Visual Studio Team Foundation Server によるチーム開発

patterns & practices

J.D.Meier

Jason Taylor

Alex Mackman

Prashant Bansode

Kevin Jones

序文 (Jeff Beehler)

序文

Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) のリリース前に、まず私たちが実際に使用して TFS を開発しました。このプロジェクトの最後の 18 か月間、プロジェクトの開発ライフ サイクルを管理するために、TFS を広範に使用しました (これは "dogfooding" として一般的に知られている手法です)。この dogfooding を通じて、作成中であったこの強力なシステムについて多くのことを学びました。品質に関する多くの問題を確実に発見して修正した結果、製品は格段に安定したものとなり、他の手法でチェックした場合よりもはるかにすぐれた機能を実現できるようになりました。そして、おそらくもっと重要なことは、作成中のツールを最もよく使用する方法 (および使用しない方法) について学んだことです。この経験は、実務におけるお客様からのフィードバックと連動して、このガイドのベースを形成しています。

一見、この情報は、製品のドキュメントに含まれるべきもの、または製品のドキュメントに代わるものであると思われるかもしれません。実際に、一時、そのように考えましたが、J.D.Meier およびこのガイドの他の著者と協力して作業していくうちに、この情報と製品ドキュメントを切り離すことが自然であり、かつ重要であることが明らかになりました。2 冊のガイドを、自動車の「所有者のマニュアル」と「ドライバーのマニュアル」に例えることが最も適した説明だと思います。これらのマニュアルは、異なる理由により、両方とも必要であるからです。従来、製品チームは、製品のドキュメント化に焦点を当てており、そのガイダンスの面は他の人に任せてきました。他の人がサポートしてくれることにまだ頼っていますが、より多くの時間と労力をガイダンスに費やし始めています。これは、ガイダンスが、製品の導入の成功に対して、および全体的な顧客満足の向上において果たす役割に対して、いかに重要であるかを認識したためです。

TFS は自動車と同様に、あなたやあなたのチームを、望むところであれば、ほとんどどこへでも連れて行くことができる強力なツールです。このガイドは、目的の実現をサポートすることができます。すべてのチームは、その特定のニーズや歴史によって、少し異なる方法で TFS にアプローチします。この理由により、読者が全体像を知りたい場合は最初から最後まで読み、ニーズが決まっている場合は特

定のトピックだけを読むことができるように、このガイドを作成しました。

お客様のフィードバックは、最初にこのガイドを作成する際の手がかりとなりました。また、現在でも継続して方向性、および目的を達成するための方法を決定する手助けとして重要な役割を果たしています。このようなプロジェクトにおいてコミュニティと密な関わりを持つことは、単独でガイドを作成するよりも、内容がより有用なものになり、最終的にはより良いガイドが完成すると確信しています。このことを念頭において、実際のユーザーは何について書けばよいか、推奨するベスト プラクティスは何か、内容をどのように編成すればよいかを決定できるようサポートしてくれました。ただし、全体的な作業は終わったわけではありません。このガイドをより良いものにするために、他に記載すべき内容について教えてくださるようお願いします。TFS が対象としている分野は非常に広いため、ときには私たちでさえ対処できなくなりそうです。頂いたご意見によって、開発したこのツールをお客様が最適にご使用いただけるようサポートできます。チームを 1 つにまとめてすばらしいソフトウェアを実現するために、TFS を設計しました。TFS の dogfooding によって、私たちのチームが 1 つにまとまり、結果としてすばらしい製品になったことに同感してくださると信じています。このガイドは、あなたのチームの次のプロジェクトにおいて、このビジョンを実現するうえでも役に立つでしょう。

ご幸運を。

Jeff Beehler

Visual Studio Team System のチーフ スタッフ

2007 年 7 月

Jeff Beehler は *Team System* のチーフ スタッフです。コロラド大学卒業後、*1990* 年に *Microsoft* 煮入社し、*Visual C++* の初期のバージョンに従事しました。*1996* 年、コンサルティングなどの他の分野を追求するために *Microsoft* を退社し、小学校で教えながら家庭を持ちました。*2003* 年、*Visual Studio Team System* に従事するために *Microsoft* に戻り、ここで計画から作成、リリースまでプロ

ジェクトの多くの局面に関わりました。*Team System* のすべてのパーツに関する熱心な *dogfooder* であり、自身の仕事がより良くできるように取り組んでいます。仕事を離れると、写真を撮ったり、広大なアメリカ北西部のアウトドアで遊んだりして、家族と過ごす時間を楽しんでいます。

序文 (Rob Caron)

序文

Visual Studio Team System の初期の頃からずっと、ソフトウェア開発チームには、リリース前に提供できるより多くのコンテンツが必要であることを認識していました。具体的に言うと、実証済みのガイダンスとベスト プラクティスが必要であることがわかっていましたが、製品をさまざまな環境、プロジェクト、シナリオのチームで試してみて、実際にどのように機能するかを検証しなければ、この知識は得られません。

残念なことに、実証済みのガイダンスおよびベスト プラクティスを検証して開発するには、時間がかかります。過去数年間に、Team System 全般、特に Team Foundation Server の使用方法について多くのことを学びましたが、これらの知識を発見して会得することは必ずしも簡単ではありませんでした。このことを実現するためには、patterns & practices のベテランである J.D. Meier および彼のチームが、組織立って専念して何か月も従事しました。

そしてやっと実現しました。Visual Studio Team Foundation Server によるチーム開発は、このプロジェクトに直接および間接的に貢献した数え切れない人たちの総体的な知識に相当します。この内容を収集したチームは、先人達の経験をないがしろにしませんでした。"整備されていない環境" における Team System の採用および使用の方法について、チームがよりよく理解できるよう、散在しているブログの投稿、フォーラムのスレッド、記事を選別しました。

その途中で、ソフトウェア開発チームに影響を与える主な分野を調べて、成功を予測および再現するにはどの手法がポイントとなるかを特定しました。いくつかの最も有用な情報によって、Team Foundation Server の多くの機能領域 (作業項目の追跡、レポーティング、プロセス テンプレートな

ど) が明確になります。

振り返ってみて、「最善の推測」が詰まった内容を提供するのではなく、ドキュメンテーション チームとして、この作業に時間をかけたことに感謝しています。今までこのようなガイドがなく苦労した方々に陳謝すると共に、それでも Team System を先駆けて使用してくれた方々に感謝します。

Rob Caron
Lead Product Manager
Microsoft Corporation
2007 年 7 月

Rob Caron は *Microsoft* の開発者コンテンツ戦略のリード プロダクト マネージャです。*Visual Studio* 製品ドキュメントのライターとして、*1999* 年から *Microsoft* で働いています。長年、*Visual Studio .NET 2002*、*Visual Studio .NET 2003*、および *Visual Studio Team System* に関するコンテンツを寄稿しました。*2004* 年の半ば、彼はブログを始め、これが *Team System* の情報の中心となりました。*7* 年間のコンテンツ作成の後、*2006* 年の秋に開発マーケティング チームへ異動しました。現在は、*Microsoft* においてますます複雑になっている開発者のストーリーを "シンプル" にするということを目的としたグループを牽引しています。

はじめに

このガイドは、Visual Studio 2005 Team Foundation Server を最大限に利用する方法を紹介し、チームベースのソフトウェア開発の効率を向上させるための支援をするものです。Team Foundation Server を既に使用している場合であっても、これから採用する場合であっても、特別なシナリオに適合するガイダンスおよび見識を見つけることができるでしょう。

このガイドの情報は、お客様のフィードバックおよび製品サポートに加えて、現場および第一線での経験から学んだ実践がベースとなっています。このガイダンスはタスクベースであり、構成は以下のようになっています。

- **「第 1 部 基本事項」**は、Team Foundation Server によるチーム開発の概要について説明します。開発およびテスト環境も含めて、ソフトウェアの開発環境についての全体像を示します。また、Team Foundation Server の基本的なアーキテクチャについても学習します。
- **「第 2 部 ソース管理」**は、ソース コードを構造化する方法、およびその依存関係を管理する方法について紹介します。ここでは、開発で分離が必要な場合に、分岐とマージの方針をどのように決定するかについても示しています。
- **「第 3 部 ビルド」**は、チームのビルドを設定する方法、開発チームに対して継続的な統合ビルドを作成する方法、およびスケジュールされたビルドをテスト チームにドロップする方法について紹介します。また、一般的な問題、およびそれらの問題への対処についても取り上げます。
- **「第 4 部 大規模なプロジェクトでの考慮事項」**は、大規模なプロジェクトで作業する場合に、必要なその他の考慮事項について説明します。
- **「第 5 部 プロジェクト管理」**は、Team Foundation Server の作業項目、エリア、イテレーションを使用して、使用しているプロジェクト管理のアプローチに関係なく、開発プロセスを合理化する方法について説明します。
- **「第 6 部 プロセス テンプレート」**は、Team Foundation Server に付随してすぐに使用できる、プロセス テンプレートとプロセス ガイダンスを最大限に利用する方法について説明します。また、プロセス テンプレートをカスタマイズする方法、作業項目とワークフローを修正して、既にチームで使用しているソフトウェア エンジニアリング プロセスと対応付ける方法についても説明します。
- **「第 7 部 レポーティング」**は、Team Foundation Server のすべてのコンポーネントについて、

データ ストアを共通のレポート メカニズムに統合する方法について説明します。デフォルトのレポートの使用方法、および独自のカスタム レポートの作成方法についても学習します。

- **「第 8 部 チーム環境のセットアップと保守」**は、Team Foundation Server の配置における不明点を解明します。シングル サーバーと複数サーバーの配置をどのように選択するかについても説明します。また、リモート開発チームをサポートする方法、および Team Foundation Server のパフォーマンスを最大にする方法についても学習します。
- **「第 9 部 Visual Studio 2008 Team Foundation Server」**は、Team Foundation Server の次期バージョンにおける変更点について説明します。ここでは、どのような機能が計画されているか、またどのような機能が大幅に改良されているかについて知ることができます。変更の中には、このガイドのどこか他の場所のガイダンスに影響を与えるものがあるため、Team Foundation Server のアップグレード計画を調整する場合は、このセクションを使用してください。

- **ガイドライン**は、Team System のビルド、プロジェクト管理、レポーティングおよびソース管理に対して簡潔な推奨を提供します。各ガイドラインには、ユーザーが行うべき内容、およびガイドラインに従う理由とその方法が示されています。
- **プラクティス**は、*Microsoft* の現場で Team Foundation Server を使用して、開発チームが学んだ教訓に基づいた、ベスト プラクティスについて説明します。それぞれのプラクティスでは、Team Foundation Server を使用したチームの効率にとって重要なタスクを実現する方法について、焦点を当てています。
- **質問と回答**は、Team Foundation Server のソース管理に関する一般的な質問に対する回答を提示します。
- **方法**は、Team Foundation Server を使用して特定のタスクを実現する方法について、段階的なガイダンスを詳しく提示します。
- **リソース**は、Team Foundation Server についてさらに詳しく調べるときに使用できる Web サイト、サービス プロバイダ、フォーラムおよびブログの要約を提示しています。これらの内容は、ツールセットにおける最新の開発を完全に網羅しています。

