.com/downloads/details.aspx?familyid=7324c3db-658d-441b-8522-689c557d0a79&displaylang=en*.
3. Выполните стандартную установку и проверьте, правильно ли установлен Power Tool, т.е., установлен ли он вместе с Process Editor:
   а) Щелкните **Start** и **Programs**.
   б) Щелкните **Microsoft Team Foundation Server Power Tool**.
   в) Если появляется команда для вызова Microsoft Visual Studio Team System Process Editor, значит, Process Editor установлен правильно. В противном случае удалите Power Tool и установите повторно, строго придерживаясь описанной ранее последовательности действий.

Шаг 2 — выбор шаблона процесса

Теперь выберите стандартный шаблон, наиболее близко соответствующий вашему процессу, чтобы для адаптации шаблона к процессу его не пришлось менять слишком сильно. С TFS поставляются два шаблона:

- Microsoft Solution Framework (MSF) for Agile Software Development (MSF Agile) — шаблон для гибкой разработки ПО.
- MSF for CMMI® Process Improvement (MSF CMMI) — шаблон для совершенствования процесса согласно рекомендациям CMMI.

MSF Agile

Шаблон процесса MSF Agile прост и потому подходит, главным образом, для небольших, срочных или неформальных проектов по разработке ПО. Он основан на сценариях и действиях, управляемых контекстом, и ориентирован как на проект, так и на участников (людей). Шаблон процесса MSF Agile может использоваться в следующих сценариях:

- Процесс разработки не документируется и не опирается на формальные процессы.
- Не требуется сертификация процесса или подтверждение его соответствия стандартам разработки.
- Разработка проекта будет непродолжительной.
- Версии продукта должны выходить часто, чтобы оперативно получать обратную информацию от сообщества пользователей.

MSF CMMI

Шаблон процесса MSF CMMI предназначен для более серьезных проектов по разработке ПО. Он расширяет функциональность шаблона MSF Agile, предоставляя поддержку аудита, верификации и формальных процессов. Этот шаблон может использоваться в следующих сценариях:

- Процесс выполнения проекта документируется или формализован.
- Предсказуемость выпускаемых версий более важна, чем гибкость.
- Проект рассчитан на длительный срок.
- Необходима сертификация процесса или подтверждение его соответствия стандартам разработки.

Примечание Вы не ограничены стандартными шаблонами. Если у вас установлены шаблоны процессов сторонних производителей, например, шаблон процесса Scrum, вы вольны использовать и их.

Шаг 3 — загрузка шаблона процесса

Скачайте выбранный шаблон процесса, чтобы далее изменить его соответственно своему процессу.

1. В Visual Studio выберите в меню **Team** команду **Team Foundation Server Settings**.
2. Щелкните **Process Template Manager**.
3. В диалоговом окне **Process Template Manager** выберите процесс, который хотите изменить, и щелкните **Download**.
4. В диалоговом окне **Download Process Template** выберите папку на локальном диске, в которую хотите сохранить шаблон, и щелкните **Save**.

Например, можно загрузить шаблон процесса MSF CMMI и разместить его на своем рабочем столе для использования на следующих шагах.

Шаг 4 — открытие шаблона в редакторе процесса

Загрузите скачанный шаблон процесса в Process Editor для изменения различных параметров.

1. В Visual Studio раскройте меню **Team**.
2. Щелкните **Process Editor** и выберите **Open Process Template**.
3. В диалоговом окне **Open Process Template fileset** перейдите к загруженному шаблону процесса и щелкните **Open**, чтобы открыть файл ProcessTemplate.xml в Visual Studio.
4. Задайте имя настраиваемой методологии.
   Допустим, вы назвали новый шаблон процесса **My Template**.

Шаг 5 — изменение типов рабочих элементов

На данном этапе вы создадите новые типы рабочих элементов, характерные для вашего процесса, или внесете изменения в существующие типы рабочих элементов.

Добавление новых типов рабочих элементов

1. В Process Template Explorer щелкните **Work Item Tracking**.
2. В правой панели перейдите на вкладку **Type Definitions**.
3. Чтобы создать новый рабочий элемент, щелкните **Add** на панели инструментов правой панели.
4. В диалоговом окне **New Work Item Type** введите имя типа рабочего элемента или выберите существующий тип из раскрывающегося списка **Copy From**.

5. Новый тип рабочего элемента будет создан и включен в список **Item List** на вкладке **Type Definitions** правой панели.
6. В меню **File** выберите команду **Save**, чтобы сохранить изменения.

Например, можно создать новую ошибку «My Bug», скопировав рабочий элемент Bug, а затем отредактировать тип рабочего элемента и изменить его поведение.

Редактирование типа рабочего элемента

Чтобы добавить или удалить поля атрибутов нового или существующего типа рабочего элемента, на вкладке **Type Definitions** щелкните правой кнопкой необходимый тип и выберите команду **Open**. Выбранный тип рабочего элемента открывается в новом окне Visual Studio, где можно добавлять или удалять необходимые атрибуты.

Добавление в рабочий элемент нового поля

1. Перейдите на вкладку **Fields** и щелкните **Add** на панели инструментов.
2. В диалоговом окне **Field Definition** введите следующие данные:
   а. **Name** — имя нового поля.
   б. **Type** — тип данных для поля.
   в. **RefName** — уникальный идентификатор поля в TFS. В имени идентификатора должна быть, по крайней мере, одна точка, например, Test.Test1.
   г. **Help Text** — текст, который будет видеть пользователь, поместив указатель над именем поля. Как правило, это описание поля.
   д. **Reportable** — атрибут, определяющий, будут ли данные поля доступны для включения в отчеты TFS. Возможны следующие варианты: **Dimension**, **Detail** и **Measure.** Первый — измерение — предназначен для полей со списком допустимых значений. Хорошие примеры измерений — Work Item Type и State. Поле **Dimension** может использоваться для фильтрации отчетов. Вариант **Detail** подходит для отчетов, поскольку разрешает размещать текст без ограничений по длине. Отчеты, включающие эти поля, должны опираться не на куб OLAP, а на реляционную базу данных. Наконец, **Measure** — это числовые значения в отчетах. Каждая мера появляется и в группе мер **Current Work Item**, и в группе мер **Work Item History**.
   е. **Formula** — раскрывающийся список, который отображается, только если вы выбрали **Measure** в поле **Reportable**. К числу его параметров относятся **sum**, **count** и **avg**.
3. Для любого поля, добавляемого в тип рабочего элемента, можно ввести ограничения на допустимые значения. Также можно наложить ограничение на поле для определенного состояния рабочего элемента и при опре-

деленном переходе рабочего элемента из состояния в состояние. Чтобы наложить ограничения, выполните следующие действия:

а. Перейдите на вкладку **Rules** и щелкните **New**.

б. В диалоговом окне **Select a rule type** выберите тип правила. Чаще всего используются такие типы:

- **AllowedValues** Используйте этот тип, чтобы задать список значений, которые можно вводить в это поле.
- **DefaultValue** Используйте этот тип, чтобы назначить полю значение по умолчанию.
- **Required** Используйте этот тип, если пользователь обязательно должен задавать значение этого поля.
- **ValidUser** Содержит имя пользователя проекта.

в. Дважды щелкните **OK**, чтобы сохранить изменения.

4. Добавив поле в рабочий элемент, решите, где и когда это поле должно отображаться. Чтобы добавить поле в разметку рабочего элемента, выполните следующее:

а. На странице **Work Item Type** перейдите на вкладку **Layout**.

б. На вкладке **Layout** выберите в дереве компоновки место, где хотите разместить новое поле.

в. На левой панели щелкните правой кнопкой выбранный узел и выберите **New Control**.

г. Добавьте соответствующую метку, а затем в качестве значения свойства **FieldName** элемента управления используйте идентификатор **RefName**, введенный для поля ранее.

д. Щелкните **Preview Form**, чтобы убедиться, что новый элемент управления правильно расположен на форме рабочего элемента.

В качестве примера добавим в созданный ранее тип рабочего элемента My Bug новое поле Security Impact. Выполните следующие действия:

1. Перейдите на вкладку **Fields** и щелкните **Add** на панели инструментов.
2. В диалоговом окне **Field Definition** введите следующие данные:

а. **Name** — Security Impact

б. **Type** — String

в. **RefName** — MyBug.CustomField.1

г. **Help Text** — степень влияния ошибки на безопасность, значения в диапазоне от 1 до 5, где 1 представляет самое слабое влияние, а 5 — самое сильное.

д. **Reportable** — задайте значение **Dimension**, чтобы иметь возможность создать отчет об ошибках, которые влияют на безопасность.

3. Перейдите на вкладку **Rules** и щелкните кнопку **Add**.

4. В диалоговом окне **Select a rule type** установите параметр **ALLOWEDVALUES** и щелкните **OK**.
5. В диалоговом окне **ALLOWEDVALUES** пятикратно щелкните кнопку **Add**, добавив значения от 1 до 5.
6. Щелкните **OK**.
7. Перейдите на вкладку **Layout**.
8. Добавьте новое поле в группу **Status**:
   а. Под заголовком **Group — Status** щелкните правой кнопкой мыши второй узел **Column** и выберите **New Control**.
   б. Присвойте параметру **FieldName** значение **MyBug.CustomField.1**.
   в. Присвойте параметру **Label** значение **S&ecurity Impact**. Символ, следующий за «&», будет «горячей» клавишей для этого поля.
   г. Щелкните **Preview Form**, чтобы убедиться, что новое поле размещено правильно.

Удаление поля из рабочего элемента

Поля часто упоминаются в запросах, отчетах и т.д. Чтобы не разыскивать и не удалять все эти ссылки, можно оставить поля в системе, но удалить их из разметки диалогового окна рабочего элемента, чтобы лишить пользователя возможности задавать их значения.

Удалим одно из полей созданного ранее типа рабочего элемента My Bug:

1. Перейдите на вкладку **Layout**, чтобы вывести на экран разметку формы для элемента My Bug.
2. На вкладке **Layout** в дереве компоновки разверните ветвь *TabPage Details Group Column Group Schedule Column*, щелкните правой кнопкой мыши **Remaining &work** и выберите команду **Delete**, чтобы удалить поле из разметки.

В будущем поле можно будет вернуть в разметку, потому что фактически само поле не было удалено.

Редактирование последовательности операций рабочего элемента

После создания типа рабочего элемента и компоновки полей наступает черед заключительного этапа — редактирования последовательности операций типа рабочего элемента. Последовательность определяет состояния, в которых может находиться рабочий элемент в ходе своего жизненного цикла, и переходы между этими состояниями. Поскольку наш рабочий элемент является копией существующего рабочего элемента, его последовательность операций также была скопирована. Теперь необходимо пересмотреть эту последовательность и принять решение о том, как ее следует откорректировать. Чтобы изменить последовательность операций рабочего элемента, выполните следующие действия:

1. Создайте состояния и переходы между состояниями.
   а. Перейдите на вкладку **Workflow** (резюме операций), чтобы открыть окно **Workflow**.
   б. Чтобы создать новое состояние, перетащите состояние из панели инструментов **WITDesigner** в окно **Workflow**.
   в. Чтобы создать переход между двумя состояниями, в панели инструментов **WITDesigner** щелкните значок **Transition Link**, а затем щелкните начальное и конечное состояние.
   г. Чтобы задать исходное состояние (то есть, состояние только что созданного рабочего элемента), щелкните **Transition Link**, затем щелкните пустую область диаграммы, затем щелкните начальное состояние. При этом будет создан переход, не имеющий начального состояния. На диаграмме должен присутствовать только один такой переход. Если он уже есть, его надо удалить.
   д. Чтобы увидеть список атрибутов каждого из состояний или переходов, щелкните значок развертывания в верхнем правом углу фигуры. В списке полей значком «+» обозначено обязательное для заполнения поле; значком «–» обозначено необязательное для заполнения поле; значком «*» обозначено поле, имеющее другие правила.
   е. Чтобы редактировать атрибуты состояния или перехода, щелкните правой кнопкой мыши заголовок состояния и выберите **Open Details** или просто дважды щелкните заголовок.

Примечание Если панель инструментов **WITDesigner** не отображается, выберите команду **Toolbox** в меню **View** Visual Studio.

---

2. Отредактируйте правила, управляющие состоянием:
   а. Щелкните правой кнопкой заголовок состояния и выберите команду **Open Details** или щелкните заголовок дважды. Откроется диалоговое окно **Workflow State Field Rules**. Каждый элемент списка ссылается на поле типа рабочего элемента и представляет набор правил, управляющих полем, когда рабочий элемент находится в данном состоянии.
   б. На вкладке **Field Reference** задайте **RefName**, чтобы обозначить поле, на которое хотите наложить ограничение в то время, когда рабочий элемент находится в этом состоянии. Раскрывающийся список с доступными полями находится на вкладке **Fields** типа рабочего элемента.
   в. На вкладке **Rules** щелкните **New**, чтобы создать новое правило, или щелкните **Open**, чтобы редактировать существующее правило. В зависимости от типа правила в диалоговом окне отображаются разные данные.
   г. Щелкните **OK**.

3. Отредактируйте переход:
   а. Щелкните правой кнопкой заголовок перехода и выберите **Open Details**, чтобы открыть диалоговое окно **Workflow Transition**.
      - На вкладке **Transition Detail** определены состояния перехода **From** и **To**. Если переход соответствует исходному состоянию, в котором рабочий элемент данного типа находится сразу после создания, состояние **From** должно быть пустым. Эти поля соответствуют связям на диаграмме последовательности операций. Поля **For** и **Not** могут содержать имя пользователя или группы, которые могут или не могут выполнять переход.
      - На вкладке **Reasons** определено допустимое значение поля **Reason** в момент перехода. Для каждого перехода должна быть задана, по крайней мере, одна причина.
      - На вкладке **Actions** определены действия переходов между состояниями, необходимые для автоматизации переходов рабочих элементов в различных точках последовательности операций. Например, система управления исходным кодом TFS должна поддерживать автоматические переходы рабочих элементов во время возврата изменений.
      - На вкладке **Fields** определены правила, ограничивающие поля во время перехода.
   б. Щелкните **OK**.

Шаг 6 — изменение стандартных рабочих элементов

На данном этапе вы добавляете или удаляете стандартные рабочие элементы, создаваемые вместе с проектом. Эти рабочие элементы создаются по умолчанию и позволяют команде организовать подготовку к проекту, назначив рабочие элементы, которые должны быть завершены до его начала.

1. В Process Template Explorer щелкните **Work Item Tracking**.
2. Перейдите на вкладку **Default Work Items** правой панели.
3. Чтобы создать новый тип рабочего элемента по умолчанию, щелкните **Add** на панели инструментов правой панели.
4. В диалоговом окне **Choose Type** выберите тип рабочего элемента.
5. В открывшемся диалоговом окне заполните соответствующие поля.
6. Щелкните **OK**.

Попробуйте удалить задачу **Setup: Migration of Source Code**, выполнив следующие действия:

1. В Process Template Explorer щелкните **Work Item Tracking**.
2. Перейдите на вкладку **Default Work Items** правой панели.

3. Выберите **Setup: Migration of Source Code**.
4. Щелкните **Remove**.

Задача по умолчанию Setup: Migration of Source Code более не будет создаваться при создании нового проекта на основе данного шаблона.

Шаг 7 — изменение и управление запросами

Измените, добавьте или удалите стандартные запросы, создаваемые одновременно с командным проектом.

1. В Process Template Explorer щелкните **Work Item Tracking**.
2. Перейдите на вкладку **Queries** (запросы) правой панели.
3. Чтобы создать новый запрос, щелкните **Add** на панели инструментов правой панели.
4. В диалоговом окне **Query Reference** введите имя нового запроса.
5. Щелкните **Edit Query Definition**.
6. Используйте вкладки **Fields**, **Sorting** и **Criteria** диалогового окна **Query Edit**, чтобы описать свой запрос.
7. Щелкните **OK**.