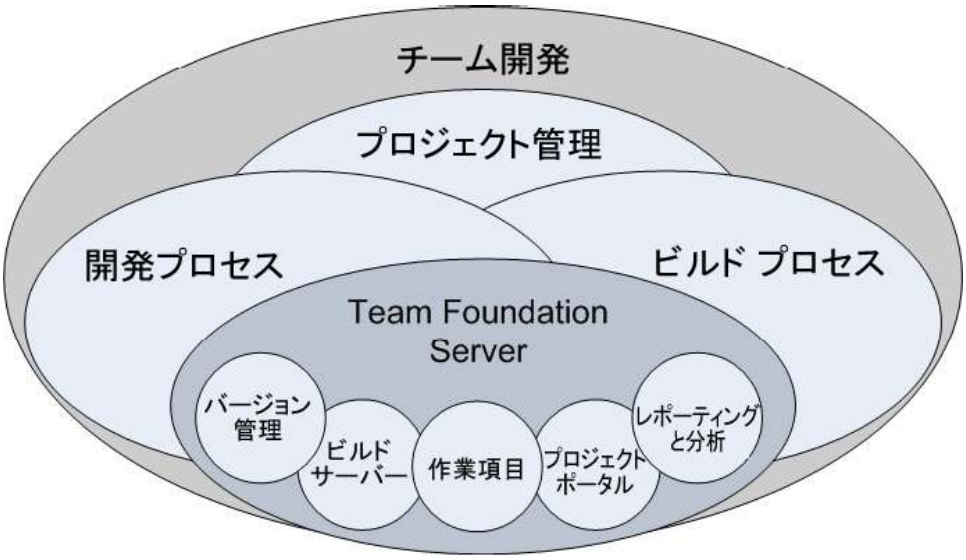
チーム開発

チームベースのソフトウェア開発プロジェクトを成功させるには、以下のような多くの要素、プロセス、

および役割を組み合わせますが、このガイドでは以下のものを中心に取り上げています。

- 開発プロセス
- ビルド プロセス
- プロジェクト管理プロセス

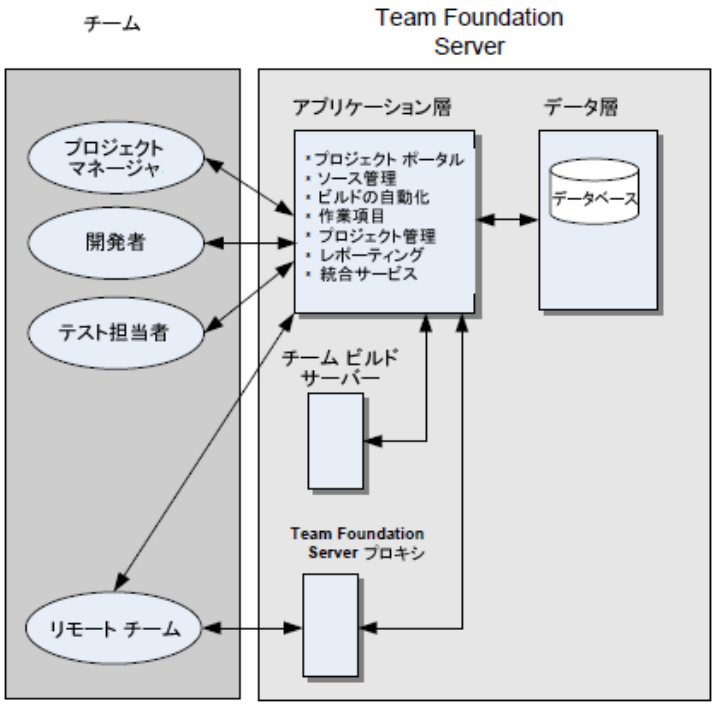
次の図は、チーム開発に関連する一般的なソフトウェア開発のプロセスと、これらの取り組みに対して水平方向の基本的サポートを提供するために Team Foundation Server をどのように利用するか、という関係を示しています。



ガイドの対象範囲

このガイドは、Team Foundation Server の配置、およびソース管理、ビルドの自動化、作業項目の管理、およびプロセス管理に対する Team Foundation Server の効率的な使用について焦点を当てています。

次の図は、Team Foundation Server がソフトウェア エンジニアリングおよび開発ライフサイクルにとって最も一般的な役割に関連している場合の、論理的な実装例を示しています。



このガイドを作成した理由

Team Foundation Server での自身の経験、そして現場で働くの従業員とお客様との会話を通じて、実社会で Team Foundation がどのように使用されているかを紹介するガイドが必要であると考えました。製品のドキュメント、ブログの投稿やフォーラムの中に情報はありますが、実社会の制約がある開発プロジェクトにおいて、これさえ見れば、Team Foundation Server を効率よく使用するための実証されたプラクティスが見つけられる、というものはありませんでした。

このガイドの対象読者

このガイドは、効率のよいチーム開発環境を作成するためのリソース、パターンおよびプラクティスを使用したソフトウェア開発プロセスに関係する個人に提供することを対象としています。以下の任務に従事している方は、このガイダンスを読んでメリットが得られるでしょう。

- Team Foundation を採用したいと考えている開発チーム
- プロジェクトおよび開発作業を管理すること、ソフトウェア開発イニシアティブの状況を提供すること、およびビジネス上の関係者にフィードバックを提供することについて、Team Foundation を

最大限に利用しようと考えているプロジェクト マネージャ
- Team Foundation の使用について調べているが、開発シナリオおよびチームの制約について Team Foundation がどのくらい役に立つかわからない関係者
- Team Foundation の配置およびインストールを計画している担当者

このガイドの参照の仕方

このガイドは、ほとんどのチームが Team Foundation について検討し、採用するときの順序に基づいて、いくつかの部分に分かれています。Team Foundation の採用を検討しているのであれば、最初から最後までガイド全体を読みたいと思われるでしょう。ソース管理やチーム ビルドなど、特定の用途に関する Team Foundation の使用を知りたい場合は、そのセクションだけを読むこともできます。コンセプトおよびガイドの方針を知りたい場合は、メインの章を使用します。実装の詳細を調べる場合は、付録の「ガイドライン」、「プラクティス」、「HOWTO」および「質問と回答」を読んでください。このように分けることによって、最初にトピックを理解し、それが適当であると思われたときに詳しく調べることができます。

このガイドの構成

このガイドは、最初から最後まで読むことも、仕事で必要な章だけを読むこともできます。

部

このガイドは、以下の 9 つの部に分かれています。
- 第 1 部 基本事項
- 第 2 部 ソース管理
- 第 3 部 ビルド
- 第 4 部 大規模なプロジェクトでの考慮事項
- 第 5 部 プロジェクト管理
- 第 6 部 プロセス テンプレート
- 第 7 部 レポーティング
- 第 8 部 チーム環境のセットアップと保守
- 第 9 部 Visual Studio 2008 Team Foundation Server

第 1 部 基本事項

- 第 1 章 チーム環境の紹介
- 第 2 章 Team Foundation Server のアーキテクチャ

第 2 部 ソース管理

- 第 3 章 ソース管理におけるプロジェクトとソリューションの構造化
- 第 4 章 Team Foundation のソース管理におけるプロジェクトとソリューションの構造化
- 第 5 章 分岐とマージの方針の定義
- 第 6 章 Visual Studio Team System におけるソース管理の依存関係の管理

第 3 部 ビルド

- 第 7 章 チーム ビルドの説明
- 第 8 章 チーム ビルドでの継続的な統合のセットアップ
- 第 9 章 チーム ビルドにおけるスケジュール ビルドのセットアップ

第 4 部 大規模なプロジェクトでの考慮事項

- 第 10 章 大規模なプロジェクトでの考慮事項

第 5 部 プロジェクト管理

- 第 11 章 プロジェクト管理の説明
- 第 12 章 作業項目の説明

第 6 部 プロセス テンプレート

- 第 13 章 プロセス テンプレートの説明
- 第 14 章 アジャイル ソフトウェア開発プロジェクトに対する MSF

第 7 部 レポーティング

- 第 15 章 レポーティングの説明

第 8 部 チーム環境のセットアップと保守

- 第 16 章 Team Foundation Server の配置
- 第 17 章 Team Foundation Server に対するインターネット アクセスの提供

第 9 部 Visual Studio 2008 Team Foundation Server

- 第 18 章 Visual Studio 2008 Team Foundation Server の新機能

ガイドライン

- ガイドライン: チーム ビルド
- ガイドライン: ソース管理
- ガイドライン: レポーティング
- ガイドライン: プロジェクト管理

プラクティス

- プラクティス一覧: チーム ビルド
- プラクティス一覧: ソース管理
- プラクティス一覧: レポーティング
- プラクティス一覧: プロジェクト管理

質問と回答

- 質問と回答: Team Foundation Server のソース管理とバージョニング

方法

- Visual Studio Team Foundation Server でプロジェクトに新しい開発者を追加する方法
- Visual Studio Team Foundation Server でチーム ビルドを使用してコード分析を自動的に実行する方法
- Visual Studio Team Foundation Server のカスタム レポートを作成する方法

- Visual Studio Team Foundation Server のリスク オーバー タイム レポートを作成する方法
- Visual Studio Team Foundation Server のカスタム チェックインを作成する方法
- Visual Studio Team Foundation Server でソース ツリーを作成する方法
- Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法
- Visual Studio Team Foundation Server でレポートをカスタマイズする方法
- Visual Studio Team Foundation Server でプロジェクトを管理する方法
- Visual Source Safe から Visual Studio Team Foundation Server へソース コードを移行する方法Visual Studio Team Foundation Server で基点のないマージを実行する方法
- Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法
- Visual Studio Team Foundation Server でスケジュール済みのビルドをセットアップする方法
- Visual Studio Team Foundation Server で ASP.NET アプリケーションを構造化する方法
- Visual Studio Team Foundation Server で Windows アプリケーションを構造化する方法
- Visual Studio Team Foundation Server でソース管理フォルダを構造化する方法

リソース

- Team Foundation Server のリソース

フィードバックとサポート

このガイドおよびそれに付随する内容の正確さを保証するために、あらゆる努力を行ってきました。

ガイドのフィードバック

このガイドに関するご意見は、以下のアドレスへ電子メールでご連絡ください。

TFSguide@microsoft.com.

特に以下の内容についてのフィードバックをお待ちしております。

- 推奨事項に特有の技術的な問題
- 実用性および有用性の問題

テクニカル サポート

マイクロソフト製品、およびこのガイドに記載されているテクノロジに関するテクニカル サポートは、Microsoft Product Support Services (PSS) により提供されます。製品サポートの情報については、http://support.microsoft.com の Microsoft Product Support の Web サイトを参照してください。

コミュニティ サポート

MSDN ニュースグループ:

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

フォーラム	アドレス
Team Foundation Server – 一般	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=22&SiteID=1
Team Foundation Server – セットアップ	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=68&SiteID=1
Team Foundation Server – 管理	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=477&SiteID=1
Team Foundation Server – ビルドの自動化	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=481&SiteID=1
Team Foundation Server – Power Toy	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1
Team Foundation Server – プロセス テンプレート	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=482&SiteID=1

Team Foundation Server – レポーティング & ウェアハウス	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=480&SiteID=1
Team Foundation Server – Team System の Web アクセス	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=1466&SiteID=1
Team Foundation Server – バージョン管理	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=478&SiteID=1
Team Foundation Server – 作業項目の追跡	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=479&SiteID=1

このガイドを作成したチーム

このガイドは、以下のチームのメンバによって作成されました。

- **J.D. Meier**
- **Jason Taylor**
- **Alex Mackman**
- **Prashant Bansode**
- **Kevin Jones**

寄稿および校閲

- **外部の寄稿および校閲:** David P. Romig、Sr; Dennis Rea、Eugene Zakhareyev、Leon Langleyben、Martin Woodward、Michael Rummier、Miguel Mendoza 、Mike Fourie、Quang Tran、Sarit Tamir、Tushar More、Vaughn Hughes
- **社内の寄稿および校閲:** Aaron Hallberg、Ahmed Salijee、Ajay Sudan、Ajoy Krishnamoorthy、Alan Ridlehoover、Alik Levin、Ameya Bhatawdekar、Bijan Javidi、Bill Essary、Brett Keown、

Brian Harry、Brian Moor、Brian Keller、Buck Hodges、Burt Harris、Conor Morrison、David Caufield、David Lemphers、Doug Neumann、Edward Jezierski、Eric Blanchet、Eric Charran、Graham Barry、Gregg Boer、Janet Williams Hepler、Jeff Beehler、Jose Parra、Julie MacAller、Ken Perilman、Lenny Fenster、Marc Kuperstein、Mario Rodriguez、Matthew Mitrik、Michael Puleio、Nobuyuki Akama、Paul Goring、Pete Coupland、Peter Provost、Granville (Randy) Miller、Rob Caron、Robert Horvick、Rohit Sharma、Ryley Taketa、Sajee Mathew、Siddharth Bhatia、Tom Hollander、Tom Marsh、Venky Veeraraghavan

成功体験募集

このガイドがお役に立った際は、お知らせください。直面した問題、およびこのガイドがどのように役に立ったかを簡単にお書きのうえ、*MyStory@Microsoft.com* のアドレスまでお送りください。

第 1 部

基本事項

第 1 部の内容

□チーム環境の紹介

□Team Foundation Server のアーキテクチャ

第 1 章 チーム環境の紹介

目的

• Microsoft® Visual Studio® Team Foundation Server がソフトウェア開発のライフサイクルをどのようにサポートするかを説明します。

• 一般的な開発チームが Team Foundation Server をどのように使用するかを説明します。

• 一般的なテスト チームが Team Foundation Server をどのように使用するかを説明します。

• 開発およびテスト チームの物理的な環境を説明します。

概要

この章では、Team Foundation Server (TFS) および Microsoft Visual Studio Team System (VSTS) がチームベースのソフトウェア開発環境でどのように使用されるかについて説明します。ここでは、TFS および VSTS の中心機能を紹介し、ソフトウェア開発プロジェクトにおける開発チームとテストチーム間のワークフローについて説明します。TFS ではソース管理、作業の追跡、レポーティング、プロジェクト管理および自動ビルド プロセスを統合するため、開発チームが協力してより効率よく作

業することができます。

成功するチームベースのソフトウェア開発プロジェクトには、効率よい作業環境を実現するために、協力してスムーズに作業しなければならない多くのプロセスがあります。その中心となるプロセスには、次のものがあります。

- 開発
- テスト
- ビルド
- 配置
- リリース

この章では、開発チームとテスト チームで TFS を使用して実行できる一般的な機能について紹介し、チーム間で効率よいコラボレーションをサポートするために、TFS を使用してどのようにワークフローを管理できるかについて説明します。

この章の参照の仕方

この章を使用して、TFS が、ソフトウェア開発のライフサイクルをサポートするために、どのように設計されているかについて学習します。この章を読んで、TFS のワークフローについて、および TFS によってチームのコラボレーションをどのように改善できるかについても学習します。

TFS のアーキテクチャおよび TFS のコア コンポーネントについての詳細は、「第 2 章 Team Foundation Server のアーキテクチャ」を参照してください。

Team Foundation Server の論理ワークフロー

TFS を使用すると、開発チームは、一括管理されているソース コード リポジトリにコードを保存することができます。ビルド サーバーを使用して、このリポジトリからビルドを作成し、これらのビルドをテスト チームに配布できます。

図 1.1 は、TFS の論理ワークフローについて、および開発とテストの環境がどのようにつながっているかについて示しています。

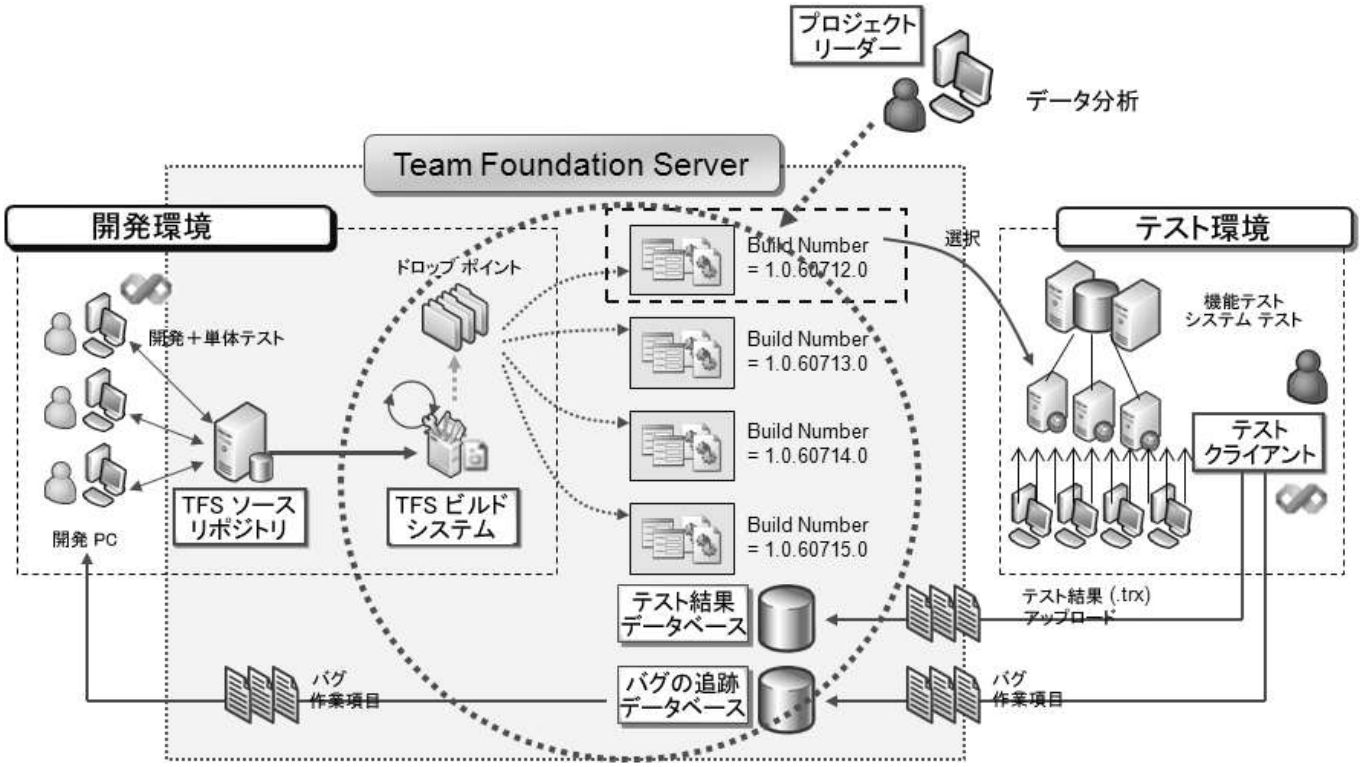


図 *1.1* *Team Foundation Server* の論理ワークフロー

テスト チームは、ドロップの場所からビルドを取得し、手動テストと自動テストの組み合わせを実行することにより、これらのビルドをテスト環境で実行します。テスト結果は TFS に保存され、ビルドの質に関するフィードバックを提供するときに使用します。テスト チームは、開発チームが対処しなければならないものについて、作業項目およびバグ (作業項目の特別なタイプ) を作成することもできます。これらの作業項目を使用してテスト チームは、開発チームの作業を追跡することができます。

開発、テストおよび本番環境の論理ワークフロー

複数の開発チームが作業している大規模な組織では、それぞれの開発チームが別の TFS を保持しており、その中には別のソース コード リポジトリおよびチーム ビルド サーバーが含まれています。図

1.2 は、統合テスト チームに対してアプリケーション ビルドを供給している 2 つの開発チームから始まるワークフローの例を示しています。

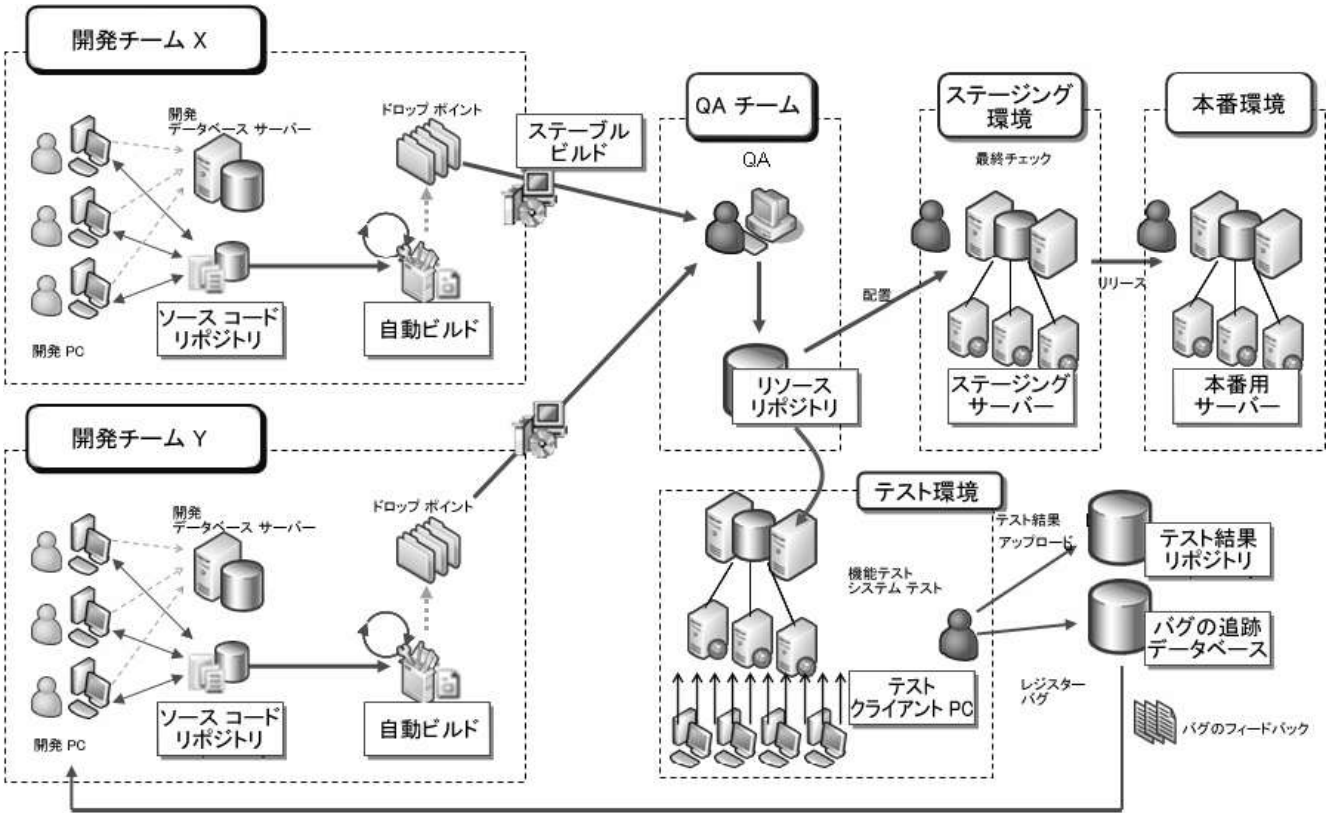


図 *1.2* *2*つの開発チームと *1* つの統合テスト チームを表す論理ワークフロー

各開発チームは、スケジュールされたビルドを、ネットワーク共有などのドロップ ポイントへ供給します。これらのビルドはテスト チームが取得し、ビルドの質を評価するためにテストします。テストのクオリティ ゲート (審査基準) に合格すると、アプリケーションは、最終的に本番用サーバーへ配置される前に、最終チェックおよびユーザー承認のためにステージング サーバーへ配置されます。