Например, создадим запрос, который будет выбирать все ошибки с самым сильным воздействием на безопасность:

1. В Process Template Explorer щелкните **Work Item Tracking**.
2. Перейдите на вкладку **Queries** правой панели.
3. Щелкните **Add**.
4. Присвойте параметру **Name** значение **Bugs with Security Impact**.
5. Щелкните **Edit Query Definition**.
6. В диалоговом окне **Query Edit** перейдите на вкладку **Fields**, выберите все поля для отображения, включая **MyBug.CustomField.1**, и щелкните кнопку со стрелкой, чтобы перенести их в окно **Selected Columns**. Если вы не видите добавленного вами поля, убедитесь, что сохранили файл ProcessTemplate.xml, и повторите попытку.
7. Перейдите на вкладку **Criteria**, оставьте столбец **And Or** пустым, а в столбце **Field** выберите значение **MyBug.CustomField.1**. В столбце **Operator** выделите стрелку, а затем в столбце **Value** введите “**1”**. Значение 1 обязательно нужно взять в кавычки, потому что это поле строкового типа. Если бы при создании поля Security Impact был выбран тип integer или другой числовой тип, кавычки были бы не нужны.
8. Дважды щелкните **OK**.

Этот запрос отобразит все рабочие элементы, у которых значение в поле MyBug.CustomField.1 превышает 1.

Примечание В Process Editor отсутствуют некоторые важные функции по управлению запросами. Их лучше редактировать и тестировать не в шаблоне процесса, а в Visual Studio, в разрабатываемом командном проекте. Запросы можно сохранять в файлы, а затем импортировать их в другие проекты или копировать в шаблоны процессов.

Шаг 8 — изменение областей и итераций

На данном этапе задаются итерации и области по умолчанию, доступные в момент создания командного проекта.

1. В Process Template Explorer щелкните **Areas & Iterations**.
2. Перейдите на вкладку **Areas** или **Iterations** правой панели.
3. С помощью панели инструментов правой панели добавляйте, удаляйте или перемещайте области или итерации.

Например, добавим несколько стандартных областей для организации проектов, создаваемых на основе этого шаблона:

1. В Process Template Explorer щелкните **Areas & Iterations**.
2. Перейдите на вкладку **Areas** правой панели.
3. Щелкните кнопку **New**.
4. В поле имени области введите **UI**.
5. Выберите **Area**, потому что вам нужно, чтобы следующая область была потомком корневой области.
6. Щелкните кнопку **New** еще раз.
7. В текстовом поле имени области введите **Back End**.

Теперь для всех новых проектов, использующих ваш шаблон процесса, автоматически будут определены области UI и Back End.

Шаг 9 — изменение групп и прав доступа

На данном этапе вы изменяете, удаляете или добавляете группы и их права доступа на момент создания командного проекта.

1. В Process Template Explorer щелкните **Groups & Permissions**.
2. Чтобы создать новую группу, щелкните **Add** на панели инструментов правой панели.
3. В диалоговом окне **Group** введите имя новой группы и краткое описание того, что могут делать ее члены. Затем щелкните **OK**.
4. В верхней части правой панели выберите группу и задайте для нее разрешения. Выберите значения **Allow**, **Deny**, или сохраните значение **Unset**. Оно подразумевает отказ в доступе.

Шаг 10 — изменение настроек системы управления исходным кодом

На этом этапе вы измените настройки системы управления исходным кодом для параллельного редактирования и возврата правок.

1. В Process Template Explorer щелкните **Source Control**.
2. Перейдите на вкладку **Checkout Settings** правой панели.
3. Чтобы разрешить одновременное редактирование файла несколькими разработчиками, установите флажок **Enable Multiple Checkout**.
4. Перейдите на вкладку **Checkin Notes** правой панели.
5. Чтобы создать новое поле Checkin Note, щелкните **Add** на панели инструментов окна справа.
6. Чтобы редактировать существующее поле Checkin Note, выберите поле и щелкните **Open**.
7. В диалоговом окне **Checkin Note** внесите необходимые изменения.
8. Щелкните **OK**.

> **Примечание** В этой версии Process Editor недопускается внесение изменений на вкладке **Permissions**.

Например, можно потребовать, чтобы все возвраты изменений сопровождались комментариями, выполнив следующие действия:

1. В Process Template Explorer щелкните **Source Control**.
2. Перейдите на вкладку **Checkin Notes** правой панели.
3. Щелкните дважды каждую строку и установите флажок **Required** в диалоговом окне **Checkin Note**.

Теперь все возвраты правок обязательно должны будут сопровождаться комментариями.

Шаг 11 — изменение портала проекта

На данном этапе вносятся изменения в портал проекта, чтобы отображать на нем определенные документы и руководства по процессу.

1. В Process Template Explorer щелкните **Portal**.
2. Чтобы создать новую библиотеку, щелкните правой кнопкой узел **Portal** в средней панели и выберите команду **New Document Library**.
3. В диалоговом окне **Document Library** введите имя и описание библиотеки.
4. Щелкните **OK**.
5. Щелкните правой кнопкой библиотеку документов и выберите **New Folder**, чтобы создать новую папку.

6. В диалоговом окне **Folder Properties** введите имя новой папки.
7. Выберите папку и затем на панели инструментов правой панели щелкните **Add**, чтобы выгрузить новый документ на сервер.
8. В диалоговом окне **File Import** перейдите к файлу, который хотите выгрузить на сервер. Поля **Destination Folder** и **Share Point Folder** будут заполнены автоматически. Оставьте неизменным значение поля **Query Id**.
9. Щелкните **Import**.
10. Чтобы редактировать свойства существующего документа в библиотеке документов, выберите документ в правой панели и щелкните **Open** на панели инструментов.
11. В диалоговом окне **File Edit** измените имя файла в поле **Name**. Остальные поля оставьте без изменений.
12. Щелкните **OK**.

Шаг 12 — изменение отчетов

На данном этапе вы добавите или удалите отчеты из набора по умолчанию, создаваемого вместе с командным проектом.

1. В Process Template Explorer щелкните **Reports**.
2. На панели инструментов щелкните **Add**.
3. В диалоговом окне **Report** на вкладке **Report Detail** введите имя отчета.
4. Перейдите к файлу .rdl, который хотите добавить в поле **File Name**. Не вносите никаких изменений на вкладках **Properties** и **Parameters**.
5. На вкладке **DataSources** введите соответствующие источники данных. Стандартные источники данных для шаблонов процессов, поставляемые с TFS, — /TfsOlapReportDS и /TfsReportDS.
6. Щелкните **OK**.

Чтобы удалить отчет из шаблона процесса, выполните следующие действия:

1. В Process Template Explorer щелкните **Reports**.
2. Выберите отчет **Load Test Detail** (данные нагрузочного тестирования) и щелкните **Remove**.

Теперь отчет Load Test Detail не будет включаться в проекты, создаваемые с использованием этого шаблона процесса.

Шаг 13 — выгрузка измененного шаблона процесса на сервер

Теперь мы выгрузим измененный шаблон процесса на TFS, после чего он станет доступен для создания новых командных проектов.

1. Щелкните **Save** на панели инструментов Visual Studio, чтобы сохранить файл ProjectTemplate.xml.
2. Щелкните **Team** и выберите **Team Foundation Server Settings**.
3. Щелкните **Process Template Manager**.
4. В диалоговом окне **Process Template Manager** щелкните **Upload**.
5. Перейдите в локальную папку, где находится измененный шаблон процесса.
6. В диалоговом окне **Upload Process Template** щелкните **Upload**. Новый шаблон процесса должен появиться в списке **Process Templates**.
7. Щелкните **Close**.

Процесс настройки завершен. При создании следующего командного проекта этот шаблон будет перечислен среди предлагаемых шаблонов процессов. Чтобы протестировать изменения, описанные в этой статье, выгрузите на сервер созданный вами шаблон процесса My Test:

1. Убедитесь, что сохранили все открытые XML-файлы шаблона процесса.
2. В Visual Studio щелкните **Team** и выберите **Team Foundation Server Settings**.
3. Щелкните **Process Template Manager**.
4. В диалоговом окне **Process Template Manager** щелкните **Upload**.
5. Перейдите в папку на рабочей станции, где вы работали с шаблоном процесса My Test.
6. В диалоговом окне **Upload Process Template** щелкните **Upload**. Новый шаблон процесса должен появиться в списке **Process Templates**.
7. Щелкните **Close**.

Создайте новый проект на основании этого шаблона процесса:

1. В меню **File** выберите **New** и щелкните **Team Project**.
2. Задайте имя (например, Test Project) и щелкните **Next**.
3. В раскрывающемся списке шаблонов выберите **My Test**.
4. Щелкните **Finish**.

Просмотрите измененные вами области:

1. Просмотрите новый тип рабочего элемента.
   a. В меню **Team** выберите **Add Work Item** и щелкните **My Bug**, чтобы создать новый рабочий элемент созданного вами типа.
   б. В блоке **Status** просмотрите поля рабочего элемента. Среди них должно присутствовать добавленное вами поле **Security Impact**.
   в. Убедитесь, что в поле можно вводить только значения в диапазоне от 1 до 5, введите значение **5** и сохраните рабочий элемент.
2. Убедитесь, что удаленного стандартного рабочего элемента больше нет в проекте.

   а. В Team Explorer откройте свой командный проект.
   б. Разверните папку **Work Items**.
   в. Разверните папку **Team Queries**.
   г. Щелкните дважды **All Work Items**.
   д. Просмотрите все предлагаемые стандартные задачи, чтобы убедиться в отсутствии задачи Setup: Migration of Source Code.
3. Проверьте запрос, который вы добавили.
   а. В Team Explorer откройте свой командный проект.
   б. Разверните папку **Work Items**.
   в. Разверните папку **Team Queries**.
   г. Щелкните дважды **Bugs with Security Impact**.
   д. Убедитесь, что созданная вами ошибка отображается правильно.
4. Просмотрите добавленные вами области.
   а. Выберите задачу из запроса **All Work Items**, выполненного в предыдущем шаге.
   б. В раскрывающемся списке **Area** найдите две добавленные вами области (**UI** и **Back End**).
5. Убедитесь, что удаленного вами отчета больше нет в проекте.
   а. В Team Explorer откройте свой командный проект.
   б. Разверните папку **Reports**.
   в. Просмотрите список и убедитесь, что отчета **Load Test Detail** нет.

Дополнительные ресурсы

- Дополнительную информацию о настройке шаблонов процессов вы найдете в статье «Process Template Customization Overview» по адресу *http://msdn2.microsoft.com/en-us/library/ms194945(VS.80).aspx*.
- Скачать Team Foundation Power Tool, включая Process Template Editor, можно по адресу *http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx*.
- Подробнее о настройке шаблона процесса с использованием инструмента Process Editor читайте в руководстве «Process Editor User Guide», устанавливаемом вместе с Process Editor Tool.

Как настроить отчет в Visual Studio 2005 Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).
- Microsoft® SQL Server™ Reporting Services.

Описание

В этой статье подробно разбирается процесс редактирования существующего отчета с последующей публикацией на портале системы отчетов TFS.

Содержание

Задачи

- Создать проект отчетов в Visual Studio.
- Настроить существующий отчет согласно своим нуждам.
- Опубликовать новый отчет на сервере отчетов.

Обзор

Отчеты, поставляемые с VSTS, основаны на использовании SQL Server Reporting Services. С помощью конструктора отчетов Visual Studio (Business Intelligence Development Studio) из комплекта клиентских инструментов SQL Server 2005 можно редактировать готовые отчеты или создавать собственные.

Возможность настраивать отчеты позволяет вводить в них дополнительную функциональность. Если вам нужен отчет, подобный существующему, вам не нужно создавать новый отчет с нуля. Возьмите имеющийся отчет и настройте его, сэкономив таким образом немало времени. Чтобы настроить существующий отчет, его необходимо экспортировать с сервера отчетов, добавить в существующий проект отчетов Visual Studio, а затем после внесения изменений повторно развернуть на портале отчетов.

Порядок операций

- Шаг 1 — создание нового проекта отчетов.
- Шаг 2 — экспорт отчета.
- Шаг 3 — создание источников данных.
- Шаг 4 — добавление отчета в проект.
- Шаг 5 — редактирование отчета.
- Шаг 6 — развертывание отчета на Team Foundation Server.
- Шаг 7 — тестирование отчета.

Прежде всего

Прежде чем приступать к настройке отчета для Team Foundation Server, убедитесь в следующем:

- На компьютере, который будет использоваться для настройки отчета, должна быть установлена среда Business Intelligence Development Studio. Чтобы проверить ее наличие, при создании нового проекта посмотрите, имеется ли в Visual Studio тип Business Intelligence Project.
- Ваша учетная запись должна быть членом роли безопасности Microsoft Analysis Server TfsWarehouseDataReaders на сервере уровня данных.

- Ваша учетная запись должна обладать правами администратора БД TFSWarehouse уровня данных.
- Ваша учетная запись должна быть членом роли Publisher в SQL Server Reporting Services на сервере уровня приложений.

Шаг 1 — создание нового проекта отчетов

Чтобы добавить в проект новый отчет и настроить его, начните с создания проекта отчетов. Выполните следующие действия:

1. В Visual Studio откройте меню **File**, выберите команду **New** и щелкните **Project**.
2. Выберите тип **Business Intelligence Project**.
3. Выберите шаблон **Report Server Project**.
4. Задайте имя и расположение проекта. Затем щелкните **OK**.

Шаг 2 — экспорт отчета

Отчет, который требуется настроить, экспортируется с портала проекта, чтобы затем импортировать его в новый проект отчетов. Выполните следующие действия, чтобы экспортировать отчет:

1. Щелкните правой кнопкой свой командный проект и выберите **Show Project Portal**.
2. На панели **Quick Launch** левой части веб-сайта портала щелкните **Reports**.
3. Выберите отчет, который хотите настраивать.
4. Щелкните **Properties**.
5. Выберите **Edit**.
6. Сохраните файл .rdl отчета в папку проекта отчетов, который был создан в шаге 1.

Шаг 3 — создание источников данных

Чтобы редактировать и публиковать настроенный отчет, необходимо добавить источники данных для хранилища данных Team Foundation Server и OLAP-куб. После добавления этих источников данных в проект Visual Studio отчет может закачивать данные с сервера.

Создание источника данных хранилища

1. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
2. На вкладке **General** введите **TfsReportDS** в текстовое поле **Name**.

3. В списке **Type** выберите **Microsoft SQL Server**.
4. Щелкните **Edit**.
5. Введите имя сервера уровня данных.
6. Выберите базу данных **TFSWarehouse**.
7. Дважды щелкните **ОК**, чтобы добавить источник данных.

Создание источника данных OLAP

1. В окне Visual Studio Solution Explorer щелкните правой кнопкой **Shared Data Sources** и выберите команду **Add New Data Source**.
2. На вкладке **General** введите **TfsOlapReportDS** в поле **Name**.
3. В списке **Type** выберите **Microsoft SQL Server Analysis Services**.
4. Щелкните **Edit**.
5. Введите имя сервера уровня данных.
6. Выберите базу данных **TFSWarehouse**.
7. Дважды щелкните **ОК**, чтобы добавить источник данных.

Шаг 4 — добавление отчета в проект

Теперь, когда в проект добавлены источники данных, можно импортировать отчет, экспортированный на шаге 2:

1. В **Solution Explorer** щелкните правой кнопкой **Reports**, выберите **Add** и затем щелкните **Existing Item**.
2. Перейдите к файлу .rdl, экспортированному на шаге 2.

Шаг 5 — редактирование отчета

Добавив отчет в проект, внесите в него коррективы и настройте соответственно своим нуждам. Чтобы открыть отчет для редактирования, дважды щелкните его в **Solution Explorer**. Теперь его можно менять следующим образом:

- менять операторы запросов в **Data Pane**;
- перетаскивать новые меры или элементы в **Data Pane**;
- менять разметку отчета в **Layout Pane**.

Шаг 6 — развертывание отчета на Team Foundation Server

Внеся изменения в отчет, разверните его на портале отчетов командного проекта:

1. В Solution Explorer щелкните правой кнопкой проект отчетов и выберите **Properties**.