開発プロセス

開発者は、ソフトウェア開発プロジェクトの期間を通じて、TFS と多数の重要なやりとりをします。たとえば、開発者として TFS を以下のように使用することができます。

- TFS からバグおよびタスクの作業項目へアクセスし、どのような作業をする必要があるかを判断し

ます。たとえば、作業項目はプロジェクト マネージャ、別の開発者、またはテスト チームによって割り当てられた可能性があります。

- VSTS のソース管理エクスプローラを使用して TFS のソース管理リポジトリへアクセスし、最新のソース コードをローカルなワークスペース、または自身の開発コンピュータへ取り出します。
- 作業項目で特定される作業を実行した後で、コードをチェックインしてソース管理データベースへ戻します。
- 設定によってはチェックイン イベントが、チーム ビルドを使用した継続的な統合ビルドをトリガします。
- ビルドが失敗すると、ビルドの不具合を追跡するために新しい作業項目が作成されます。

テスト プロセス

テスト チームのメンバとして、TFS を以下のように使用することができます。

- 特定のドロップ場所から、スケジュールされたビルドの結果を取得します。
- さまざまな VSTS ツールを使用したセキュリティ テスト、パフォーマンス テスト、Web テストなど、手動および自動のテストを実行します。
- 後で参照するために、テストの結果を、TFS のテスト結果データベースへアップロードします。
- テストで特定されたバグを、新しい作業項目として TFS へ記録します。
- 最新のビルドによって、以前に記録したバグが修正された場合に、既存のバグを解決します。

開発およびテストの物理的な環境

開発環境とテスト環境に関連付けられているコンピュータの規模および数は、チームとプロジェクトの規模によって異なります。図 1.3 は、一般的な開発およびテストの物理的な環境を示しています。

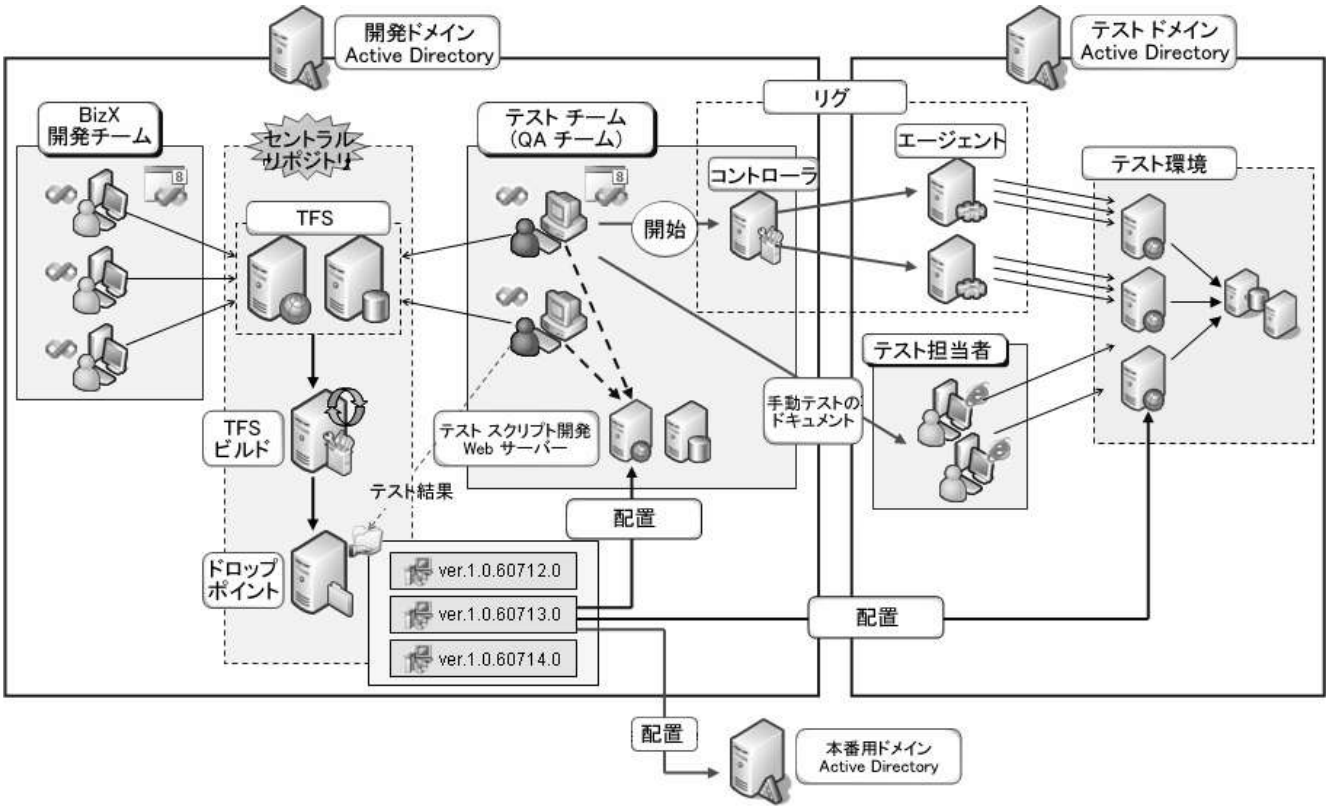


図 *1.3* 開発およびテストの物理的な環境

開発環境

開発環境は、開発およびビルドのプロセスをサポートします。開発環境には、以下のコンピュータが含まれています。

- Team Foundation Server
- ビルド サーバー
- ビルド サーバーからのドロップを保存しておくサーバー
- 開発者のワークステーション

開発チームが TFS にリモートでアクセスする場合、または特に規模の大きいチームがあり、それによりセントラル TFS サーバーでパフォーマンスの問題が生じる場合には、TFS プロキシを設定してパフォーマンスの向上を支援することもできます。

テスト環境

テスト環境は、Visual Studio Team Edition for Software Testers がインストールされている 1 台以上のテスト ワークステーションで構成されます。これを使用してテストのライフサイクルを管理し、機能テスト、システム テスト、セキュリティ テスト、パフォーマンス テストおよび Web テストを実行します。チームのメンバは TFS を使用して作業項目、バグおよびテスト結果を管理します。

テスト環境には、パフォーマンステスト用の Visual Studio Team Test Load が含まれることもあります。

まとめ

VSTS および TFS は、ソース管理、作業の追跡、レポーティング、プロジェクト管理、自動のビルドプロセスなど、ソフトウェア開発のさまざまな局面を統合することによってソフトウェア開発のライフサイクルをサポートする目的で設計されています。
TFS は、テスト チームと開発チーム間のコラボレーションにおいて重要な役割を果たします。開発チームは、開発サイクルを通じて TFS を使用し、バグおよび作業項目へアクセスして、どのような作業を行う必要があるかを判断したり、ソース管理へアクセスして開発を行ったりします。テスト チームは TFS を使用し、テストの実行、テスト結果のアップロード、およびバグの記録を行います。

参考資料

- TFS の基本事項に関する詳細については、http://msdn2.microsoft.com/en-us/library/ms364062(vs.80).aspx の「Team Foundation Server Fundamentals: A Look at the Capabilities and Architecture」を参照してください。
- Team Foundation の概要は、Microsoft MSDN® Web サイトの http://msdn2.microsoft.com/ja-jp/library/ms181232(vs.80).aspx で Team Foundation の製品のドキュメントを参照してください。

第 2 章 Team Foundation Server のアーキテクチャ

目的

- Microsoft® Visual Studio® Team System (VSTS) および Team Foundation Server (TFS) のアーキテクチャについて説明する。
- クライアント、アプリケーションおよびデータ層を構成するコンポーネントを特定する。
- シングルサーバー配置とマルチサーバー配置の違いを明確にする。

概要

この章では、TFS のアーキテクチャについて、および基本的な配置のトポロジについて紹介します。TFS には論理的な 3 層のアーキテクチャがあり、この中にはクライアント層、アプリケーション層およびデータ層が含まれています。TFS のクライアントはさまざまな Web サービスを介してアプリケーション層とやりとりし、アプリケーション層は、データ層のさまざまな Microsoft SQL Server™ データベースを使用します。

アプリケーション層とデータ層は、同じ物理サーバーへインストールするか、または別のサーバーへインストールするかを選択することができます。この選択は、主にチームの規模によって決まります。シングルサーバー配置は、メンバが 50 人以下のチームで最適に機能しますが、十分な能力を備えた強力なサーバーは最大 400 人のユーザーをサポートできます。デュアルサーバー配置では、約 2,000 人のユーザーまで拡張できます。

この章の参照の仕方

この章を使用して、TFS の中心となるコンポーネントについて学習し、これらのコンポーネントがどのように相互作用するかを学習します。この章を読むと、これらの各コンポーネントの目的について、およびこれらのコンポーネントが最も一般的にはどのように配置されるかについてもわかります。

TFS を初めて使用する場合は、最初に「第 1 章 チーム環境の紹介」を読んで開発チームとテスト チームがどのように TFS を使用するのか、また TFS を使用して、ソフトウェア開発作業のコラボレー

ションおよび全体の効率をどのように改善するのかについて学習します。

Team Foundation Server のアーキテクチャ

TFS には論理的な 3 層のアーキテクチャがあり、クライアント層、アプリケーション層、およびデータ層が含まれています。TFS のクライアントはさまざまな Web サービスを介してアプリケーション層とやりとりします。アプリケーション層は、データ層のさまざまなデータベースでサポートされています。図 2.1 は、TFS の各層のコンポーネントおよびその相互作用について示しています。

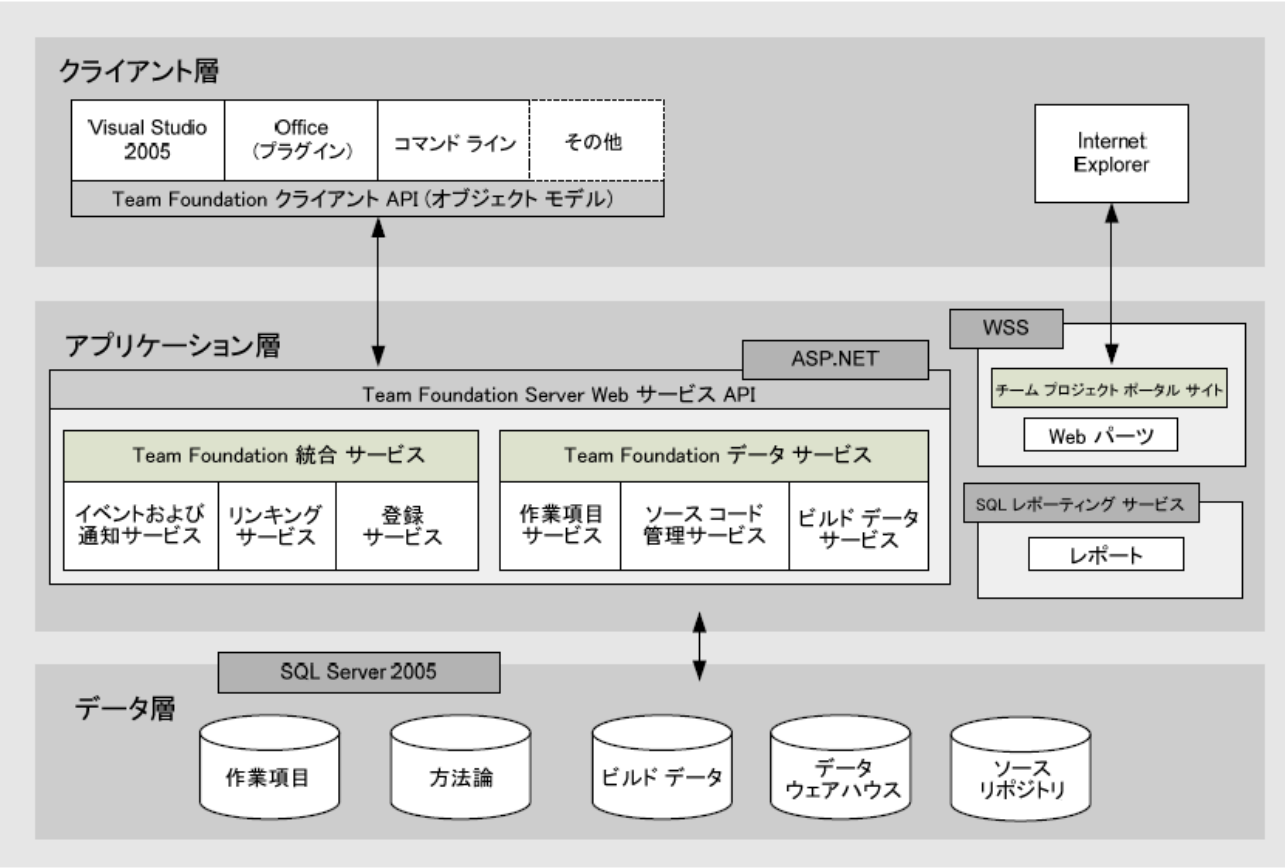


図 *2.1* *TFS* のコンポーネントと層

クライアント層

クライアント層には、以下の重要なコンポーネントが含まれています。

- **Team Foundation Server のオブジェクト モデル。**これは、TFS とのやりとりで使用されるパ

ブリック API です。このオブジェクト モデルを使用して、TFS とやりとりする独自のクライアント側アプリケーションを作成することができます。

- **Visual Studio Industry Partners (VSIP) のコンポーネント。**これらのコンポーネントは、Visual Studio で使用するためのサードパーティ ツール、アドイン、および言語です。
- **Microsoft Office 統合。**これは、Microsoft Office Excel® および Microsoft Office Project のアドイン セットで構成されています。これを使用して、TFS 作業項目追跡データベース内の作業項目を問い合わせおよび更新することができます。これは、これらのツールを既に広範に使用しているプロジェクト マネージャにとって特に有用です。
- **コマンドライン ツール。**これは、コマンドラインから TFS を使用できるようにするツールです。これらのほとんどのツールはソース管理機能を提供しますが、これは繰り返しのタスクを自動化したり、タスクをスケジュールする場合に便利です。
- **チェックイン ポリシー フレームワーク。**これはチェックイン ポリシー機能をサポートします。この機能は、チェックイン プロセスの間にコードの妥当性を検証することができる拡張可能なメカニズムです。

アプリケーション層

アプリケーション層は、クライアント層がアクセスする以下の ASP.NET Web サービスを提示します。これらの Web サービスは、サードパーティのインテグレータがプログラムすることは意図していませんが、ここでは完全性のために記載しています。Web サービスは、以下のコレクションに分類されます。

- Team Foundation データ サービス
- Team Foundation 統合サービス

Team Foundation データ サービス

この Web サービス セットは、主にデータ層のデータの操作に関係するものです。具体的には次のサービスがあります。

- **バージョン管理 Web サービス。**クライアント層はこの Web サービスを使用してさまざまな

TFS ソース管理機能を実行し、ソース管理のデータベースとやりとりします。

- **作業項目の追跡 Web サービス。**クライアント層はこの Web サービスを使用して、作業項目の追跡データベース内の作業項目を作成、更新および問い合わせします。
- **Team Foundation ビルド Web サービス。**クライアント層および MSBuild フレームワークはこの Web サービスを使用してビルド プロセスを実行します。

Team Foundation 統合サービス

この Web サービス セットは、統合および自動化の機能を提供します。これらのサービスは、データ層とはやりとりしません。Team Foundation 統合サービスには、次のものが含まれています。

- **登録 Web サービス。**このサービスは、他のさまざまな TFS サービスを登録するときに使用します。このサービスでは、情報を登録データベースに保持します。サービスはこの情報を使用して、互いにどのようにやりとりするかを検出および特定します。
- **セキュリティ Web サービス。**このサービスは、グループ セキュリティ サービスと承認サービスで構成されます。グループ セキュリティ サービスを使用して、すべての TFS ユーザーおよびグループを管理します。承認サービスは、TFS に対してアクセス コントロール システムを提供します。
- **リンキング Web サービス。**このサービスにより、ツールは、保持しているデータ要素間において緩やかに結合した関係 (リンク) を確立することができます。たとえば TFS は、欠陥作業項目と、欠陥を修正するために変更したソース コードの間の関係を、リンクを使用して保持します。
- **イベンティング Web サービス。**このサービスにより、ツールまたはサービスはイベントの種類を登録できます。ユーザーはこれらのイベントにサブスクライブし、電子メール、または Web サービスの起動によって通知を受け取ります。たとえば、チェックイン イベントを使用して、継続的な統合ビルドをトリガすることができます。
- **分類 Web サービス。**このサービスは、リンキング Web サービスと共に作用し、あらかじめ定義されている分類法に従って、TFS の成果物を分類することができます。これにより、データを整理するために共通の分類法を使用していない成果物に対しても、クロスツール レポーティングをサポートします。たとえば、作業項目がチームによって正常に整理されており、テストはコンポーネントによって正常に整理されている場合に、テストもチームによって整理して、作業項目と並べてレポートすることができます。

データ層

TFS は、データ層に保存されているデータに対する、クライアント アプリケーションからの直接アクセスはサポートしていません。代わりに、データに対するすべての要求は、必ずアプリケーション層の Web サービスを介して実施されます。TFS のデータ層は、アプリケーション層のデータ サービスに対応する以下のデータ ストアから構成されています。

- **作業項目の追跡。**作業項目に関するすべてのデータを保存します。
- **バージョン管理。**ソース管理に関するすべてのデータを保存します。
- **Team Foundation ビルド。**TFS のチーム ビルド機能に関するすべての情報を保存します。
- **レポーティング ウェアハウス。**TFS のすべてのツールおよび機能に関する情報を保存します。レポーティング ウェアハウスによって、複数のツールのデータを組み合わせるレポートの作成が簡潔になります。

配置のトポロジ

シングルサーバーのインストレーションから、より複雑な複数サーバーのトポロジまで、さまざまなトポロジを使用して TFS を配置することができます。どのトポロジを使用する場合でも、重要な要件に注意する必要があります。

重要な要件

選択した配置のトポロジに関係なく、次のようにします。

- アプリケーション層とデータ層は同じドメインの中にインストールする必要があります。ただし、これらの 2 つの層は同じサーバー ノードにあっても、別のサーバー ノードにあってもかまいません。
- Microsoft Windows Server™ 2003 Service Pack 1 (SP1) 以降が搭載されているコンピュータに TFS をインストールする必要があります。
- すべての TFS アプリケーション層の Web サービスを同じサーバーへインストールする必要があ

ります。

- 物理的な 1 台のコンピュータには 1 つの TFS インスタンスをインストールします。
- 1 台の物理サーバーに対して、TFS の複数のインスタンスをインストールすることはできません。
- 複数のデータベース サーバーにまたがって TFS のデータベースを分散することはできません。すべてのプロジェクトは、同じ Team Foundation のサーバー グループ上になければなりません。また複数のグループ間に配置することはできません。
- 既存の Microsoft SharePoint® Portal Server のインフラストラクチャを使用して、チーム プロジェクト ポータルをホストすることはできません。TFS SharePoint ポータルをホストするための専用サーバーを使用することを検討してください。
- ドメイン コントローラとして構成されているサーバーに TFS をインストールしてはいけません。これはサポートされていないためです。
- デュアルサーバー配置の場合は、TFS サービスを実行する場合に使用するいくつかのドメイン アカウントを準備する必要があります。たとえば、DOMAIN¥TFSSERVICE や DOMAIN¥TFSREPORTS などのアカウントを作成します。

シングルサーバー配置

シングルサーバー配置は最も簡単なトポロジで、ユーザーが 400 人以下の開発チームやパイロット プロジェクトに適しています。このアプローチでは、アプリケーション層およびデータ層のすべてのコンポーネントを 1 台のサーバー上にインストールし、同じドメインからそれらにアクセスします。

パフォーマンス テスト用にテスト リグ コンポーネントをインストールする必要がある場合は、これらのコンポーネントをサーバー ノードにインストールすることも、複数のクライアント上にインストールすることもできます。図 2.2 は、シングルサーバーのトポロジを示しています。

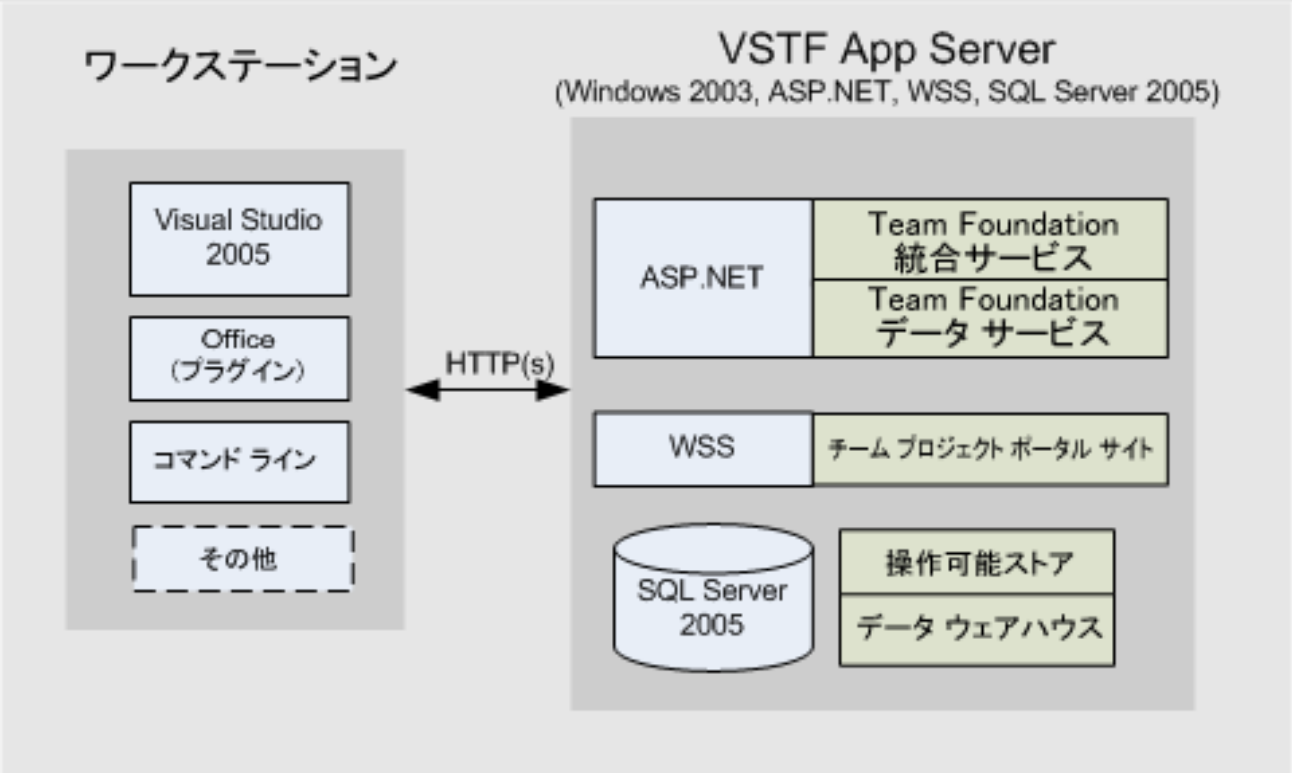


図 *2.2* シングル サーバーのトポロジ

デュアルサーバー配置

デュアルサーバー配置トポロジは、ユーザーが 2,000 人程度の大規模な開発チームにとって有用です。この配置トポロジでは、アプリケーション層を、データ層とは別のサーバー ノードにインストールします。