2. Убедитесь, что атрибуту **OverwriteDataSources** присвоено значение **false**.
3. Измените значение **TargetDataSourceFolder** согласно имени своего командного проекта, например:

```
TargetDataSourceFolder = TestProject
```

4. Измените значение **TargetReportFolder** согласно имени своего командного проекта, например:

```
TargetReportFolder = TestProject
```

5. Присвойте параметру **TargetServerURL** значение *http://<имя сервера уровня данных>/reportserver*, например:

```
TargetServerURL = http://tfsrtm/reportserver
```

6. Щелкните **OK.**
7. В Solution Explorer щелкните правой кнопкой файл .rdl и выберите **Deploy**.
8. Посмотрите на **Output Pane**, чтобы убедиться в успешности операции.

Шаг 7 — тестирование отчета

Опубликовав отчет на сервере отчетов своего командного проекта, протестируйте его, чтобы убедиться в успешности развертывания:

1. В **Team Explorer** разверните узел своего командного проекта, щелкните правой кнопкой **Reports** и выберите команду **Show Report Site**.
2. На сайте отчетов выберите созданный отчет.
3. Убедитесь, что он выглядит так, как ожидалось.

Дополнительные ресурсы

- Дополнительную информацию по работе с проектами отчетов вы найдете в статье «Reporting Services Tutorials» по адресу *http://msdn2.microsoft.com/en-us/library/ms170246.aspx*.
- О редактировании отчетов читайте в статье «How to: Edit Reports in Report Designer» по адресу *http://msdn2.microsoft.com/en-us/library/ms244655(VS.80).aspx*.
- О ролях системы безопасности уровня данных читайте в статье «Securing Access Through Analysis Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms174839.aspx*.
- О ролях системы безопасности уровня приложений читайте в статье «Securing Reporting Services» по адресу *http://msdn2.microsoft.com/en-us/library/ms157198.aspx*.

Как управлять проектами в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).

Описание

В этой статье подробно разбирается процесс запуска новых проектов по разработке ПО путем создания и настройки новых командных проектов на основе выбранного шаблона. Также здесь показано, как использовать инструменты TFS для организации, управления и отслеживания процессов разработки ПО.

Содержание

- Шаг 8 — определение требований QoS.
- Шаг 9 — планирование итерации.
- Шаг 10 — отслеживание процесса разработки.
- Рекомендации по работе с областями и итерациями.
- Дополнительные ресурсы.

Задачи

- Научиться создавать командные проекты в TFS.
- Научиться управлять проектами в TFS.

Обзор

В комплект Visual Studio Team System входят инструменты и отчеты, с помощью которых руководители проектов могут единообразно и централизованно управлять всем циклом разработки ПО. Благодаря возможности управления разработкой ПО непосредственно из VSTS и использованию централизованной базы данных, с которой работают все члены команды, улучшается обмен информацией внутри команды и автоматизируется передача рабочих элементов между ее членами. Кроме того, отчеты, создаваемые на основе данных хранилища TFS, намного упрощают задачу по контролю за ходом выполнения проекта.

В этой статье последовательно рассматривается процесс работы с командным проектом, начиная с создания и настройки нового проекта до контроля за его выполнением.

Порядок операций

- Шаг 1 — выбор шаблона процесса.
- Шаг 2 — создание командного проекта.
- Шаг 3 — создание групп доступа (при необходимости).
- Шаг 4 — добавление членов команды в Team Foundation Server.
- Шаг 5 — определение итерационного цикла.
- Шаг 6 — описание сценариев проекта в TFS.
- Шаг 7 — выбор сценариев для итерации.
- Шаг 8 — определение требований QoS.
- Шаг 9 — планирование итерации.
- Шаг 10 — отслеживание процесса разработки.

Прежде всего

Прежде чем приступать к созданию нового командного проекта, необходимо собрать следующую информацию:

- **Процесс, который будет использован в проекте** От выбора процесса зависит выбор шаблона, на основе которого будет создан новый командный проект. Проанализировав процесс, вы также оцените, насколько сильно придется изменять поставляемые стандартные шаблоны. Если вы предпочитаете гибкий стиль выполнения проекта, выбирайте шаблон MSF Agile.
- **Имя проекта** Тщательно продумайте соглашение о присвоении имен командным проектам. Проследите, чтобы имена проектов были достаточно уникальны и чтобы другие пользователи могли без труда найти проект по имени. В некоторых организациях в имя проекта включают его код проекта, в других используют сочетание названия отдела и заголовка проекта.
- **Структура системы управления исходным кодом** Подумайте, начнете ли вы свой проект с пустого дерева исходного кода или возьмете за основу уже существующий исходный код. Разрабатывайте структуру исходного кода на основе требований к коду и ветвлению. Дополнительную информацию по этому вопросу вы найдете в статье «Как структурировать папки управления исходным кодом в Visual Studio Team Foundation Server».

Шаг 1 — выбор шаблона процесса

Продумайте, какой процесс вы хотели бы использовать для своего проекта, а затем рассмотрите возможности каждого из поставляемых с TFS шаблонов. Обращать внимание следует на описания рабочих элементов (есть ли в них все необходимые вам поля?), шаблоны документов, последовательности операций, политики возврата после правки и отчеты.

В TFS включено два шаблона процессов:

- **MSF for Agile Software Development (MSF Agile)** Простой шаблон процесса для небольших, срочных или неформальных проектов по разработке ПО. Он основывается на сценариях и управляемых контекстом действиях. Ориентирован на проект и членов группы.
- **MSF for CMMI® Process Improvement (MSF CMMI)** Шаблон процесса для более серьезных проектов по разработке ПО. Он расширяет функциональность шаблона MSF Agile, предоставляя поддержку аудита, верификации и формальных процессов. Ориентирован на процесс, на соответствие процессу и на организацию.

В случае необходимости стандартные шаблоны можно настраивать, чтобы они максимально соответствовали вашим процессам. Настройке шаблонов процессов посвящен раздел «Как настроить шаблон процесса в Visual Studio Team Foundation Server». В дальнейшем предполагается, что выбран шаблон MSF Agile.

Шаг 2 — создание командного проекта

После выбора шаблона вы готовы создать новый командный проект. Выполните следующие действия:

1. Убедитесь, что Visual Studio соединена с экземпляром TFS.
2. В **Team Explorer** щелкните правой кнопкой мыши сервер и выберите **New Team Project**.
3. На первой странице мастера **New Project Creation Wizard** введите имя командного проекта и щелкните **Next**.
4. На странице **Select a Process Template** из раскрывающегося списка укажите шаблон процесса, выбранный на шаге 1. В нашем примере выберите шаблон **MSF for Agile Software Development —v4.0** и щелкните **Next**.
5. На странице **Specify the Settings for the Project Portal** введите имя и описание портала командного проекта и щелкните **Next**. Указанное здесь имя используется при создании веб-сайта Microsoft Windows SharePoint® Services для портала проекта.
6. На странице **Specify Source Control Settings** задайте **Create an empty source control folder** и щелкните **Next**.
7. На странице **Confirm Team Project Settings** проверьте настройки и щелкните **Finish**.

На сервере TFS будет создан командный проект на базе шаблона процесса MSF Agile.

Шаг 3 — создание групп доступа (при необходимости)

Независимо от выбранного шаблона процесса в новом проекте TFS создается четыре стандартные группы. По умолчанию каждая из этих групп обладает предопределенным набором прав, определяющих, что разрешается делать ее членам. Создаются группы:

- Project Administrator.
- Contributor.
- Reader.
- Build Services.

В своем проекте вы вольны создать и другие группы доступа, чтобы максимально выполнить требования по безопасности конкретной организации. Создание групп доступа — эффективный способ предоставить группе пользователей в командном проекте конкретный набор полномочий. Необходимо предоставлять группе лишь минимально необходимые права доступа и вносить в нее только тех пользователей или другие группы, которым нужны эти права.

Чтобы создать новую группу, вы должны быть членом группы **Project Administrators**. Быть администратором Microsoft Windows® вам не нужно.

1. В Team Explorer выберите командный проект, для которого хотите создать группу.
2. В меню **Team** выберите команду **Team Project Settings** и щелкните **Group Membership**.
3. В диалоговом окне **Project Groups** щелкните **New**.
4. В диалоговом окне **Create New Team Foundation Server Group** в поле **Group name** введите имя группы командного проекта.
5. В поле **Description** введите описание группы.
6. Щелкните **OK** и **Close**.

Создав группу командного проекта, предоставьте ей соответствующие права доступа и добавьте в нее членов группы. По умолчанию вновь созданная группа командного проекта не получает никаких разрешений.

Шаг 4 — добавление членов команды в Team Foundation Server

На этом этапе определяются ресурсы, которые будут работать над проектом, и их роли. Затем члены команды добавляются в TFS. Пользователей можно добавлять в существующие группы проекта или группы уровня сервера. Также вы должны добавить пользователей во вновь созданные вами группы. Для этого вы должны быть членом группы **Team Foundation Administrators**.

1. Выберите нужный командный проект в Team Explorer.
2. В меню **Team** выберите команду **Team Project Settings** и щелкните **Group Membership**. Чтобы добавить пользователей в группу уровня сервера, выберите команду **Team Foundation Server Settings** и щелкните **Group Membership**.
3. В диалоговом окне **Project Groups** выберите группу, в которую хотите добавлять пользователей, и щелкните **Properties**.
4. В диалоговом окне **Team Foundation Server Group Properties** на вкладке **Members** в разделе **Add member** выберите **Windows User or Group**.
5. Щелкните **Add**.
6. В диалоговом окне **Select Users or Groups** в разделе **Enter the object names to select** введите имя домена и имена пользователей, которого хотите добавить, в формате **домен\имя пользователя**. Чтобы добавить сразу несколько пользователей, введите их имена через точку с запятой.
7. Дважды щелкните **OK** и щелкните **Close**.

Шаг 5 — определение итерационного цикла

Итерации (iteration) — это промежутки времени фиксированной длины, для которых вы продумываете, планируете и осуществляете работу. Все компоненты цикла разработки ПО, начиная с определения требований и заканчивая анализом, проектированием, разработкой, написанием кода и тестированием, группируются в итерации, продолжительность которых обычно составляет от 2 до 6 недель.

На этом этапе вы определяете итерационный цикл своего проекта. При этом необходимо принять во внимание следующие соображения:

- Итерационный цикл должен быть достаточно продолжительным, чтобы члены команды имели возможность выполнить существенный объем работ, и должен охватывать, по крайней мере, несколько сценариев.
- Итерационный цикл должен быть достаточно коротким. Это обеспечивает его гибкость, позволяя своевременно вносить изменения и иначе расставлять приоритеты.

Продолжительность итерационного цикла зависит от размера и сложности проекта. На практике для большинства проектов подходит двухнедельный итерационный цикл.

Шаг 6 — описание сценариев проекта в TFS

Описание сценария проекта в TFS позволяет лучше спланировать ход проекта и отслеживать его выполнение.

1. На основании данных, предоставляемых различными заинтересованными сторонами, включая заказчиков, бизнес-аналитиков, конечных пользователей и руководителей, ответственных за выпуск продукта, создайте перечень задач по проекту (project back log, PBL). Его обычно составляют в Microsoft Office Word.
2. Используйте PBL в качестве входных данных для выработки различных сценариев проекта.
3. Можно выделить все сценарии до начала первой итерации или создавать их по мере продвижения проекта. Чтобы получить полную картину проекта и облегчить контроль его выполнения, рекомендуется описывать все сценарии заранее.

Создание сценария в проекте на основе шаблона процесса MSF Agile

1. В **Team Explorer** разверните узел проекта и щелкните правой кнопкой папку **Work Items**.
2. Выберите **Add Work Item** и щелкните **Scenario**.
3. На странице **New Scenario** введите данные сценария.

4. Сохраните новый сценарий.
5. Повторите предыдущие шаги для всех сценариев проекта.

Шаг 7 — выбор сценариев для итерации

Теперь описанные сценарии нужно распределить по итерациям. Этот этап повторяется на каждом итерационном цикле.

1. Исходя из данных, полученных от заинтересованных сторон, и приоритетности компонентов, выберите сценарии, которые будут выполняться в ходе данной итерации.
2. Назначьте эти сценарии итерации.

Извлечение рабочих элементов и их связывание с конкретной итерацией

1. Разверните папки **Work Items** и **Team Queries**, а затем дважды щелкните запрос **All Scenarios**, чтобы вывести на экран все сценарии проекта.
2. Щелкните дважды сценарий, над которым хотите работать в текущей итерации.
3. В поле **Iteration** введите путь текущей итерации и щелкните значок **Save**.
4. Повторите эти шаги для всех выявленных сценариев.

Шаг 8 — определение требований QoS

На этом этапе для каждого из сценариев, реализуемых во время данного итерационного цикла, определяются требования QoS. Этот этап повторяется на каждом итерационном цикле. Это помогает определить критерий приемки для сценария. Требования QoS формулируются на основе целей, требований к проекту и спецификаций, если таковые имеются.

1. Щелкните правой кнопкой папку **Work Items** проекта и выберите **Add Work Item**. Затем щелкните **Quality of Service Requirements**.
2. На странице **New Quality of Service Requirements** введите следующую информацию:
   а. Присвойте полю **Type** нужное значение: Performance, Scalability, Stress или Security.
   б. Назначьте значение **Iteration** текущего итерационного цикла.
   в. На вкладке **Links** свяжите требование QoS с конкретным сценарием, чтобы упростить контроль.
3. Сохраните требование QoS.
4. Создайте по одному требованию QoS для каждой дисциплины или типа требования QoS. Помните, что у каждого сценария может быть несколько требований QoS.

5. Убедитесь, что создали требования QoS для всех сценариев, реализуемых в течение данного итерационного цикла.

> **Важно!** Позже требования QoS могут быть разложены на задачи тестирования.

Шаг 9 — планирование итерации

Планирование итерации заключается в разделении на задачи сценариев, требований QoS и других рабочих элементов, в предварительной оценке времени, необходимого для выполнения каждой задачи, и в распределении задач между членами команды. Этот этап повторяется на каждом итерационном цикле.

Задачи разработки

1. Разбейте выбранные сценарии на уровни разработчиков.
2. Разделите уровни разработчиков на задачи разработки.
3. Зафиксируйте задачи разработки в TFS как рабочие элементы Task:
   а. В Team Explorer в каталоге проекта щелкните правой кнопкой папку **Work Items**, выберите **Add Work Item** и щелкните **Task**.
   б. На странице **New Task** введите следующие данные:
      - Присвойте полю **Discipline** значение **Development**.
      - Задайте в качестве **Iteration** текущий итерационный цикл.
      - На вкладке **Links** свяжите задачу с конкретным сценарием для упрощения контроля.
      - На вкладке **New Task Page** помимо описания задайте критерий приемки задачи, по которому можно будет судить о ее успешном завершении.
      - В поле **Assigned to** задайте разработчика, который будет заниматься данной задачей.
   в. Сохраните новую задачу.
   г. Повторите перечисленные шаги для всех задач.
4. Повторите эти шаги для всех сценариев итерации.

Задачи тестирования

1. Разбейте требования QoS для данного сценария на сценарии тестирования.
2. Разделите сценарии тестирования на задачи тестирования и зафиксируйте их в TFS как рабочие элементы Task:

а. В Team Explorer в каталоге проекта щелкните правой кнопкой папку **Work Items**, выберите **Add Work Item** и щелкните **Task**.

б. На странице **New Task** введите следующие данные:

- Присвойте полю **Discipline** значение **Test**.
- Задайте в качестве **Iteration** текущий итерационный цикл.
- На вкладке **Links** свяжите задачу с конкретными требованиями QoS для упрощения контроля.
- На вкладке **New Task Page** помимо описания задайте критерий приемки задачи, по которому можно будет судить о ее успешном завершении.
- В поле **Assigned to** задайте тестировщика, который будет заниматься данной задачей.

в. Сохраните новую задачу.

г. Повторите перечисленные шаги для всех задач.

4. Повторите эти шаги для всех требований QoS итерации.

Прочее

1. Опишите задачи итерации, для других дисциплин, например, Architecture, Release Management, Project Management, Requirements, которые нуждаются в контроле.
2. Свяжите каждую из этих задач с соответствующим сценарием.

В крупных проектах с огромным количеством рабочих элементов используйте возможность интеграции с Microsoft Office Excel®. В этом случае рабочие элементы создаются в электронной таблице Excel и затем загружаются на TFS. Подробнее об этом читайте в статье «Working with Work Item Lists in Microsoft Excel» по адресу *http://msdn2.microsoft.com/en-us/library/ms181694(VS.80).aspx*.