Team Foundation ビルド サービスはアプリケーション層にインストールできますが、大規模なチームに対しては、1 つ以上の専用ビルド サーバーをセットアップすることを推奨します。プロジェクトでパフォーマンス テストが必要な場合は、追加のサーバー ノードに対してテスト リグ (コントローラとエージェント) を配置することができます。図 2.3 は、デュアルサーバーのトポロジを示しています。

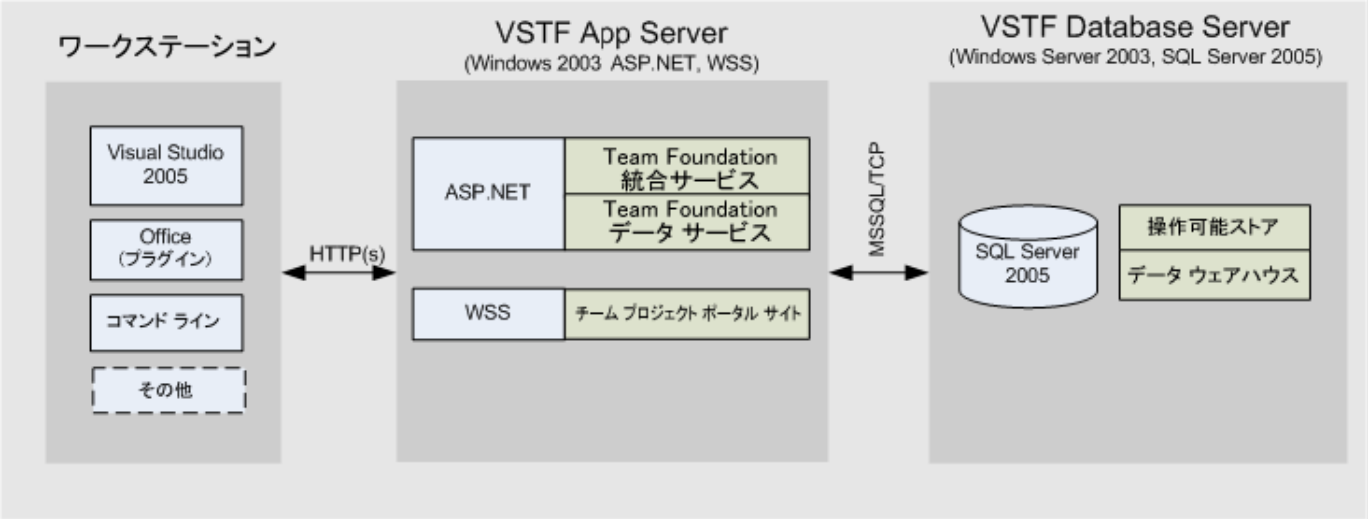


図 *2.3* デュアル サーバーのトポロジ

まとめ

Team Foundation Server のアーキテクチャは、クライアント層、アプリケーション層およびデータ層の 3 つの層で構成されています。

- クライアント層には、Visual Studio 2005 のチーム エクスプローラ、Microsoft Office 統合、コマンドライン ツールなどのクライアント コンポーネントが含まれています。
- アプリケーション層には、Team Foundation のバージョン管理サービス、作業項目の追跡サービス、ビルド サービスなどのコンポーネントが含まれています。
- データ層には、作業項目の追跡、バージョン管理、チーム ビルド、およびレポーティング ウェアハウスで必要なデータを格納するためのデータベースが含まれています。

TFS はシングルサーバー配置とデュアルサーバー配置のトポロジをサポートしています。シングルサーバー配置では、アプリケーション層とデータ層が同じマシンにインストールされます。この配置は、規模の小さいチームや、パイロット プロジェクトを実施する場合などに適しています。デュアルサーバー配置では、アプリケーション層とデータ層は別のサーバーにインストールされます。この配置は、ユーザーの人数を増やす必要がある、大規模なチームにとって有用です。

参考資料

- Team Foundation の基本事項の詳細については、http://msdn2.microsoft.com/en-us/library/ms364062(vs.80).aspx の「Team Foundation Server Fundamentals: A Look at the Capabilities and Architecture」を参照してください。
- Team Foundation の概要については、Microsoft MSDN® Web サイトの http://msdn2.microsoft.com/ja-jp/library/ms181232(vs.80).aspx で Team Foundation 製品のドキュメントを参照してください。
- Team Foundation Server の拡張性の制限の詳細については、http://blogs.msdn.com/bharry/archive/2006/01/04/509314.aspx の「Team Foundation Server Capacity Planning」を参照してください。

第 2 部

ソース管理

第 2 部の内容

□ソース管理におけるプロジェクトとソリューションの構造化

□Team Foundation のソース管理におけるプロジェクトとソリューションの構造化

□分岐とマージの方針の定義

□Visual Studio Team System におけるソース管理の依存関係の管理

第 3 章 ソース管理におけるプロジェクトとソリューションの構造化

目的

- Microsoft® Visual Studio® Team System のソリューションとプロジェクトを適切に構造化します。
- 複数ソリューションを使用する場合と、単一ソリューションを使用する場合について理解します。
- 小規模、中規模、および大規模のチームに対する適切な構造を特定します。

概要

この章では、Visual Studio のソリューションおよびプロジェクト ファイルを、チーム開発にふさわしい方法で構造化するさまざまなオプションについて説明します。Visual Studio はソリューション (.sln) ファイルを使用して、関連する Visual Studio のプロジェクト (.csproj および .vproj) ファ

イルをグループ化します。プロジェクトおよびソリューションをどのように構造化するかを決めることは重要です。これは、選択するパターンによって、多数の影響があるためです。たとえば、この決定は、開発チームのメンバがソース管理に対して、いかに簡単にソリューションとプロジェクトを挿入および取得できるか、また、依存関係を参照するために使用するメカニズムやビルド プロセスにも影響を与えます。

小規模なプロジェクトで作業している場合は、単一ソリューションを使用してすべてのプロジェクト ファイルを含めることができます。多数のプロジェクト ファイルを持つソフトウェア開発プロジェクトで作業している場合は、複数ソリューション ファイルを使用して、関連するプロジェクトをグループ化する必要があります。これらのプロジェクトは、チーム プロジェクト全体の一部の機能に相当します。特有のシナリオによっては、すべてのプロジェクト ファイルをグループ化するために単一ソリューションも必要な場合があります。

この章の参照の仕方

この章を使用して、Visual Studio のソリューションとプロジェクトを構造化するためのアプローチを選択します。この章を最大限に利用するには、次のようにします。

- **方針リストを使用します。**方針の最初のリスト (単一ソリューション、パーティション分割したソリューション、および複数ソリューション) を使用して、シナリオに対する最適なアプローチを簡単に評価します。
- **ニーズに最も関係のあるシナリオのセクションを読みます。**選択したオプションを実装する方法が記載されているセクションを読みます。
- **次に「第 4 章 Team Foundation Server のソース管理におけるプロジェクトとソリューションの構造化」を読みます。**第 4 章は、Team Foundation Server (TFS) のソース管理でコードを保存する場合に留意する必要がある重要な事項について説明します。
- **「第 6 章 Visual Studio Team System におけるソース管理の依存関係の管理」を読みます。**プロジェクトの構造は、複数のプロジェクトおよびソリューション間で依存関係を管理する場合に、使用できる方針に影響を与えます。依存関係を管理する方法の詳細については、第 6 章を参照してください。

- **このガイドの「方法」を読みます。**この章で説明しているさまざまな手順について順を追って理解するには、次の「方法」を読んでください。
    - Visual Studio Team Foundation Server で ASP.NET アプリケーションを構造化する方法
    - Visual Studio Team Foundation Server で Windows アプリケーションを構造化する方法
    - Visual Studio Team Foundation Server でソース管理フォルダを構造化する方法

ソリューションおよびプロジェクトの構造の方針

ソリューションおよびプロジェクト ファイルの構造化では、最も一般的な方針として次の 3 つが使用されます。

- **単一ソリューション。**小規模なシステムで作業している場合は、単一ソリューションを作成し、その中にすべてのプロジェクトを設定します。
- **パーティション分割したソリューション。**大規模なシステムで作業している場合は、複数ソリューションを使用して、関連するプロジェクトをグループ化します。開発者がひとまとめにして修正しそうなプロジェクトのサブセットを論理的にグループ化するソリューションを作成し、すべてのプロジェクトを含む 1 つのマスタ ソリューションを作成します。このアプローチによって、特定のプロジェクトで作業する必要がある場合に、ソース管理から取得しなければならないデータ量が減少します。
- **複数ソリューション。**数十個、またはそれ以上のプロジェクトを必要とする、非常に規模の大きいシステムで作業している場合は、サブシステムで作業するための複数ソリューションを使用します。ただし依存関係のマッピングおよびパフォーマンス上の理由から、すべてのプロジェクトを含む 1 つのマスタ ソリューションは作成しないでください。

一般的には次のようにします。

- 結果のソリューションが大きすぎて Visual Studio にロードできない場合を除いては、単一ソリューションの方針を使用します。
- アプリケーションのサブシステム上に特別なビューを作成するには、複数ソリューションを使用します。
- ソリューションをロードする時間、および開発者のビルド時間を短縮するには複数ソリューション

を使用します。

プロジェクトおよびソリューションの構造を設計する場合には、以下のことに留意する必要があります。

- 各プロジェクトは、ビルド時に 1 つのアセンブリを生成します。最初に、どのようなアセンブリを作成したいのかを決定し、次にこれに基づいて、どのようなプロジェクトが必要なのかを決定します。このことを使用して、コードベースをプロジェクトにどのように組み込むかを決定します。
- 最初に、最もシンプルな単一ソリューションの構造から始めて、実際に必要になったときに構造を複雑にします。
- マルチソリューションの構造を設計する場合:
  - プロジェクトの依存関係について考慮します。互いに依存関係にあるプロジェクトを、同じソリューションにグループ化します。これにより、ソリューション内でプロジェクト参照を使用することができます。ファイル参照の代わりにプロジェクト参照を使用すると、Visual Studio でビルドの構成 (デバッグ/リリース) の同期を保持することが可能になり、バージョン管理を追跡して、プロジェクトを再ビルドしなければならないタイミングを決定することができます。異なるソリューション間でのプロジェクト参照の数は、最少になるようにします。
  - ソースの共有について考慮します。同じソースを共有するプロジェクトは、同じソリューション内に設定します。
  - チームの構造について考慮します。関連するプロジェクト セットでのチームの作業が簡単になるよう、ソリューションを構造化します。
- ファイル システムまたはソース管理フォルダの構造を変更せずに、プロジェクトをソリューション内に簡単にグループ化できるよう、プロジェクト構造はフラットにします。

単一ソリューション

小規模なシステムで作業している場合は、単一の Visual Studio ソリューションを使用して、すべてのプロジェクトを含めることを検討します。この構造では、ソリューションを開いたときにすべてのコードが使用できるため、開発がシンプルになります。またこの方針では、すべての参照がソリューション内のプロジェクトにあるため、参照のセットアップが簡単になります。購入したコンポーネントなど、ソリューションの外部にあるサードパーティのアセンブリを参照するために、ファイル参照を使用しなければならない場合もあります。図 3.1 は、単一ソリューションのアプローチを示しています。

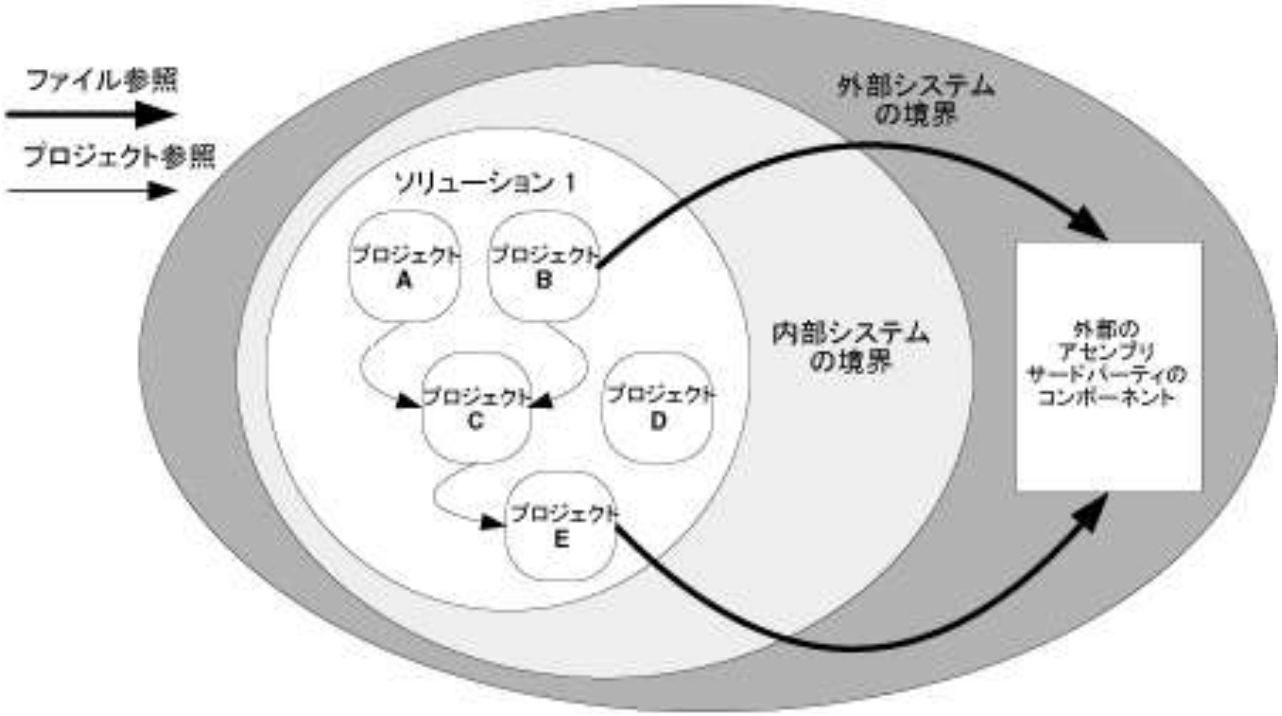


図 *3.1* 単一ソリューションのアプローチ

この構造を選択する主な理由は次のとおりです。

- ビルド スクリプトをシンプルにできる。
- ソリューション内のプロジェクト間で依存関係を簡単にマップできる。

すべての開発者が同じソリューションを使用しており、同じプロジェクト セットを所有している場合は、この構造を使用する必要があります。大規模なシステムで、サブシステムまたは機能によってプロジェクトを編成したい場合に、この構造を使用すると、問題が発生することがあります。

パーティション分割したソリューション

大規模なシステムで作業している場合は、複数ソリューションの使用を検討してください。複数ソリューションは、それぞれがアプリケーションのサブシステムを表します。開発者はこれらのソリューションを使用して、全プロジェクトのすべてのコードをロードせずに、システムの比較的小さい部分で作業することができます。依存関係を持つすべてのプロジェクトが一緒にグループ化されるように、ソリューションの構造を設計します。このようにすると、ファイル参照ではなくプロジェクト参照を使用する

ことができます。また、すべてのプロジェクトが含まれる 1 つのマスタ ソリューション ファイルを作成することも検討します。これを使用して、アプリケーション全体をビルドすることができます。

注意: Visual Studio の前バージョンとは異なり、Visual Studio 2005 は MSBuild をベースにしています。現在は、参照先のすべてのプロジェクトを含まないソリューション構造を作成し、エラーなしでビルドできるようになりました。マスタ ソリューションを最初にビルドし、各プロジェクトからのバイナリ出力を生成しておけば、MSBuild は、ソリューションおよびビルドの境界の外のプロジェクト参照を正常に追跡することができます。これはファイル参照ではなく、プロジェクト参照を使用している場合のみ機能します。Visual Studio のビルド コマンド ラインおよび IDE からこのようにしてソリューションを正常にビルドして作成することができますが、チーム ビルドでは既定ではビルドできません。チーム ビルドで正常にビルドするには、すべてのプロジェクトおよび依存関係が含まれているマスタ ソリューションを使用します。

図 3.2 は、パーティション分割したソリューションのアプローチを示しています。

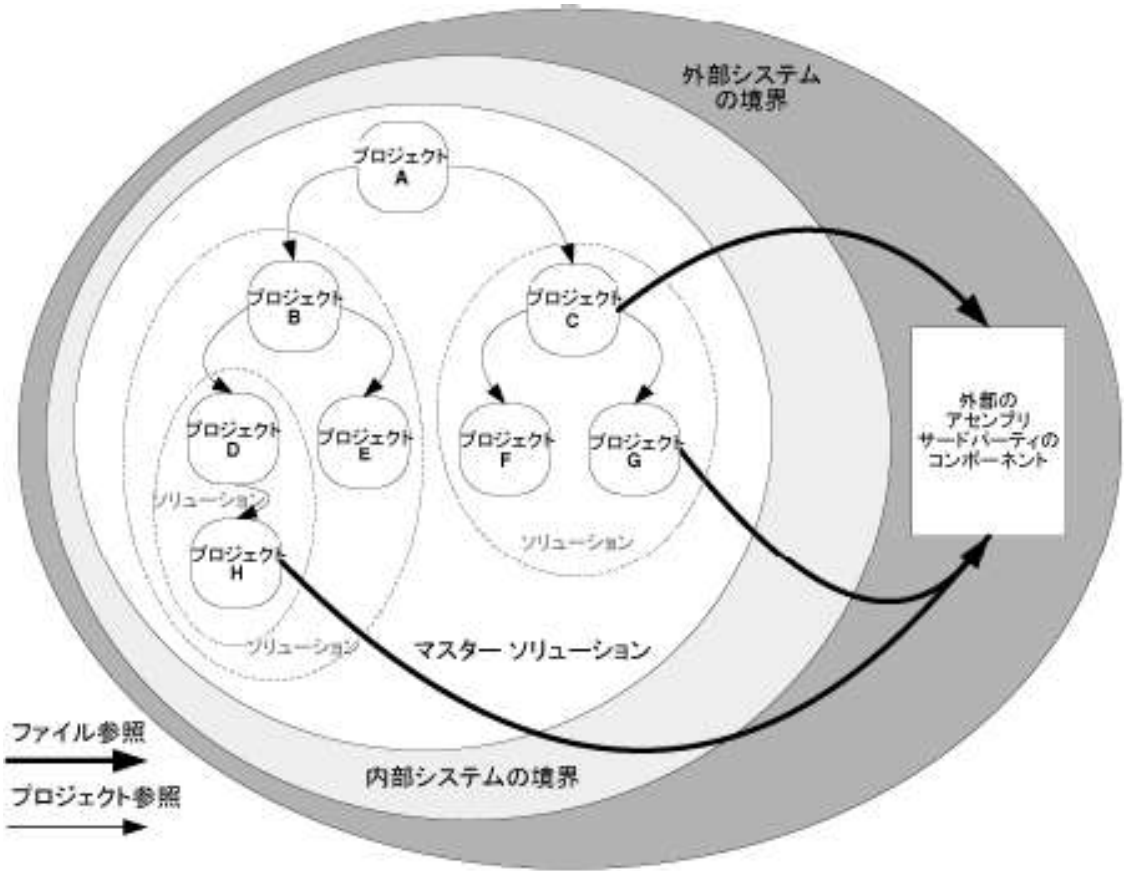


図 *3.2* パーティション分割したソリューションのアプローチ

複数ソリューションで作業している場合は、すべてのプロジェクトに対してフラット ファイル構造を使用します。一般的な例は、Microsoft Windows® フォーム プロジェクト、ASP.NET プロジェクト、Windows サービス、および一部またはすべてのプロジェクトで共有しているクラス ライブラリ プロジェクトのセットを備えているアプリケーションです。

すべてのプロジェクトに対して以下のフラット構造を使用できます。

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

構造をフラットにしておくと、プロジェクトのさまざまなビューを示すソリューションを使用するうえで、多大な柔軟性と機能性を実現できます。特に、別のソリューションからクラス ライブラリを再利用する必要がある場合は、ソリューション ファイルの周囲の物理的なフォルダ構造を変更するのが非常に困難です。

この構造を使用する理由は、次のとおりです。

- アプリケーションのサブソリューションをロードおよびビルドするときのパフォーマンスが改善される。
- 開発のサブチームまたはコード共有の境界をベースとして、サブソリューションを使用して、プロジェクト セットのビューを作成できる。
- マスタ ソリューションを使用してアプリケーション全体をビルドできる。

- 各サブソリューションのプロジェクト間で依存関係を簡単にマップできる。
- ソリューションが論理的に細分化されている場合は、全体の複雑さが軽減する。たとえば、テクノロジまたは機能の系列に従ってソリューションを細分化すると、新しい開発者は、どのソリューションで作業しているのかわかりやすくなります。