В крупных проектах, где задействовано множество ресурсов, используйте возможность интеграции с Microsoft Office Project для отслеживания рабочих элементов и равномерного распределения задач между исполнителями. Дополнительную информацию по этому вопросу вы найдете в статье «Working with Work Items in Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms244368(VS.80).aspx*.

Шаг 10 — отслеживание процесса разработки

В каждый шаблон процесса включен набор отчетов, которые можно использовать для контроля за состоянием проекта. Ниже описаны стандартные отчеты.

Ошибки

Отчеты об ошибках позволяют следить за тем, какие типы ошибок возникают, как они устраняются, какие возникают тенденции. К вашим услугам следующие отчеты об ошибках:

- **Bugs by Priority** Правильно ли выявлены ошибки? В этом отчете показывается соотношение выявленных высокоприоритетных ошибок и ошибок с низким приоритетом. Он доступен в обоих стандартных шаблонах.
- **Bug Rates** Насколько эффективно происходит выявление, исправление и закрытие ошибок? В этом отчете показывается общая тенденция по выявлению новых ошибок, приводятся списки незакрытых ошибок и сведения по исправлению ошибок. Он доступен в обоих стандартных шаблонах.

Управление выпусками

Отчеты по управлению выпусками позволяют судить, насколько разрабатываемое ПО готово к выпуску. К вашим услугам следующие отчеты:

- **Actual Quality versus Planned Velocity** Сколько сценариев можно завершить, прежде чем качество станет неприемлемым? На каждой итерации этот отчет представляет соотношение примерного объема проекта и общего качества. Отчет доступен в обоих стандартных шаблонах.
- **Builds** Каково качество сборки? В этом отчете содержится список имеющихся сборок, а также их качество и другая подробная информация. Отчет доступен в шаблоне MSF CMMI.
- **Quality Indicators** Каково качество ПО? В этом отчете собраны результаты тестов, ошибки, сведения о покрытии кода и его изменчивости. Отчет доступен в обоих стандартных шаблонах.
- **Velocity** Насколько быстро команда справляется с работой? Из этого отчета вы узнаете, насколько своевременно команда выполняет плановые задания и как темп ее работы меняется изо дня в день. Отчет доступен в обоих стандартных шаблонах.
- **Scenario Details** Для каких сценариев мы готовим приложение? В отчете содержатся сведения обо всех сценариях, включая информацию о завершенности, рисках и испытаниях.

Тестирование

Отчеты о тестировании позволяют следить за эффективностью испытаний. Доступны следующие отчеты о тестировании:

- **Regressions**Какие тесты ранее выполнялись, а теперь — нет? Их список содержится в этом отчете. Отчет доступен в шаблоне MSF CMMI.
- **Requirements Test History** Насколько хорошо протестированы сценарии и требования? В этом отчете показаны результаты испытаний определенных сценариев и требований. Отчет доступен в шаблоне MSF CMMI.

- **Test Failure Without Active Bug** Каждый ли из известных дефектов документирован как ошибка? В этом отчете показаны неудачные испытания, с которыми не связаны открытые ошибки. Отчет доступен в шаблоне MSF CMMI.
- **Test Passing With Open Bug** Своевременно ли обновляется список ошибок и согласуется ли он с качеством приложения? Отчет отображает список устаревших ошибок, тесты для которых теперь выполняются. Доступен в шаблоне MSF CMMI.
- **Load Test Summary** К каким выводам о производительности приложения привели испытания под нагрузкой? В отчете содержатся результаты нагрузочного тестирования. Отчет доступен в шаблоне MSF Agile.

Рабочие элементы

Отчеты о рабочих элементах позволяют оценивать текущее состояние проекта и его продвижение. Доступны следующие отчеты о рабочих элементах:

- **Open Issues and Blocked Work Items Trend** Сколько у вас осталось неразрешенных проблем? В отчете перечислены открытые проблемы и наметившиеся тенденции к их разрешению. Отчет доступен в шаблоне MSF CMMI.
- **Reactivations** Сколько рабочих элементов было повторно активировано? В отчете указаны рабочие элементы, которые были преждевременно закрыты или помечены как разрешенные. Отчет доступен в обоих стандартных шаблонах.
- **Related Work Items** Как одни рабочие элементы зависят от других рабочих элементов? В отчете отображается список рабочих элементов, которые связана с другими рабочими элементами, что позволяет прослеживать зависимости между ними. Отчет доступен в шаблоне MSF CMMI.
- **Remaining Work** Сколько осталось выполнить работ и когда они будут завершены? В отчете отражена незавершенная работа, а также разрешенная и закрытая работа. Выявив тенденции, вы определите время, к которому код будет завершен. Отчет доступен в обоих стандартных шаблонах.
- **Triage** Какие рабочие элементы нуждаются в уточнении? В этом отчете показаны рабочие элементы, все еще имеющие статус предложения. Отчет доступен в шаблоне MSF CMMI.
- **Unplanned Work** Сколько выполняется внеплановых работ? В отчете полная работа сопоставляется с уже выполненной с разделением плановых и внеплановых задач. Отчет доступен в обоих стандартных шаблонах.
- **Work Items** Какие рабочие элементы активны? В отчете перечислены все активные рабочие элементы. Отчет доступен в шаблоне MSF CMMI.
- **Work Items by Owner** Сколько работы назначено каждому члену команды? В этом отчете рабочие элементы отсортированы по владельцам.

Отчет доступен в шаблоне MSF CMMI.

- **Work Items by State** Сколько имеется активных, разрешенных и закрытых рабочих элементов? В этом отчете рабочие элементы отсортированы по состоянию. Отчет доступен в шаблоне MSF CMMI.

Рекомендации по работе с областями и итерациями

При работе с областями и итерациями необходимо учитывать следующие соображения:

- Области используются для организации работы в командном проекте. Например, работу по проекту можно разбить на области UI (пользовательский интерфейс), Application (приложение) и Database (база данных), а затем распределить по этим областям сценарии и рабочие элементы. С помощью областей рабочие элементы группируются для запросов и отчетов. Можно начать с одной корневой области, а потом в ходе разработки проекта по мере надобности создавать дополнительные области.
- С помощью итераций мы определяем, сколько раз в ходе разработки приложения команда будет повторять определенный набор основных действий (планирование, разработка, тестирование). Итерации используются для группировки рабочих элементов и тем самым оказывают влияние на создание рабочих элементов, запросов к рабочим элементам и отчетов.

Дополнительные ресурсы

- Дополнительную информацию об использовании электронных таблиц Microsoft Office Excel для работы с рабочими элементами вы найдете в статье «Working with Work Item Lists in Microsoft Excel» по адресу *http://msdn2.microsoft.com/en-us/library/ms181694(VS.80).aspx*.
- Дополнительную информацию об использовании Microsoft Office Project для отслеживания рабочих элементов и равномерного распределения задач между исполнителями вы найдете в статье «Working with Work Items in Microsoft Project» по адресу *http://msdn2.microsoft.com/en-us/library/ms244368(VS.80).aspx*.

Как перенести исходный код из Visual Source Safe в Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual SourceSafe® (VSS).

Описание

В этой статье подробно разбирается процесс переноса исходного кода из VSS в TFS с использованием VSS Converter. Конвертер позволяет переносить файлы, папки, историю версий, метки и пользовательскую информацию из базы данных VSS в базу данных системы управления исходным кодом TFS. Прежде чем выполнять перенос исходного кода, необходимо создать резервную копию и подготовить исходную базу данных VSS.

Содержание

Задачи

- Научиться анализировать проекты VSS и выполнять подготовку к переносу.
- Перенос проектов VSS в TFS.

Обзор

В Team Foundation Server включены инструменты, облегчающие перенос исходного кода из базы данных Visual Source Safe (VSS) в систему управления исходным кодом TFS. В частности, в TFS предлагается инструмент VSS Converter, позволяющий переносить файлы, папки, историю версий, метки и пользовательскую информацию. VSS Converter используется в два этапа: сначала для выявления потенциальных проблем с его помощью выполняется анализ существующей базы данных VSS, а затем выполняется фактический перенос.

Следуя рекомендациям из этой статьи, вы успешно перенесете свой исходный код. Основные проблемы, которые могут при этом возникнуть, обусловлены некоторыми различиями процесса управления исходным кодом в TFS по сравнению с VSS. Например, в TFS не поддерживается совместное использование файлов. При переносе общий файл копируется в целевую папку в том состоянии, в котором он был на момент начала совместного использования. Ветвление в VSS осуществляется через совместное использование файлов, поэтому перенос ветвей также заключается в копировании файлов в целевую папку системы управления исходным кодом TFS. Поскольку TFS не поддерживает фиксацию версий, инструмент VSSConverter маркирует все файлы, версии которых ранее были фиксированы в базе данных VSS, меткой «PINNED», чтобы их можно было найти в системе управления исходным кодом TFS.

Прежде всего

Чтобы успешно выполнить все действия, описанные в этой статье, необходимо следующее:

- Вы должны зарегистрироваться под учетной записью, входящей в группу Team Foundation Administrators.
- На компьютере, где работает конвертер, должен быть установлен клиент VSS 2005. Если используется более ранняя версия VSS, работа конвертера завершается с выводом предупреждения. Также обязательно используйте команду **Analyze** клиента VSS 2005, поскольку она способна находить проблемы, которые предыдущими версиями не определялись. База данных VSS необязательно должна быть базой данных именно версии VSS 2005.

- На компьютере, где выполняется конвертер, должен быть установлен и активирован Microsoft SQL Server™ 2005 Express Edition. Во время преобразования конвертер использует локальный экземпляр SQL Server как временную базу данных. SQL Server Express Edition устанавливается по умолчанию с Visual Studio 2005.
- Используйте имя домена TFS и список имен пользователей TFS так, как они определены в Microsoft Active Directory®.
- Заранее узнайте имя пользователя и пароль администратора VSS, а также имя пользователя и пароль администратора проекта TFS.

Порядок операций

- Шаг 1 — создание резервной копии базы данных VSS.
- Шаг 2 — анализ базы данных VSS с точки зрения целостности данных.
- Шаг 3 — анализ проектов в VSS.
- Шаг 4 — подготовка к переносу проектов.
- Шаг 5 — перенос проектов.

Шаг 1 — создание резервной копии базы данных VSS

Прежде чем выполнять перенос, создайте резервную копию базы данных VSS, которую собираетесь переносить.

1. Попросите всех пользователей возвратить редактируемые ими файлы и выйти из базы данных VSS. Попросите пользователей закрыть Visual Studio Integrated Development Environment (IDE) и VSS Explorer.

Важно! Извлеченные для редактирования файлы не будут перенесены в TFS.

---

2. Убедитесь, что никто не подключен к базе данных.
3. Убедитесь, что для базы данных не запланированы никакие задачи анализа.
4. Скопируйте следующие подпапки папки установки VSS в папку резервного копирования:

```
\DATA
\Temp
\USERS
```

5. Скопируйте в папку резервного копирования следующие файлы:

```
User.txt
Srcsafe.ini
```

По умолчанию эти файлы размещаются в папке \Program Files\Microsoft Visual Studio\VSS.

Шаг 2 — анализ базы данных VSS с точки зрения целостности данных

На данном этапе с помощью утилиты Visual SourceSafe Analyze выявляются и исправляются проблемы целостности в базе данных.

1. Откройте окно командной строки и введите команду Analyze.exe, чтобы выявить повреждения или ошибки в базе данных:

   ```
   analyze "<sourcesafe data directory>"
   ```

   Для автоматического выполнения используйте параметр -I.

2. Если у вас возникают проблемы с правами доступа, ошибки «unable to checkout files» (невозможно извлечь файлы), «losing checkout status» (потеря статуса «извлечен для редактирования») или любые другие ошибки со ссылками на файлы Status.dat или Rights.dat, выполните программу Ddconv.exe или Ddconvw.exe. Они обновляют формат базы данных VSS. По умолчанию эти программы установлены в подпапке \Admin.

Шаг 3 — анализ проектов в VSS

Теперь выберите проекты, которые будете переносить, и запустите инструмент командной строки TFS VSSConverter.exe, чтобы выявить в базе данных VSS потенциальные проблемы, способные вызвать трудности при миграции кода.

1. Создайте XML-файл настроек, назвав его, например, ConversionSettings.xml:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <SourceControlConverter>
      <ConverterSpecificSetting>
         <Source name="VSS">
            <VSSDatabase name="c:\VSSDatabase">
            </VSSDatabase>
         </Source>
         <ProjectMap>
            <Project Source="$/MyFirstProject"></Project>
            <Project Source="$/MySecondProject"></Project>
         </ProjectMap>
      </ConverterSpecificSetting>
   </SourceControlConverter>
   ```

В этом примере **MyFirstProject** и **MySecondProject** — имена папок проектов в VSS, которые подлежат переносу. Чтобы перенести всю базу данных VSS, используйте **<Project Source="$/"></Project>**.

2. Чтобы проанализировать проекты, запустите VSSConverter.exe с параметром **Analyze**:

```
VSSConverter Analyze ConversionSettings.xml
```

   По запросу введите пароль администратора Visual SourceSafe.

3. Конвертер создает файл VSSAnalysisReport.xml с результатами анализа. Изучите полученный отчет и выявите возможные ошибки преобразования.
4. Сопоставьте пользователей VSS с пользователями TFS. Инструмент VSSConverter создает файл UserMap.xml, содержащий список всех пользователей VSS, которые хотя бы один раз обращались к базе данных VSS. Отредактируйте файл UserMap.xml, добавляя соответствующие имена пользователей TFS (учетные записи Windows). Имена пользователей должны быть заданы с включением домена (Домен\ИмяПользователя). Ниже показан пример файла Usermap.xml. В атрибуте **From** содержатся учетные записи пользователей VSS, а в атрибуте **To** указаны соответствующие им имена пользователей TFS.

```
<?xml version="1.0" encoding="utf-8" ?>
<UserMappings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <!--
      Этот файл автоматически создается VSS Converter. Он может использоваться
      для сопоставления пользователя VSS с пользователем Team
Foundation. Например,
      <UserMap From="Jane" To="MyDomain\Janep"></UserMap>
      Это сопоставление приводит к тому, что все действия VSS-пользователя “Jane”
      при переносе будут отнесены к пользователю Team Foundation
“MyDomain\Janep”.
   -->
   <UserMap From="ADMIN" To="Contoso\Administrator" />
   <UserMap From="Dave" To="Contoso\DaveM" />
   <UserMap From="Chris" To="Contoso\ChrisP" />
   <UserMap From="John" To="Contoso\JohnR" />
</UserMappings>
```

Шаг 4 — подготовка к переносу проектов

На данном этапе с помощью инструмента VSSConverter.exe выполняется перенос проектов VSS.

1. Измените файл ConversionSettings.xml, созданный на шаге 3, добавив в него новый элемент **<Settings>** с тегом **<TeamFoundationServer>**:

```
<SourceControlConverter>
   <ConverterSpecificSetting>
   …
   </ConverterSpecificSetting>
   <Settings>
      <TeamFoundationServer name="YourTFSServerName"
                            port="PortNumber"
                            protocol="http">
      </TeamFoundationServer>
   </Settings>
   …
</SourceControlConverter>
```

   Вставьте элемент **<Settings>** сразу после **</ConverterSpecificSettings>**, как дочерний элемент элемента **<SourceControlConverter>**.

2. Измените файл ConversionSettings.xml, добавляя атрибуты **Destination** в элементы **<Project>**, как показано ниже. В качестве значений атрибутов **Destination** задавайте пути к папкам проекта TFS Team, куда вы хотите перенести файлы.

```
<?xml version="1.0" encoding="utf-8"?>
<SourceControlConverter>
   <ConverterSpecificSetting>
      <Source name="VSS">
         <VSSDatabase name="c:\VSSDatabase">
         </VSSDatabase>
      </Source>
      <ProjectMap>
         <Project Source="$/MyFirstProject"
                  Destination="$/MyTeam_ProjectOne">
         </Project>
         <Project Source="$/MySecondProject"
                  Destination="$/MyTeam_ProjectTwo">
         </Project>
      </ProjectMap>
   </ConverterSpecificSetting>
   <Settings>
         <TeamFoundationServer name="YourTFSServerName"
                               port="PortNumber"
                               protocol="http">
         </TeamFoundationServer>
   </Settings>
</SourceControlConverter>
```

В разделе **<ProjectMap>** для каждой переносимой папки VSS вместо **MyFirstProject** укажите исходные папки VSS и вместо **MyTeam_Project-One** — заданные папки системы управления исходным кодом TFS. Включите запись **<Project>** для всех проектов, которые хотите переносить.

Шаг 5 — перенос проектов

Скопируйте базу данных VSS в локальную папку (например, C:\VSSDatabases) на компьютере, на котором хотите выполнить анализ и перенос. Базу данных VSS можно перенести и в совместно используемую папку на удаленном компьютере, но миграция в этом случае займет намного больше времени.

1. В командном окне введите:

```
VSSConverter Migrate Conversionsettings.xml
```

2. Введите **Y**, чтобы подтвердить перенос. По запросу введите пароль пользователя с правами администратора Visual SourceSafe.

Дополнительные рекомендации

Если в VSS применялись совместное использование файлов, ветвление или фиксация версий файлов, необходимо учитывать следующие соображения:

- **Совместное использование** Team Foundation Server не поддерживает совместного использования файлов. Перенос таких файлов осуществляется путем копирования версии файла на момент начала его совместного использования. Все последующие изменения, вносимые в совместно используемый файл, реплицируются как в общую, так и в исходную копию.
- **Ветвление** Поскольку ветвление в VSS осуществляется посредством совместного использования файлов, перенос ветвей приводит к тому, что файлы копируются в заданную папку системы управления исходным кодом TFS. После ветвления изменения, вносимые в любую из ветвей, переносятся в соответствующую копию в системе управления исходным кодом TFS.
- **Фиксация версии** Team Foundation Server не поддерживает фиксацию версий файлов. Чтобы найти в системе управления исходным кодом TFS элементы, версии которых ранее были фиксированы в базе данных VSS, инструмент VSSConverter маркирует такие файлы меткой «PINNED».

Для оптимизации производительности (это особенно важно для больших БД VSS) выполняйте преобразования на сервере TFS.

Дополнительные ресурсы

- Дополнительную информацию о подготовке к переносу вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms181246(en-us,vs.80).aspx*.

- О том, как выполнять перенос, читайте по адресу *http://msdn2.microsoft.com/en-us/library/ms181247(VS.80).aspx*.
- Дополнительную информацию об ограничениях конвертера Visual SourceSafe вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms252491(VS.80).aspx*.
- Инструмент VSS Analyze подробно описан по адресу *http://msdn2.microsoft.com/en-us/library/ysxsfw4x.aspx*.
- Команда migrate инструмента VSSConverter подробно рассматривается в статье по адресу *http://msdn2.microsoft.com/en-us/library/ms400685 (VS.80). aspx*.

Как выполнить слияние без основы в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).

Описание

В этой статье подробно разбирается процесс слияния двух ветвей, которые никак не связаны друг с другом. Процесс слияния элементов, не являющихся непосредственными ответвлениями друг друга, называется *слиянием без общей основы* (baseless nerge). Выполняется такое слияние с помощью команды **Tf merge**. Осуществить слияния без основы из интерфейса Visual Studio невозможно.

Содержание

- Задачи.
- Обзор.
- Порядок операций.
- Шаг 1 — оценка необходимости выполнения слияния без основы.
- Шаг 2 — выполнение слияния без основы с использованием Tf.exe.
- Шаг 3 — разрешение конфликтов, возникших при слиянии.
- Шаг 4 — возврат изменений, внесенных в результате слияния, в систему управления исходным кодом.
- Дополнительные ресурсы.

Задачи

- Научиться анализировать необходимость в слиянии без основы.
- Выполнить слияние без основы и разрешить возникшие в результате него конфликты.

Обзор

Процесс слияния элементов, не являющихся непосредственными ответвлениями друг друга, называется *слиянием без основы*. Примером такого слияния может быть перенос изменений между двумя ветвями выпускаемых версий, являющимися ветвями одного уровня, без переноса изменений в родительскую ветвь. Слияние без основы может выполняться только посредством команды **Tf merge**. Его невозможно выполнить из интерфейса Visual Studio.

При слиянии без основы TFS не располагает никакой информацией об отношениях между файлами ветвей. Например, если файл был переименован, это будет рассматриваться как удаление файла и добавление в ветвь нового файла. Поэтому при таком слиянии приходится разрешать больше конфликтов, чем при обычном слиянии. Однако это делается только один раз. После первого слияния без основы TFS сохраняет историю слияния и устанавливает отношения между папками и файлами.

Порядок операций

- Шаг 1 — оценка необходимости выполнения слияния без основы.
- Шаг 2 — выполнение слияния без основы с использованием Tf.exe.
- Шаг 3 — разрешение конфликтов, возникших при слиянии.
- Шаг 4 — возврат изменений, внесенных в результате слияния, в систему управления исходным кодом.

Шаг 1 — оценка необходимости выполнения слияния без основы

На данном этапе рассматриваются ветви и элементы, подлежащие слиянию, и принимается решение о том, следует ли выполнять слияние без основы или можно обойтись обычным слиянием.

Если вы являетесь владельцем или администратором командного проекта, отношения между ветвями или элементами вам известны. Из Visual Studio можно выполнять слияние только тех ветвей, между которыми установлено отношение «родитель-потомок». Если в проекте имеются ветви или элементы, между которыми нет таких отношений, придется выполнять слияние без основы.

Если вам не известны отношения между ветвями или элементами, необходимость в выполнении слияния без основы можно установить следующим образом:

1. Откройте Source Code Explorer.
2. Щелкните правой кнопкой мыши папку ветви и щелкните **Merge**.
3. В диалоговом окне Source Control Merge Wizard щелкните раскрывающийся список **Target branch**. Отсутствие в этом списке ветви, для которой вы хотите выполнять слияние, указывает, что между данными ветвями нет отношения слияния. В этом случае необходимо выполнять слияние без основы.

Шаг 2 — выполнение слияния без основы с использованием Tf.exe

Чтобы выполнить слияние без основы с помощью инструмента командной строки Tf.exe, выполните следующие действия:

1. Подготовьте рабочую область, выполнив операцию **Get Latest** для подлежащих слиянию ветвей:
   a. Откройте Source Code Explorer.
   б. Щелкните правой кнопкой мыши папку первой ветви, участвующей в слиянии, и выберите команду **Get Latest Version**.
   в. Повторите это действие для второй ветви, участвующей в слиянии.

      Если сопоставление рабочей области не задано, Visual Studio предложит выбрать папку на локальном диске.
2. Откройте окно командной строки Visual Studio.
3. Введите в окне командной строки следующую команду:

```
Tf merge /baseless <исходный путь> <целевой путь> /recursive
```

   Например:

```
Tf merge /baseless c:\data\proj1 c:\data proj2 /recursive
```

   Если необходимо выполнить слияние конкретных версий изменений, используйте параметр /version:

```
tf merge /baseless <исходный путь> <целевой путь> / recursive /
version:<набор изменений, из которого переносятся изменения>~<набор изменений, в который переносятся изменения>
```

   Например:

```
tf merge /baseless c:\data\proj1 c:\data\proj2 /recursive /version:
C123~C125
```

Шаг 3 — разрешение конфликтов, возникших при слиянии

При слиянии без основы часто возникают конфликты. После выполнения команды Tf.exe на экран выводится диалоговое окно **Resolve Conflicts** со списком файлов, в которых возникли конфликты.

1. Выберите все файлы и щелкните **Resolve**.
2. В диалоговом окне **Resolve version conflict** выполните следующие действия:
   a. Если содержимое файлов не изменялось, выберите параметр **Keep changes in the target branch** и щелкните **OK**.
   б. Если содержимое файлов изменялось, выберите параметр **Merge changes in merge tool** и щелкните **OK**.
3. В инструменте слияния выбирайте конфликтные области в верхнем окне или вводите изменения в нижнем окне, обрабатывая таким образом каждую из строк, в которых возникли конфликты.
4. Разрешив все конфликты, щелкните **OK** в инструменте слияния.
5. Щелкните **Close**.

Шаг 4 — возврат изменений, внесенных в результате слияния, в систему управления исходным кодом

На этом этапе изменения, внесенные в результате слияния без основы, возвращаются в систему управления исходным кодом.

1. Откройте Source Code Explorer.
2. Щелкните правой кнопкой целевую папку, в которую были перенесены изменения, и выберите команду **Check-in pending changes**.
3. В диалоговом окне **Check-In Source Files** выделите все файлы, которые подлежат возврату.
4. Щелкните **Check In**.

Дополнительные ресурсы

- Подробнее о слиянии папок читайте в статье «How to: Merge Files and Folders» по адресу *http://msdn2.microsoft.com/en-us/library/ms181428(VS.80).aspx*.
- Дополнительную информацию о слиянии без основы вы найдете в разделе «Merge Command» статьи по адресу *http://msdn2.microsoft.com/en-us/library/bd6dxhfy(VS.80).aspx*.

Как настроить непрерывную интеграцию в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).

Описание

В Visual Studio 2005 Team Foundation Server нет стандартного решения для непрерывной интеграции (Continuous Integration, CI), но имеется вся инфраструктура, необходимая для реализации собственного решения. В этой статье подробно разбирается процесс настройки сборки CI в TFS с помощью решения, предоставленного группой VSTS. Это решение устанавливает веб-службу, выполняющуюся от имени учетной записи с правом доступа к серверу TFS. Благодаря подписке на событие **CheckinEvent** эта веб-служба будет запускать сборку при каждом возврате правок.

Примечание Пользователи TFS 2008 могут настроить процесс непрерывной интеграции прямо из Visual Studio. Щелкните правой кнопкой описание типа сборки в узле Builds дерева Team Explorer, выберите команду **Edit Build Definition**, щелкните **Trigger** и активируйте сборку при возврате правок.

Содержание

- Шаг 1 — создание и тестирование сборки.
- Шаг 2 — установка решения непрерывной интеграции.
- Шаг 3 — настройка решения непрерывной интеграции.
- Шаг 4 — подписка на событие CheckinEvent.
- Шаг 5 — тестирование непрерывной интеграции.
- Шаг 6 — настройка уведомлений по электронной почте.
- Дополнительные ресурсы.

Задачи

- Познакомиться с непрерывной интеграцией.
- Создать сборку непрерывной интеграции при помощи TFSBuild и решения группы VSTS.

Обзор

Для повышения качества кода разработчикам необходима постоянная обратная связь по качеству каждого возвращаемого ими изменения. Это особенно важно, когда изменения приводят к сбою сборки и ошибкам компиляции. Чем раньше разработчик получит отзыв, тем быстрее он исправит ошибки и устранит препятствие в работе других разработчиков и тестировщиков. В результате повысится производительность всех членов команды.

В такой ситуации особую ценность приобретает непрерывная интеграция — процесс создания сборок при каждом возврате изменений в систему управления исходным кодом. Сборка непрерывной интеграцией обеспечивает максимально быструю обратную связь. Для еще большей эффективности работы в сборки CI можно ввести пороги качества.

Порядок операций

- Шаг 1 — создание и тестирование сборки.
- Шаг 2 — установка решения непрерывной интеграции.
- Шаг 3 — настройка решения непрерывной интеграции.
- Шаг 4 — подписка на событие CheckinEvent.
- Шаг 5 — тестирование непрерывной интеграции.
- Шаг 6 — настройка уведомлений по электронной почте.

Прежде всего

Убедитесь, что учетная запись, от имени которой выполняется служба сборки, обладает разрешением Start a build на сервере Team Foundation Server.

Шаг 1 — создание и тестирование сборки

На начальном этапе мы создадим тестовую сборку и проверим возможность ее выполнения из Visual Studio. Если сборку выполнить не удается, перед переходом к следующему этапу необходимо исправить в ней ошибки.

1. Создайте для тестирования сценария сборки проект Microsoft Windows® Forms.
2. Убедитесь, что сборка проекта выполняется правильно.
3. Верните проект в систему управления исходным кодом.
4. Создайте сценарий командной сборки:
   a. В Team Explorer щелкните правой кнопкой **Team Builds** и выберите **New Team Build Type**.
   б. Заполните страницы мастера **Team Build Type Creation Wizard**.
5. Проверьте работоспособность сценария командной сборки:
   a. В Team Explorer щелкните правой кнопкой созданный тип сценария командной сборки.
   б. В контекстном меню выберите команду **Build Team Project** <Имя вашего сценария сборки>.
   в. Убедитесь, что тип выбран правильно, и щелкните **Build**.
6. Просмотрите результаты сборки и убедитесь, что в процессе сборки не возникло ошибок.

Важно! Убедитесь, что учетная запись TFSService, от имени которой выполняются службы сборки, обладает полным доступом к общей папке для размещения результатов сборки, заданной в **Team Build Type Wizard**.

Шаг 2 — установка решения непрерывной интеграции

Установите решение непрерывной интеграции, предоставленное командой VSTS. Более подробную информацию об этом решении вы найдете по адресу *http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx*.

1. Скачайте решение из источника, расположенного по адресу *http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482 ci.msi* и установите его на своем сервере TFS.
2. На второй странице мастера установки убедитесь, что выбран сайт «**Team Foundation Server**».

При установке в корневой папке сервера уровня приложений TFS создается виртуальный каталог. Это должен быть веб-сайт Internet Information Services (IIS) с именем Team Foundation Server, связанный с портом 8080.

Шаг 3 — настройка решения непрерывной интеграции

На данном этапе выполняется настройка CI: задается проект, сборка которого будет выполняться непрерывно, сервер сборки и используемый тип сборки.

Чтобы приложение, содержащееся в виртуальном корневом каталоге, выполнялось в пуле приложений TfsAppPool, выполните следующие действия:

1. Раскройте меню **Start**, **Administrative Tools** и выберите команду **IIS Manager**.
2. Разверните веб-сайт **Team Foundation Server**.
3. Щелкните правой кнопкой **CI Web Application** и убедитесь, что для параметра **Application Pool** задано значение **TfsAppPool**.

Чтобы настроить процесс непрерывной интеграции, выполните следующие действия:

1. Откройте файл Web.config для CI Web Application, размещенный в папке C:\Program Files\Microsoft Visual Studio 2005 Team Foundation Server\Web Services\CI\.
2. Задайте следующие свойства в новом разделе, разместив его под разделом **<appsettings>**:
   - **TeamFoundationServer** — URL уровня приложений, задается в формате *http://компьютер:8080*.
   - **TeamProject** — командный проект, для которого активируется непрерывная интеграция.
   - **BuildType** — тип сборки, который должен использоваться при непрерывной интеграции. Обычно сценарий включает только саму сборку, но сюда можно добавить также базовое тестирование и статический анализ.
   - **Build Machine** — необязательный параметр, используется для переопределения сервера сборки, заданного в типе сценария сборки по умолчанию.

   Ниже приведен пример настроек:

```
…
<add key="1" value="TeamServer=http://TFSRTM:8080;TeamProjectName=Adventure
Works;BuildType=Test Build"/>
…
```

Шаг 4 — подписка на событие CheckinEvent

Подпишитесь на событие **CheckinEvent**, используя инструмент **bisubscribe**, поставляемый с TFS.

1. Откройте окно командной строки и перейдите в папку C:\Program Files\Microsoft Visual Studio 2005 Team Foundation Server\TF Setup\.
2. Выполните команду:

```
Bissubscribe /eventType CheckinEvent /address http://TFSRTM:8080/
ci/notify.asmx /deliveryType Soap /domain http://TFSRTM:8080
```

3. Чтобы убедиться, что подписка выполнена, сделайте следующее:
   а. Откройте Microsoft SQL Server™ Management Studio.
   б. Откройте базу данных **tfsIntegration**.
   в. Откройте таблицу **tbl_subscription**.

В этой таблице содержатся записи для всех событий, на которые вы подписаны. Там должна быть и запись о том, что решение CI подписано на событие CheckinEvent. В случае необходимости вы вольны отказаться от подписки на любое из зарегистрированных событий, удалив соответствующую запись из таблицы.

Шаг 5 — тестирование непрерывной интеграции

Теперь нужно проверить правильность выполнения CI-сборки.

1. Откройте созданное на шаге 1 приложение Windows Forms.
2. Внесите в код какие-нибудь незначительные изменения, которые гарантированно не вызовут сбой сборки.
3. Возвратите внесенные изменения в систему управления исходным кодом.
4. Если у вас есть доступ к компьютеру, на котором выполняется сборка, откройте Диспетчер задач (Task Manager) и проверьте степень загрузки центрального процессора этого компьютера. Она должна возрастать после возврата изменений, что свидетельствует о выполнении сборки. В списке процессов должно быть видно, что центральный процессор используется процессами msbuild, csc и (или) aspnet_compiler.
5. Дайте сборке завершиться, а затем в **Team Explorer** щелкните дважды **All Build Types** и найдите свою сборку.

Устранение неисправностей

Если сборки нет в списке, выполните следующие действия:

1. Убедитесь, что правильно подписались на событие. Подробнее об этом — в пункте 3 предыдущего шага.
2. Убедитесь, что правильно настроили файл Web.config веб-приложения CI. Подробнее — в шаге 3.

3. Убедитесь, что веб-приложение CI выполняется в соответствующем контексте и имеет доступ к серверу TFS.
4. Если все сделано правильно, но сборка по-прежнему дает сбой, проведите отладку веб-приложения:
   а. Создайте новый проект веб-сайта.
   б. Добавьте в проект существующий веб-сайт.
   в. Щелкните кнопку **Local IIS** и выберите из списка веб-приложение CI.
   г. Откройте **notify.cs** и создайте точку останова в методе **Notify**.
   д. Нажмите F5, чтобы начать отладку.
   е. Внесите изменения в тестовое приложение Windows Forms и верните код в систему управления исходным кодом.
      - Если в ходе выполнения точка останова не достигнута, событие не детектируется. Попробуйте подписаться на него еще раз.
      - Если в ходе выполнения приложение останавливается в точке останова, продолжайте отладку.
      - Убедитесь, что в Microsoft Internet Explorer включена отладка сценариев. В меню **Сервис (Tools)** выберите **Свойства обозревателя (Internet Options)** и перейдите на вкладку **Дополнительно (Advanced)**. Убедитесь, что флажок **Отключить отладку сценариев (Internet Explorer) (Disable Script Debugging (Internet Explorer))** сброшен.

Шаг 6 — настройка уведомлений по электронной почте

Можно настроить оповещение всех заинтересованных сторон о завершении сборки по электронной почте.

1. В Team Explorer щелкните правой кнопкой соответствующий командный проект.
2. Выберите **Project Alerts**.
3. Выберите параметр **A build completes** и введите адрес или адреса электронной почты для рассылки уведомлений.

Дополнительные ресурсы

- Дополнительную информацию об использовании решения Visual Studio Team System для поддержки непрерывной интеграции вы найдете в статье «Continuous Integration Using Team Foundation Build» по адресу *http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx*.
- Скачать программу установки решения Visual Studio Team System для поддержки непрерывной интеграции можно по адресу *http://download.*

microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi.

- Подробнее о гибкой разработке и непрерывной интеграции в Team Foundation Server читайте в статье «Extend Team Foundation Server To Enable Continuous Integration» по адресу *http://msdn.microsoft.com/msdnmag/issues/06/03/TeamSystem/default.aspx*.

Как настроить плановую сборку в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).

Описание

В Visual Studio 2005 Team Foundation Server не предусмотрена возможность плановых сборок, но с помощью команды **TFSBuild** разработчик легко реализует их самостоятельно. В этой статье подробно разбирается процесс настройки плановой сборки с помощью команды **TFSBuild** и планировщика задач Microsoft Windows®.

Примечание Пользователи TFS 2008 могут настраивать плановые сборки прямо из Visual Studio. Щелкните правой кнопкой мыши описание типа сборки в узле Builds дерева Team Explorer, выберите **Edit Build Definition**, щелкните **Trigger** и задайте график выполнения сборки.

Содержание

- Шаг 4 — создание пакетного файла.
- Шаг 5 — тестирование пакетного файла.
- Шаг 6 — добавление плановой задачи.
- Шаг 7 — тестирование плановой задачи.
- Дополнительные ресурсы.

Задачи

- Изучить, что такое плановая сборка.
- Создать плановую сборку с помощью утилиты командной строки **TFSBuild** и планировщика задач Windows.

Обзор

Чрезвычайно важно, чтобы команда разработки проекта регулярно формировала сборки. Это обеспечит ей своевременное получение обратной связи от тестировщиков и других заинтересованных сторон. Такое взаимодействие можно обеспечить за счет плановых сборок. Можно спланировать выполнение сборок каждую ночь, неделю, раз в две недели или по любому другому графику в зависимости от масштабов проекта и предъявляемых требований.

Плановые командные сборки в TFS не поддерживаются, но эту проблему решает утилита **TFSBuild**, которая позволяет запускать сценарии сборки из командной строки. Используя любой планировщик задач, например, Windows Task Scheduler, можно запускать утилиту **TFSBuild** по определенному графику, создавая сборки через заданные интервалы времени.

Порядок операций

- Шаг 1 — создание и тестирование сборки.
- Шаг 2 — создание команды для запуска TFSBuild.
- Шаг 3 — тестирование команды для запуска TFSBuild.
- Шаг 4 — создание пакетного файла.
- Шаг 5 — тестирование пакетного файла.
- Шаг 6 — добавление плановой задачи.
- Шаг 7 — тестирование плановой задачи.

Прежде всего

Убедитесь, что учетная запись, от имени которой выполняется служба сборки, обладает разрешением **Start a build** на сервере Team Foundation Server.

Шаг 1 — создание и тестирование сборки

На начальном этапе мы создадим тестовую сборку и проверим возможность ее выполнения из Visual Studio. Если сборку выполнить не удается, перед переходом к следующему этапу необходимо исправить в ней ошибки.

1. Создайте для тестирования сценария сборки проект Microsoft Windows® Forms.
2. Убедитесь, что сборка проекта выполняется правильно.
3. Верните проект в систему управления исходным кодом.
4. Создайте сценарий командной сборки:
   a. В Team Explorer щелкните правой кнопкой **Team Builds** и выберите **New Team Build Type**.
   б. Заполните страницы мастера **Team Build Type Creation Wizard**.
5. Проверьте работоспособность сценария командной сборки:
   a. В Team Explorer щелкните правой кнопкой созданный тип сценария командной сборки.
   б. В контекстном меню выберите команду **Build Team Project** <Имя вашего сценария сборки>.
   в. Убедитесь, что тип выбран правильно, и щелкните **Build**.
6. Просмотрите результаты сборки и убедитесь, что в процессе сборки не возникло ошибок.

Важно! Убедитесь, что учетная запись TFSService, от имени которой выполняются службы сборки, обладает полным доступом к общей папке для размещения результатов сборки, заданной в **Team Build Type Wizard**.

Шаг 2 — создание команды для запуска TFSBuild

Сформируем команду для утилиты **TFSBuild**, по которой будет запускаться сборка.

1. Чтобы начать сборку, в утилиту командной строки TFSBuild необходимо передать ряд параметров:
   - **Team Foundation Server** — URL сервера TFS, куда возвращаются изменения в собираемых решениях.
   - **Team Project** — имя командного проекта, сборка которого выполняется.
   - **Build Type** — тип сборки, созданный на шаге 1.
   - **Build Machine** — имя сервера, который будет использоваться для сборки проекта. Это необязательный параметр; по умолчанию использует-

ся сервер, заданный в типе сборки.

- **Build Directory** — путь к папке, в которой выполняется сборка. Это необязательный параметр; по умолчанию используется путь, заданный в типе сборки.

2. Создайте следующую команду для выполнения сборки:

```
TfsBuild start <TFS-сервер> <КомандныйПроект> <ИмяТипаСборки>
```

Если требуется переопределить имя компьютера и путь к папке сборки, команда выглядит так:

```
TfsBuild start <TFS-сервер> <КомандныйПроект> <ИмяТипаСборки>
/m:<ИмяКомпьютера> /d:<ПапкаСборки>
```

Шаг 3 — тестирование команды для запуска TFSBuild

Проверим правильность выполнения команды TFSBuild.

1. Откройте окно командной строки Visual Studio.
2. Введите команду, созданную на шаге 2.
3. Просмотрите полученный результат, чтобы убедиться в отсутствии ошибок и успешности выполнения сборки.

Шаг 4 — создание пакетного файла

Для планирования сборок удобно использовать пакетный файл.

1. Откройте Блокнот (Notepad) и введите следующую команду:

```
"C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\TFSBuild" start
<TFS-сервер> <КомандныйПроект> <ИмяТипаСборки>
```

Обратите внимание, что здесь задан полный путь к файлу TFSBuild.exe, чтобы его можно было запускать из окна командной строки Windows. Если требуется переопределить компьютер и папку сборки, команда в пакетном файле будет выглядеть так:

```
"C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\TFSBuild" start
<КомандныйПроект> <ИмяТипаСборки> /m:<ИмяКомпьютера> /d:<ПапкаСборки>
```

2. Сохраните файл с расширением .bat, например, batchbuild.bat.
3. Поместите файл в папку сборки в системе управления исходным кодом TFS, например, **Main\Scripts**.

Шаг 5 — тестирование пакетного файла

Проверим работу пакетного файла.

1. Откройте окно командной строки Windows.

> **Примечание** Не открывайте окно командной строки Visual Studio. По умолчанию планировщик Windows будет выполнять пакетный файл в окне командной строки Windows.

2. Введите имя пакетного файла.
3. Убедитесь, что сборка выполняется без ошибок.

Шаг 6 — добавление плановой задачи

На данном этапе вы добавите задачу для регулярного запуска сборки.

1. Откройте панель управления.
2. Щелкните дважды значок **Назначенные задания (Scheduled Tasks)**, затем щелкните **Добавить задание (Add Scheduled Tasks)**.
3. На первой странице **Мастера планирования заданий (Scheduled Task Wizard)** щелкните **Далее (Next)**.
4. Щелкните **Обзор (Browse)** и выберите пакетный файл, созданный на шаге 4. Затем щелкните **Далее (Next)**.
5. Введите имя задачи и выберите частоту сборки, например, **Ежедневно (Daily)**. Затем щелкните **Далее (Next)**.
6. Задайте **Время начала** (**Start time**) и **Дату начала (Start date)**. Щелкните **Далее (Next)**.
7. ведите имя и пароль учетной записи с разрешением **Start a build** и щелкните **Далее (Next)**.
8. Щелкните **Готово (Finish)**.

Шаг 7 — тестирование плановой задачи

Убедитесь, что сборка выполняется правильно и в заданное время.

1. Дождитесь момента, когда должна быть выполнена плановая задача, или откройте панель управления, дважды щелкните значок **Назначенные задания (Scheduled Tasks)**, щелкните правой кнопкой свою плановую задачу и выберите команду **Запустить (Run)**.
2. Откроется окно командной строки, и начнется выполнение сборки.
3. Если вы в момент выполнения задачи не имеете доступа к компьютеру, на котором выполняется сборка, ее результаты можно проверить позже:
   a. В Team Explorer щелкните дважды **All Build Types**.

б. Просмотрите список сборок и проверьте, соответствует ли время их выполнения заданному графику.

Дополнительные ресурсы

- Подробно о настройке плановой сборки читайте в статье «How to: Configure a Scheduled Build (Command Line)» по адресу *http://msdn2.microsoft.com/en-us/library/ms181727(VS.80).aspx*.

Как структурировать приложения ASP.NET в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).
- ASP.NET applications.

Описание

В этой статье подробно разбирается процесс организации и структурирования веб-приложений ASP.NET для Team Foundation Server. Здесь описана рекомендованная структура дерева исходного кода системы управления исходным кодом TFS.

Содержание

- Рекомендации по работе с общим кодом.
- Дополнительные ресурсы.

Задачи

- Научиться создавать структуру дерева исходного кода приложения ASP.NET.
- Разобраться, какую структуру дерева исходного кода рекомендуется использовать в TFS.

Обзор

В этой статье рассказано, как создавать структуру каталогов системы управления исходным кодом для веб-приложений ASP.NET. Поскольку в веб-проекты ASP.NET часто включаются дополнительные библиотеки классов, это необходимо предусмотреть в структуре. Папки для хранения веб-проектов ASP.NET располагаются в системе управления исходным кодом под папками верхнего уровня **/Main/Source**. Такая структура позволяет добавлять папки **Development** и **Releases** при необходимости создания ветвей для изолированной разработки и обслуживания предыдущих выпусков. Дополнительную информацию о создании структуры каталогов верхнего уровня вы найдете в разделе «Как структурировать папки управления исходным кодом в Visual Studio Team Foundation Server».

Порядок операций

- Шаг 1 — создание локальных папок веб-проекта.
- Шаг 2 — создание пустого решения.
- Шаг 3 — добавление веб-сайта в решение.
- Шаг 4 — добавление библиотеки классов (при необходимости).
- Шаг 5 — проверка структуры решения.
- Шаг 6 — проверка локальной структуры каталогов.
- Шаг 7 — добавление решения в систему управления исходным кодом.

Шаг 1 — создание локальных папок веб-проекта

На этом этапе вы создадите на своем компьютере локальную структуру каталогов для веб-проекта. Чтобы обеспечить единый подход к командной разработке и эффективную организацию проектов на своем компьютере, сгруппируйте разрабатываемый исходный код всех командных проектов,

над которыми вы работаете, в одной корневой папке, например, C:\Dev-Projects. Если такой папки еще нет на компьютере, создайте ее.

Шаг 2 — создание пустого решения

Начните создание веб-приложения ASP.NET с явного создания файла решения Visual Studio (.sln). Затем добавьте в него свой веб-сайт и все необходимые дополнительные проекты, например, библиотеки классов. Вот как создается решение в папке верхнего уровня C:\DevProjects.

1. В меню **File** выберите команду **New** и щелкните **Project**.
2. Разверните **Other Project Types** и выберите **Visual Studio Solutions**.
3. Выберите **Blank Solution**.
4. Назовите свое решение **MyWebAppSln**.
5. Присвойте свойству **Location** значение **C:\DevProjects** и щелкните **OK**.

Будет создана папка C:\DevProjects\MyWebAppSln. Visual Studio добавляет в эту папку файл решения (.sln) и файл пользовательских параметров решения (.suo). Обратите внимание, что на шаге 7 в систему управления исходным кодом добавляется только файл .sln.

Шаг 3 — добавление веб-сайта в решение

Теперь добавьте в свое решение веб-сайт ASP.NET. Эта процедура немного варьируется в зависимости от типа веб-сайта, т.е., от того, создается веб-сайт в локальной файловой системе с использованием функций веб-разработки Visual Studio или на веб-сервере с использованием Internet Information Services (IIS).

Сайт в файловой системе

Чтобы добавить в решение веб-проект, располагающийся в локальной файловой системе, выполните следующие действия:

1. В **Solution Explorer** щелкните правой кнопкой решение **MyWebAppSln**, выберите команду **Add** и щелкните **New Web Site**.
2. В диалоговом окне **Add New Web Site** не меняйте значения **Location** (по умолчанию задан вариант **File System**) и **Language** (по умолчанию **Visual C#**).
3. Присвойте параметру **Location** значение C:\DevProjects\MyWebAppSln\Source\MyWebAppWeb.
4. Щелкните **OK**, чтобы закрыть диалоговое окно **Add New Web Site**.

Обратите внимание, что мы использовали суффикс «Web» в этом примере для четкого обозначения корневой папки веб-сайта.

Сайт на веб-сервере

Создание веб-сайта ASP.NET на базе веб-сервера IIS, доступ к которому при разработке будет осуществляться по протоколу HTTP, начинается с явного создания виртуального каталога. Необходимо, чтобы каталог веб-сайта располагался в заданном расположении, а не в папке \inetpub\wwwroot.

Создание виртуального каталога веб-сайта

1. В проводнике Windows перейдите в папку C:\DevProjects\MyWebAppSln\Source.
2. Создайте в ней папку **MyWebAppWeb**.
3. Щелкните папку **MyWebAppWeb** правой кнопкой и выберите **Общий доступ и безопасность (Sharing and Security)**.
4. Перейдите на вкладку **Доступ через веб (Web Sharing)**.
5. Щелкните **Предоставить совместный доступ к папке (Share this folder)**.
6. Не меняйте значение свойства **Псевдоним (Alias)**, оставьте заданные по умолчанию разрешения доступа и разрешения для приложений и щелкните **OK**.
7. Дважды щелкните **OK**.

Добавление веб-сайта в решение

1. В **Solution Explorer** щелкните правой кнопкой решение **MyWebAppSln**, выберите **Add** и щелкните **New Web Site**.
2. В диалоговом окне **Add New Web Site** задайте в поле **Location** значение **HTTP** и не меняйте значение **Language** по умолчанию (**Visual C#**).
3. Задайте для **Location** URL-адрес *http://localhost/MyWebAppWeb*.
4. Щелкните **OK**, чтобы закрыть диалоговое окно **Add New Web Site**.

Visual Studio добавит ваши файлы Default.aspx и Default.aspx.cs в папку C:\DevProjects\MyWebAppSln\Source\MyWebAppWeb и создаст дочерние папки **Bin** и **App_Data**.

Шаг 4 — добавление библиотеки классов (при необходимости)

Если в веб-приложении используются дополнительные библиотеки классов, добавьте их следующим образом:

1. В **Solution Explorer** щелкните правой кнопкой решение **MyWebAppSln**, выберите **Add** и щелкните **New Project**.
2. Выберите тип проекта **Visual C#** и шаблон **Class Library**.
3. Введите имя **ClassLibrary**, в качестве **Location** задайте C:\DevProjects\MyWebAppSln\Source и щелкните **OK**.

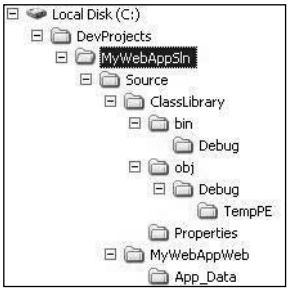
Все новые проекты будут добавляться в папку **Source**.

Шаг 5 — проверка структуры решения

В Solution Explorer проверьте структуру решения. Она должна выглядеть следующим образом:

Шаг 6 — проверка локальной структуры каталогов

В проводнике Windows проверьте локальную структуру каталогов. Она должна выглядеть следующим образом:

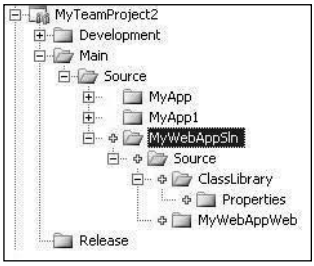


Шаг 7 — добавление решения в систему управления исходным кодом

На данном этапе вы добавите решение в систему управления исходным кодом TFS.

1. Щелкните правой кнопкой мыши свое решение и выберите **Add Solution to Source Control**.
2. В диалоговом окне **Add Solution MyWebAppSln to Source Control** выберите свой командный проект.

3. Щелкните **Make New Folder** и назовите новую папку **Main**.
4. Выделите вновь созданную папку **Main** и щелкните **Make New Folder**. Назовите новую папку **Source**.
5. Выберите вновь созданную папку **Source** и щелкните **OK**.
6. Проверьте структуру папок в системе управления исходным кодом, дважды щелкнув **Source Control** в **Team Explorer**. Структура должна выглядеть следующим образом:



7. Теперь вы можете просматривать изменения, ожидающие возвращения на сервер, и возвращать файлы исходного кода. Для этого в меню **View** выберите **Other Windows** и щелкните **Pending Changes**. Выберите свой проект и исходные файлы, которые необходимо возвратить, введите комментарий к возвращаемым изменениям и щелкните **Check In**.

Рекомендации по работе с общим кодом

Существует два основных варианта использования общего кода в веб-приложениях ASP.NET:

- использование кода из общей папки;
- создание ветви общего кода.

Использование кода из общей папки

В этом случае вы сопоставляете исходный код из общего каталога, например, из другого командного проекта, с рабочей областью на своем компьютере. При этом конфигурация, объединяющая исходный код из общего каталога с кодом вашего проекта, создается на вашем компьютере.

Преимущество этого подхода в том, что изменения в общем коде переносятся в вашу рабочую область при каждой загрузке последней версии исходного файла. Допустим, у вас имеется командный проект Common, являющийся контейнером для общего исходного кода. Чтобы использовать код этого проекта в другом проекте, вы должны сопоставить оба проекта с общей папкой на компьютере разработчика. Например:

- C:\DevProjects\MyWebAppSln\
- C:\DevProjects\SharedCommon\

Используются следующие сопоставления рабочих областей:

Папка системы управления исходным кодом	Локальная папка
$/MyTeamProject1/Main/Source/MyWebAppApp	C:\DevProjects\MyWebAppSln
$/MyTeamProject2/Main/Source/Common	C:\DevProjects\Common

Дополнительную информацию по этому вопросу вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Создание ветви общего кода

В этом случае вы создаете в своем командном проекте ветвь для исходного кода из общего командного проекта. При этом также возникает конфигурация, объединяющая исходный код из общей папки и ваш проект. Отличие состоит в том, что изменения общего исходного кода переносятся в процессе слияния ветвей. Таким образом, внесение изменений в общий код становится более явным. Вы сами принимаете решение о слиянии для переноса последних изменений.

В качестве примера опять рассмотрим командный проект Common, являющийся контейнером для совместно используемого кода. Чтобы создать ветвь для общего кода, выполните следующие действия:

1. В **Source Control Explorer** щелкните правой кнопкой мыши корневую папку командного проекта Common.
2. Щелкните **Branch**.
3. В диалоговом окне **Branch** в качестве **Target** задайте корневую папку командного проекта $/MyTeamProject1/Main/Source/ и щелкните **OK**.
4. После завершения ветвления не забудьте возвратить созданную ветвь в систему управления исходным кодом.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочих областей вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию о ссылках на файлы сборок в разных командных проектах вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Как структурировать приложения Windows в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).
- Microsoft Windows® Form Applications.

Описание

В этой статье подробно разбирается процесс организации и структурирования приложений Windows Form в Team Foundation Server. Описана рекомендованная структура дерева исходного кода в системе управления исходным кодом TFS.

Содержание

- Шаг 7 — проверка локальной структуры каталогов.
- Шаг 8 — добавление решения в систему управления исходным кодом.
- Рекомендации по работе с совместно используемым кодом.
- Дополнительные ресурсы.

Задачи

- Научиться создавать структуру дерева исходного кода приложения Windows Forms.
- Разобраться, какую структуру дерева рекомендуется использовать в системе управления исходным кодом TFS.

Обзор

В статье показано, как создавать структуру каталогов системы управления исходным кодом для приложений Windows Forms. Поскольку в проекты Windows Forms часто включаются дополнительные библиотеки классов и элементов управления, структура должна это учитывать. В системе управления исходным кодом папки для хранения проектов Windows Forms располагаются под папками верхнего уровня **/Main/Source**. Такая структура позволяет добавлять папки **Development** и **Releases** при необходимости создания ветвей для изолированной разработки и обслуживания выпущенных версий. Дополнительную информацию о создании структуры каталогов верхнего уровня вы найдете в разделе «Как структурировать папки управления исходным кодом в Visual Studio Team Foundation Server».

Порядок операций

- Шаг 1 — создание локальных папок проекта Windows Forms.
- Шаг 2 — создание пустого решения.
- Шаг 3 — добавление проекта Windows Forms в решение.
- Шаг 4 — добавление библиотеки элементов управления (при необходимости).
- Шаг 5 — добавление библиотеки классов (при необходимости).
- Шаг 6 — проверка структуры решения.
- Шаг 7 — проверка локальной структуры каталогов.
- Шаг 8 — добавление решения в систему управления исходным кодом.

Шаг 1 — создание локальных папок проекта Windows Forms

На этом этапе вы создадите на своем компьютере локальную структуру каталогов для проекта Windows Forms. Чтобы обеспечить единый подход к командной разработке и эффективную организацию проектов на своем компьютере, сгруппируйте разрабатываемый исходный код всех командных проектов, над которыми вы работаете, в одной корневой папке, например, C:\DevProjects. Если такой папки еще нет на компьютере, создайте ее.

Шаг 2 — создание пустого решения

Начните создание приложения Windows Forms с явного создания файла решения Visual Studio (.sln). Затем добавьте в него проект Windows Forms и все необходимые дополнительные проекты, например, библиотеки классов или элементов управления. Вот как создается решение в папке верхнего уровня C:\DevProjects.

1. В меню **File** выберите команду **New** и щелкните **Project**.
2. Разверните **Other Project Types** и выберите **Visual Studio Solutions**.
3. Выберите **Blank Solution**.
4. Назовите свое решение **MyApp**.
5. Присвойте свойству **Location** значение **C:\DevProjects** и щелкните **OK**.

Будет создана папка C:\DevProjects\MyApp. Visual Studio добавляет в эту папку файл решения (.sln) и файл пользовательских параметров решения (.suo). Обратите внимание, что на шаге 8 в систему управления исходным кодом добавляется только файл .sln.

Шаг 3 — добавление проекта Windows Forms в решение

Добавьте проект Windows Forms в свое решение.

1. В **Solution Explorer** щелкните правой кнопкой решение **MyApp**, выберите **Add** и щелкните **New Project**.
2. В диалоговом окне **Add New Project** выберите **Visual C#** в качестве типа проекта и **Windows Application** в качестве шаблона.
3. Присвойте параметру **Location** значение **C:\DevProjects\MyApp\Source**.
4. Назовите проект **MyWinFormApp**.
5. Щелкните **OK**, чтобы закрыть диалоговое окно **Add New Project** и добавить проект.

Шаг 4 — добавление библиотеки элементов управления (при необходимости)

Если в приложении Windows Forms используются дополнительные библиотеки элементов управления, добавьте их следующим образом:

1. В **Solution Explorer** щелкните правой кнопкой свое решение, выберите **Add** и щелкните **New Project**.
2. В диалоговом окне **Add New Project** выберите **Visual C#** в качестве типа проекта и **Windows Control Library** в качестве шаблона.
3. Присвойте параметру **Location** значение **C:\DevProjects\MyApp\Source**.
4. Назовите библиотеку элементов управления **ControlLibrary**.
5. Щелкните **OK**, чтобы закрыть диалоговое окно **Add New Project**.

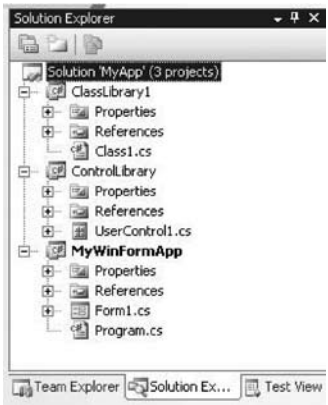
Шаг 5 — добавление библиотеки классов (при необходимости)

Если в приложении Windows Forms используются дополнительные библиотеки классов, добавьте их следующим образом:

1. В **Solution Explorer** щелкните правой кнопкой свое решение, выберите **Add** и щелкните **New Project**.
2. В диалоговом окне **Add New Project** выберите **Visual C#** в качестве типа проекта и **Class Library** в качестве шаблона.
3. Присвойте параметру **Location** значение **C:\DevProjects\MyApp\Source**.
4. Назовите библиотеку классов **ClassLibrary1**
5. Щелкните **OK**, чтобы закрыть диалоговое окно **Add New Project**.

Шаг 6 — проверка структуры решения

В Solution Explorer проверьте структуру решения. Она должна выглядеть следующим образом:



Шаг 7 — проверка локальной структуры каталогов

В проводнике Windows проверьте локальную структуру каталогов. Она должна выглядеть следующим образом:

Шаг 8 — добавление решения в систему управления исходным кодом

Добавьте в систему управления исходным кодом TFS свое решение, содержащее проект Windows Forms, а также, возможно, библиотеки классов и элементов управления.

1. Щелкните правой кнопкой мыши **MyApp** и выберите **Add Solution to Source Control**.
2. В диалоговом окне **Add Solution MyApp to Source Control** выберите свой командный проект.
3. В диалоговом окне щелкните **Make New Folder** и назовите новую папку **Main**.
4. Выделите вновь созданную папку **Main** и щелкните **Make New Folder**. Назовите новую папку **Source**.
5. Выделите вновь созданную папку **Source** и щелкните **OK**.
6. Проверьте структуру папок в системе управления исходным кодом, щелкнув дважды **Source Control** в **Team Explorer**. При этом откроется Source Control Explorer. Структура папок должна выглядеть следующим образом:

7. Теперь вы можете просматривать изменения, ожидающие возвращения на сервер, и возвращать файлы исходного кода. В меню **View** выберите **Other Windows** и щелкните **Pending Changes**. Выберите проект и исходные файлы, которые необходимо возвратить, введите комментарий к возвращаемым изменениям и щелкните **Check In**.

Рекомендации по работе с общим кодом

Существует два основных варианта использования общего кода в веб-приложениях Windows Forms:

Использование кода из общей папки

В этом случае вы сопоставляете исходный код из общего каталога, например, из другого командного проекта, с рабочей областью на своем компьютере. При этом конфигурация, объединяющая исходный код из общего каталога с кодом вашего проекта, создается на вашем компьютере.

Преимущество этого подхода в том, что изменения в общем коде переносятся в вашу рабочую область при каждой загрузке последней версии исходного файла. Допустим, у вас имеется командный проект Common, являющийся контейнером для общего исходного кода. Чтобы использовать код этого проекта в другом проекте, вы должны сопоставить оба проекта с общей папкой на компьютере разработчика. Например:

- C:\MyProjects\MyApp\
- C:\MyProjects\SharedCommon\

Используются следующие сопоставления рабочих областей:

Папка системы управления исходным кодом	Локальная папка
$/MyTeamProject1/Main/Source/MyApp	C:\DevProjects\MyApp
$/MyTeamProject2/Main/Source/Common	C:\DevProjects\Common

Дополнительную информацию по этому вопросу вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Создание ветви общего кода

В этом случае вы создаете в своем командном проекте ветвь для исходного кода из общего командного проекта. При этом также возникает конфигурация, объединяющая исходный код из общей папки и ваш проект. Отличие состоит в том, что изменения общего исходного кода переносятся в процессе слияния ветвей. Таким образом, внесение изменений в общий код становится более явным. Вы сами принимаете решение о слиянии для переноса последних изменений.

В качестве примера опять рассмотрим командный проект Common, являющийся контейнером для совместно используемого кода. Чтобы создать ветвь для общего кода, выполните следующие действия:

1. В **Source Control Explorer** щелкните правой кнопкой мыши корневую папку командного проекта Common.
2. Щелкните **Branch**.
3. В диалоговом окне **Branch** в качестве **Target** задайте корневую папку командного проекта $/MyTeamProject1/Main/Source/ и щелкните **OK**.
4. После завершения ветвления не забудьте возвратить созданную ветвь в систему управления исходным кодом.

Дополнительные ресурсы

- Дополнительную информацию о создании рабочих областей вы найдете в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.
- Дополнительную информацию о ссылках на файлы сборок в разных командных проектах вы найдете в статье «Working with multiple team projects in Team Build» по адресу *http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx*.

Как структурировать папки управления исходным кодом в Visual Studio Team Foundation Server

Область применения

- Microsoft® Visual Studio® 2005 Team Foundation Server (TFS).
- Microsoft Visual Studio Team System (VSTS).

Описание

В этой статье подробно разбирается процесс организации структуры каталогов TFS. Эти рекомендации основаны на практическом опыте как Microsoft, так и пользователей продуктов корпорации. Представленная здесь структура является лишь ориентиром, гораздо важнее понять, на каких основаниях она построена. Разберитесь с этими основаниями и используйте их при формировании собственной структуры и выработке соглашения о наименовании папок в вашем сценарии.

Содержание

- Шаг 6 — создание папки Releases для изоляции выпущенных сборок (при необходимости).
- Дополнительные рекомендации.
- Дополнительные ресурсы.

Задачи

- Научиться создавать сопоставления рабочих областей.
- Разобраться, как структурировать и именовать папки системы управления исходным кодом.
- Разобраться, как ветвление влияет на структуру исходного кода.

Обзор

В этой статье описывается создание структуры каталогов системы управления исходным кодом, которая подходит для большинства типичных проектов. В представленной структуре есть три папки верхнего уровня:

- **Main** Главная корневая папка, выполняющая роль контейнера для основного дерева исходного кода, а также для артефактов проекта, например, проектной документации, сценариев и тестирования. Также в папке **Main** располагаются файлы Visual Studio Solution (.sln).
- **Development** Необязательная папка корневого уровня, которая может использоваться для изоляции разработки компонентов или работы отдельных команд. Она создается как ветвь папки **Main**.
- **Releases** Необязательная папка корневого уровня для предыдущих сборок, которые необходимо изолировать от основной разработки, например, чтобы обслуживать выпущенную версию.

Представляемая в данной статье структура каталогов выглядит следующим образом:

```
{
/Main                      →Может содержать файлы решений (.sln),
                             охватывающих несколько проектов
            /Source
      /MyApp1              →Содержит файл MyApp1.sln
         /Source           →Содержит папку для всего исходного кода
            /ClassLibrary1 →Содержит ClassLibrary1.csproj
            /MyApp1Web     →Содержит Default.aspx
            /Build         →Содержит результат сборки (двоичные файлы)
   /Docs                   →Содержит документацию по продукту и пр.
   /Tests                  →Содержит тесты

/Development
   /FeatureBranch1
```

```
      /Source
         /MyApp1
            /Source
               /MyApp1Web
               /ClassLibrary1
   /FeatureBranch2

/Releases
            /Release1
               /Source
                  /MyApp1
                     /Source
                        /MyApp1Web
                        /ClassLibrary1
            /Release 1.1
            /Release 1.2
}
```

Помните, что эта структура — лишь ориентир с примерным набором папок. Важно, чтобы структура соответствовала сути проекта, а имена папок отражали их назначение. Подробно принципы построения структуры рассматриваются в главе 5.

Порядок операций

- Шаг 1 — создание сопоставления рабочей области.
- Шаг 2 — создание папки Main.
- Шаг 3 — создание папок для артефактов проекта.
- Шаг 4 — добавление решения в дерево исходного кода.
- Шаг 5 — создание папки Development для изоляции разработки (при необходимости).
- Шаг 6 — создание папки Releases для изоляции выпущенных сборок (при необходимости).

Шаг 1 — создание сопоставления рабочей области

На начальном этапе для описания соответствия между структурой каталогов на сервере TFS и на клиенте создается сопоставление рабочей области. Сначала вы создаете дерево исходного кода в своей рабочей области и затем передаете его на сервер TFS.

Сопоставление рабочей области можно создать двумя способами: явно или выполнив для командного проекта операцию Get.

Явное создание сопоставления рабочей области

1. В меню **File** Visual Studio выберите **Source Control** и щелкните **Workspaces**.
2. В диалоговом окне **Manage Workspaces** выберите имя своего компьютера и щелкните **Edit**.
3. В диалоговом окне **Edit Workspace** в списке **Working folders** щелкните **Click here to enter a new working folder**.
4. Щелкните многоточие, выберите свой командный проект (например, **MyTeamProject1**) и щелкните **OK**.
5. Щелкните ячейку локальной папки, чтобы появилась еще одна кнопка с многоточием.
6. Щелкните многоточие под **Local Folder** и выберите локальную папку на своем компьютере, в которой вы хотите разместить свой командный проект, например, C:\DevProjects\MyTeamProject1.
7. Дважды щелкните **OK**, чтобы закрыть диалоговое окно **Edit Workspace**.
8. Щелкните **OK** в ответ на сообщение Microsoft Visual Studio с информацией об изменении одной или нескольких рабочих папок.
9. Щелкните **Close**, чтобы закрыть диалоговое окно **Manage Workspaces**.

Выполнение операции Get для командного проекта

1. В **Team Explorer** разверните узел своего командного проекта, например, **MyTeamProject1**.
2. Щелкните дважды **Source Control** под командным проектом.
3. В **Source Control Explorer** щелкните правой кнопкой корневую папку **MyTeamProject1** и выберите команду **Get Latest Version**.
4. В диалоговом окне **Browse For Folder** выберите локальный путь (например, C:\DevProjects\MyTeamProject1) и щелкните **OK**. Корневая папка командного проекта с TFS сопоставлена с локальным каталогом на вашем компьютере.

Шаг 2 — создание папки Main

Теперь создадим в системе управления исходным кодом новую корневую папку **Main** для вашего командного проекта. Она используется как корневой контейнер для исходного кода и других артефактов проекта. В ней также могут располагаться файлы решений Visual Studio (.sln), охватывающих несколько проектов. Папки решений также хранятся в папках соответствующих приложений, располагающихся на более низком уровне структуры дерева.

Чтобы создать папку Main, выполните следующие действия:

1. В Team Explorer разверните свой командный проект и дважды щелкните

Source Control.

2. В Source Control Explorer выберите корневую папку командного проекта.
3. Щелкните правой кнопкой мыши правую панель и выберите **New Folder**.
4. Введите **Main** и нажмите Enter.

Шаг 3 — создание папок для артефактов проекта

Далее в папке **Main** создаются папки для размещения дерева исходного кода и сопутствующих артефактов, например, сборок, документации, файлов сценариев и тестов:

```
{
/Main
   /Build
   /Docs
   /Source
   /Tests
}
```

Чтобы создать папки для размещения ресурсов проекта, выполните следующие действия:

1. Разверните командный проект и в левой панели Source Control Explorer выберите папку **Main**.
2. Щелкните правой кнопкой правую панель, щелкните **New Folder**, введите **Build** и нажмите Enter.
3. Повторите шаг 2, чтобы создать остальные подпапки папки **Main**.

Шаг 4 — добавление решения в дерево исходного кода

Теперь добавим в дерево исходного кода TFS решение Visual Studio (.sln), проект (.vsproj, .vbproj) и исходные файлы.

Поскольку на шаге 1 вы сопоставили корневой узел с папкой C:\DevProjects\MyTeamProject1, убедитесь, что структуры папок на клиенте и на сервере синхронизированы. Файл решения (.sln) должен находиться в папке C:\DevProjects\MyTeamProject1\Main\Source\MyApp1. Связанные с проектом файлы компонентов MyApp1Web и ClassLibrary1 должны располагаться в папке C:\DevProjects\MyTeamProject1\Main\Source\MyApp1\Source.

Чтобы добавить решение в дерево исходного кода, выполните следующие действия:

1. Откройте решение в Visual Studio.
2. В Solution Explorer щелкните решение правой кнопкой и выберите **Add Solution to Source Control**. Поскольку сопоставление рабочей области

уже было создано на шаге 1, теперь в систему добавляется решение.

3. Убедитесь, что значение параметра **Local Workspace** выбрано правильно. Это должна быть та же рабочая область, что использовалась при создании структуры каталогов в системе управления исходным кодом TFS.
4. Щелкните **OK**, чтобы добавить решение в систему управления исходным кодом.

Структура каталогов в системе управления исходным кодом должна выглядеть следующим образом:

```
{
/Main
   /Source
      /MyApp1
         /Source
            /MyApp1Web
            /ClassLibrary1
/Build
   /Docs
   /Tests
}
```

Дополнительную информацию о настройке структуры каталогов на стороне клиента вы найдете в статьях «Как структурировать приложения Windows в Visual Studio Team Foundation Server» и «Как структурировать приложения ASP.NET в Visual Studio Team Foundation Server».

Структура каталогов при наличии в проекте модульных тестов

Если в проект включены модульные тесты, используйте следующую структуру каталогов, чтобы обеспечить их хранение отдельно от главной папки исходного кода:

```
{
/Main
   /Source
      /MyApp1
         /Source
            /MyApp1Web
            /ClassLibrary1
         /UnitTests
            /Build
   /Docs
   /Tests
}
```

Шаг 5 — создание папки Development для изоляции разработки (при необходимости)

Чтобы изолировать работу над компонентом или работу отдельной команды, создайте папку **Development** путем ветвления папки **Main**. Помните, что перед ветвлением необходимо возвратить все изменения.

Чтобы создать ветвь **Development**, выполните следующие действия:

1. Создайте корневую папку **Development** (на одном уровне с **Main**).
2. Создайте подпапку **FeatureBranch1**.
3. Щелкните папку $/TeamProject/Main/Source правой кнопкой и выберите команду **Branch**.

> **Примечание** Если команда **Branch** недоступна, проверьте, все ли изменения возвращены.

4. В диалоговом окне **Branch** щелкните **Browse**, выберите MyTeamProject/Development/FeatureBranch1 и щелкните **OK**.
5. Щелкните **OK**, чтобы создать ветвь.
6. Возвратите изменения на сервер.

Структура каталогов в системе управления исходным кодом теперь должна выглядеть следующим образом:

```
{
/Development
   /FeatureBranch1
      /Source
         /MyApp1
            /Source
               /MyApp1Web
               /ClassLibrary1
   /FeatureBranch2

/Main
   /Source
      /MyApp1
         /Source
            /MyApp1Web
            /ClassLibrary1
   /Build
   /Docs
   /Source
   /Tests
}
```

Шаг 6 — создание папки Releases для изоляции выпущенных сборок (при необходимости)

Если параллельно с основной разработкой требуется обслуживать ранее выпущенные версии, изолируйте работы по обслуживанию, создав папку **Releases**. В ней можно создавать подпапки для каждой выпущенной версии.

Чтобы создать ветвь **Releases**, выполните следующие действия:

1. Создайте корневую папку **Releases** (на одном уровне с папками **Main** и **Development**).
2. Создайте подпапку **Release1**.
3. Щелкните папку $/TeamProject/Development/Source правой кнопкой и выберите **Branch**.

> **Примечание** Если команда **Branch** недоступна, проверьте, все ли изменения возвращены.

5. В диалоговом окне **Branch** щелкните **Browse**, выберите $/TeamProject/Maintenance/Release1 и щелкните **OK**.
6. В разделе **Branch from version** в списке **By** щелкните **Label** и введите имя метки в одноименное поле. Чтобы выбрать метку, щелкните кнопку с многоточием рядом с полем **Label**.
7. Щелкните **OK**, чтобы создать ветвь.

Новая структура каталогов в системе управления исходным кодом должна выглядеть следующим образом:

```
{
/Main
   /Source
      /MyApp1
         /Source
            /MyApp1Web
            /ClassLibrary1
   /Build
   /Docs
   /Source
   /Tests

/Releases
            /Release1
      /Source
         /MyApp1
            /Source
               /MyApp1Web
```

```
            /ClassLibrary1
        /Release 1.1
        /Release 1.2
}
```

Дополнительные рекомендации

При создании структуры каталогов в TFS учитывайте следующие рекомендации:

- Ветвление следует выполнять только в случае необходимости. Выпускаемые версии можно пометить метками и создать для них ветви позже, когда возникает реальная потребность в их изоляции.
- Рассмотренная выше структура каталогов идеально подходит для сценариев, в которых проект состоит из одного файла решения Visual Studio (.sln), содержащего все файлы проекта. При таком сценарии в папке **Main** размещается файл .sln и подпапки для каждого файла проекта (.vsproj, .vbproj). В нашем примере папками проектов являются MyApp1, MyApp2 и т. д. Такой подход может также использоваться и при наличии одного файла проекта, в частности, если имеется только файл проекта без файла решения.
- В сценариях с несколькими решениями файлы .sln могут размещаться в папке **Main**.

Дополнительные ресурсы

- Создание рабочей области подробно рассматривается в статье «How to: Create a Workspace» по адресу *http://msdn2.microsoft.com/en-us/library/ms181384(VS.80).aspx*.

Ресурсы Team Foundation Server

Содержание

- Группа patterns & practices.
- Сайты CodePlex.
- Веб-сайты Team Foundation.
- Партнеры и поставщики услуг.
- Форумы.
- Блоги Microsoft Team Foundation.
- Пакеты обновлений.
- Обучение.

Группа patterns & practices

Подробнее о группе «patterns and practices» — на домашней странице по адресу *http://msdn.microsoft.com/practices/*.

Сайты CodePlex

- Справочник по ветвлению Team Foundation — *http://www.codeplex.com/BranchingGuidance*.
- Обозреватель руководств — *http://www.codeplex.com/guidanceExplorer*.
- Справочник по Team Foundation Server — *http://www.codeplex.com/TFSGuide*.
- Руководство по Team Foundation Server — *http://www.codeplex.com/VSTSGuidance*.

Веб-сайты Team Foundation

- Центр команды Team Foundation Server — *http://msdn2.microsoft.com/en-us/teamsystem/aa718934.aspx*.

- Часто задаваемые вопросы о Team Foundation Server — *http://msdn2.microsoft.com/en-us/teamsystem/aa718916.aspx*.
- Видеозаписи и презентации Team System — *http://msdn2.microsoft.com/en-us/teamsystem/aa718837.aspx*.
- Документация MSDN для Team Foundation Server — *http://msdn2.microsoft.com/en-us/library/ms181232(vs.80).aspx*.
- Справочник по установке Team Foundation Server — *http://go.microsoft.com/fwlink/?linkid=40042*.

Партнеры и поставщики услуг

- Каталог партнеров — *http://catalog.vsipmembers.com/catalog*.
- Кроссплатформенная поддержка Teamprise — *http://www.teamprise.com*.
- Поддержка Scrum от компании Conchango — *http://www.conchango.com/Web/Public/Content/Home.aspx*.
- ПО для переноса исходного кода ComponentSoftware — *http://www.componentsoftware.com/Products/converter/index.htm*.
- Управление требованиями от Borland — *http://www.borland.com*.
- Моделирование бизнес-процессов от RavenFlow — *http://www.n8systems.com*.

Форумы

Список форумов MSDN вы найдете по адресу *http://forums.microsoft.com/MSDN/default.aspx?ForumGroupID=5&SiteID=1*.

Форум	Адрес
Team Foundation Server — общие вопросы	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=22&SiteID=1
Team Foundation Server — настройка	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=68&SiteID=1
Team Foundation Server — администрирование	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=477&SiteID=1
Team Foundation Server — автоматизация сборки	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=481&SiteID=1
Team Foundation Server — Power Toys	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1
Team Foundation Server — шаблоны процессов	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=482&SiteID=1
Team Foundation Server — отчеты и хранилище	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=480&SiteID=1

Форум	Адрес
Team Foundation Server — Team System Web Access	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=1466&SiteID=1
Team Foundation Server — управление версиями	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=478&SiteID=1
Team Foundation Server — отслеживание рабочих элементов	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=479&SiteID=1

Блоги Microsoft Team Foundation

- Ask Burton — http://blogs.msdn.com/askburton
- Brian Harry — http://blogs.msdn.com/bharry
- Buck Hodges — http://blogs.msdn.com/buckh
- James Manning — http://blogs.msdn.com/jmanning
- Rob Caron — http://blogs.msdn.com/robcaron
- Team Foundation Blog — http://blogs.msdn.com/team_foundation

Пакеты обновлений

- Visual Studio 2005 SP1 — *http://msdn2.microsoft.com/en-gb/vstudio/bb265237.aspx*.
- Microsoft® Visual Studio® 2005 Team Foundation Server Service Pack 1 — *http://www.microsoft.com/downloads/details.aspx?FamilyId=A9AB638C-04D2-4AEE-8AE8-9F00DD454AB8*.
- Visual Studio 2005 Service Pack 1 Update для Windows Vista — *http://www.microsoft.com/downloads/details.aspx?FamilyID=90e2942d-3ad1-4873-a2ee-4acc0aace5b6&displaylang=en*.
- SQL Server 2005 Service Pack 2 — *http://technet.microsoft.com/en-us/sqlserver/bb426877.aspx*.

Обучение

- Список обучающих компаний — *http://msdn2.microsoft.com/en-us/teamsystem/aa718793.aspx*.
- Developmentor — *http://www.develop.com/*.
- Pluralsight — *http://www.pluralsight.com*.
- Notion Solutions — *http://www.notionsolutions.com*.