この構造を使用しない主な理由は次のとおりです。

- ソリューションのメンテナンス コストが増加する。新しいプロジェクトを追加すると、複数のソリューション ファイルの変更が必要になることがあります。

複数ソリューション

数十ものプロジェクトが必要な大規模なソリューションで作業している場合に、ソリューションの大きさの制限に直面することがあります。このような場合は、アプリケーションを複数のソリューションに細分化しますが、全体のアプリケーションに対して 1 つのマスタ ソリューションを作成しないようにします。これは、各ソリューション内のすべての参照がプロジェクト参照であるためです。各ソリューションの外部のプロジェクトに対する参照 (サードパーティのライブラリや、別のサブソリューションのプロジェクトなど) はファイル参照です。これは、このような場合には "マスタ" ソリューションがあり得ないことを意味しています。

代わりに、ソリューションをビルドする必要がある順序を示すスクリプトを使用する必要があります。複数ソリューションの構造に関するメンテナンス タスクの 1 つは、開発者がソリューション間で循環参照を不注意に作成していないことを確実にすることです。この構造では、複雑なビルド スクリプト、および依存関係の明示的なマッピングが必要です。この構造では、Visual Studio でアプリケーションをそのまま全部ビルドすることはできません。代わりに、TFS チーム ビルドまたは MSBuild を直接使用できます。

図 3.3 は、複数ソリューションのアプローチを示しています。

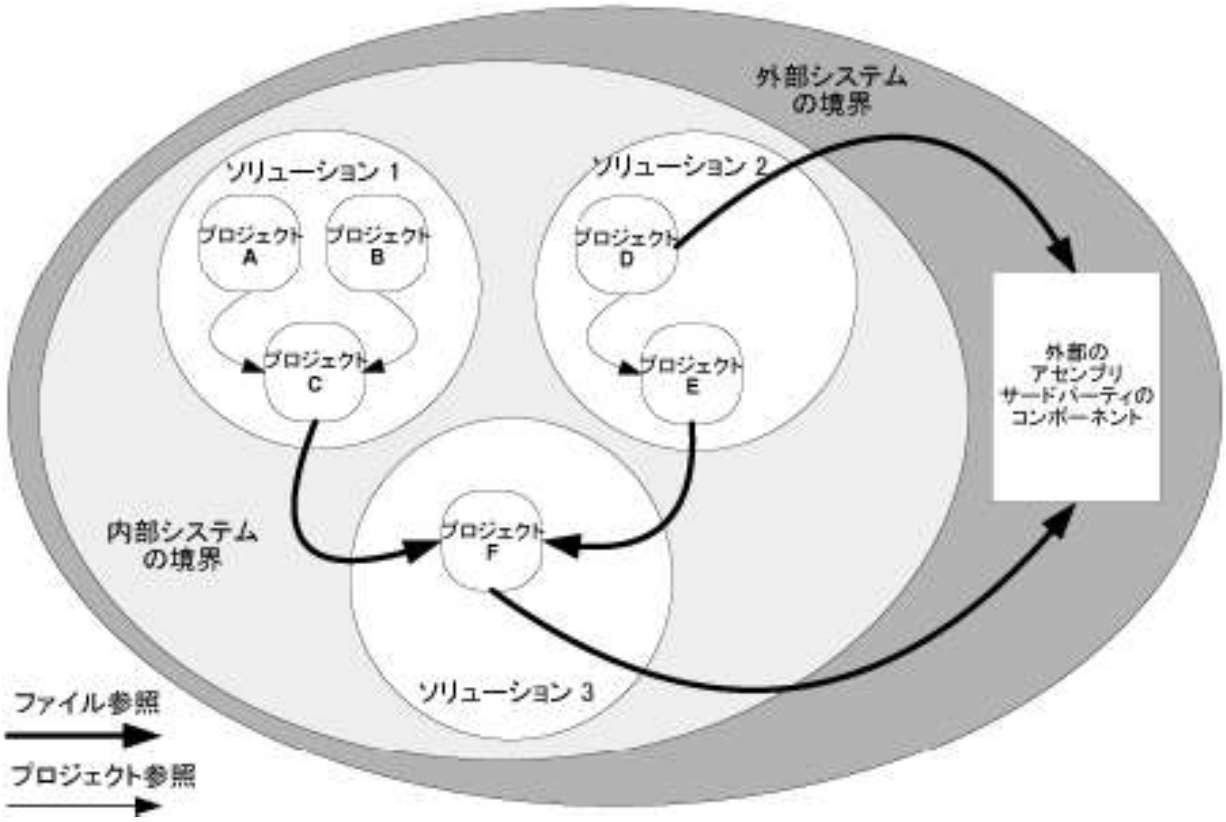


図 *3.3* 複数ソリューションのアプローチ

非常に規模の大きいアプリケーションに対して、Visual Studio IDE のパフォーマンスおよび大きさの制限を回避するには、この構造を使用する必要があります。

この構造を使用しない理由の 1 つは、ソリューションを正しい順序でビルドすることによって、サブソリューション間で依存関係を扱うための複雑なビルド スクリプトが必要になるためです。

大規模なプロジェクトでの考慮事項

大規模な開発チームは多くの場合、以下の点において規模の小さいチームとの違いが顕著になります。

- より複雑な分岐およびマージの構造が必要になる。
- 複数のソリューションおよびチーム プロジェクト間で依存関係を管理することが多くなる。
- コンポーネントおよびチームに対して複数のビルドを保持することが多くなる。

パーティション分割したアプローチは、最も規模の大きいプロジェクトで適切に機能します。このアプローチでは、アプリケーションのビルドに使用できる単一ソリューションを保持しながら、ソリューションの柔軟性が実現されるためです。アプリケーションの規模が大きく、ソリューションの大きさの制限を受けてしまう場合は、複数ソリューションのアプローチを使用します。

まとめ

ソースを個別のサブソリューションにパーティション分割する必要がない、小規模なプロジェクトに対しては単一ソリューションを使用します。

開発者がひとまとめにして修正しそうなプロジェクトのサブセットを論理的にグループ化するには、パーティション分割したソリューションを使用し、すべてのプロジェクトが含まれる 1 つのマスタ ソリューションを作成します。

サブシステム上に特別なビューを作成し、アプリケーションのロードおよびビルド時間を短縮するには、複数ソリューションを使用します。

パーティション分割したアプローチは、最も規模の大きいプロジェクトで適切に機能します。このアプローチでは、アプリケーションのビルドに使用できる単一ソリューションを保持しながら、ソリューションの柔軟性が実現されるためです。

参考資料

- プロジェクトおよびソリューションの構造の詳細 (Team Foundation Server には直接適用されません) については、http://msdn2.microsoft.com/en-us/library/ms998208.aspx の「Team Development with Visual Studio .NET and Visual SourceSafe」を参照してください。

第 4 章 Team Foundation のソース管理におけるプロジェクトとソリューシ

ョンの構造化

目的

- Microsoft® Visual Studio® Team Foundation Server (TFS) のソース管理における効率よいチーム開発のためにプロジェクトを構造化する。
- サーバー側およびクライアント側のフォルダ構造の同期を維持する。
- 単体テストの構造に関する方針を選択する。
- さまざまな分岐シナリオをサポートするフォルダ構造を作成する。
- ワークスペースとは何か、およびローカル ファイルをどのようにソース管理にマップするかについて学習する。
- どのファイルがソース管理に追加されるかを理解する。

概要

新しいソリューションおよびプロジェクトを作成するときに Visual Studio で使用されるデフォルトのフォルダ規則の多くは、チーム開発および TFS のソース管理での使用に対して最適化されていません。Visual Studio の新しいプロジェクトおよびソリューションを作成する場合には、デフォルトをそのまま使用せずに、ローカルおよびサーバー側のフォルダ構造について熟慮する必要があります。

この章では最初に、開発コンピュータ (クライアント側) のソリューションおよびプロジェクトをどのように構造化するか、および TFS のソース管理 (サーバー側) のフォルダをどのように構造化するかについて説明します。ここでは Microsoft Windows® フォーム、スマート クライアント、Web アプリケーションなど、さまざまなアプリケーションの構造に対してフォルダの例を示します。次に、ワークスペースをどのように使用して、クライアントとサーバーのフォルダ構造間のマッピングを管理するかについて説明します。

この章の参照の仕方

この章を使用して、さまざまな規模および複雑さを備えたチーム開発プロジェクトに適したフォルダ構造の例を学習します。この章を最大限に利用するには、次のようにします。

- **サーバー側の構造の提示を使用します。**提示されたサーバー側のフォルダ構造を使用して、TFS ソース管理内のプロジェクト ソース コードを編成します。
- **クライアント側の構造の提示を使用します。**提示されたクライアント側のフォルダ構造を使用して、ローカルな開発ワークスペース内のプロジェクト ソース コードを編成します。
- **このガイドの「HOWTO」を読みます。**次の方法では、この章で説明しているいくつかのプロセスについて順を追って示しています。
    - Team Foundation Server でソース ツリーを作成する方法
    - Team Foundation Server で ASP.NET アプリケーションを構造化する方法
    - Team Foundation Server で Windows アプリケーションを構造化する方法
    - Team Foundation Server でソース管理フォルダを構造化する方法

サーバー側の構造

ほとんどのチーム プロジェクトには、1 つ以上の Visual Studio ソリューションが含まれており、そのそれぞれに 1 つ以上の Visual Studio プロジェクトが含まれています。独立した開発パスをサポートするための分岐が必要な場合には、(クライアントとサーバーの両方で) **Main** というルート レベルのフォルダを使用して、Visual Studio のプロジェクトをグループ化します。以下に、TFS のソース管理内のフォルダ構造の例を示します。

$MyTeamProject1
/Main →ソリューション (.sln) ファイルを含めることが可能
/Source
/MyApp1 →□MyApp1.sln ファイルが含まれている
/Source →□すべてのソースのフォルダが含まれている
/ClassLibrary1 →□ClassLibrary1.csproj が含まれている
/MyApp1Web →□Default.aspx が含まれている
/UnitTests →□単体テスト用のフォルダが含まれている
/ClassLibrary1Tests →□テスト プロジェクトおよびコードが含まれている
/MyApp1WebTests →□テスト プロジェクトおよびコードが含まれている
/SharedBinaries →□共有しているバイナリ (ライブラリなど)
/SharedSource →□共有しているソース コード

/Docs →□製品ドキュメントが含まれている

/Tests →□テスト用のコンテナ

/FunctionalTests

/PerformanceTests

/SecurityTests

　/TeamBuildTypes →□チーム□ビルドで自動的に作成される

/BuildType1

/BuildType2

Main は、ソース ファイル、および関連する成果物 (ビルドのアウトプット、設計のドキュメント、テスト ケースなど) のコンテナフォルダです。アプリケーション フォルダ (前述の例の **MyApp1** など) には、Visual Studio の関連するプロジェクトをグループ化するための Visual Studio ソリューション (.sln) が含まれています。それぞれのプロジェクト ファイル (.vcproj または .vbproj) は、/Main/Source/MyApp1/Source にある専用のプロジェクト フォルダの中にあります。各ソース プロジェクトに付随している単体テストは、**UnitTests** フォルダの下にあります。追加の Visual Studio ソリューション (.sln) ファイルを **Main** フォルダ内に配置して、プロジェクトをグループ化したものをいくつかまとめて作業することができます。

Docs および **Test** フォルダは、製品のドキュメントや自動化されたテストなど、チーム プロジェクトに関連する他の成果物を保持するために使用します。

TeamBuildTypes フォルダは、最初にチーム ビルドを生成したときに自動的に作成されます。チーム ビルドの種類を手動でチェックインしたい場合は、このフォルダを手動で作成し、自身のチーム ビルド ファイルを追加します。このようにすると、TFS はこのフォルダを自動的に認識します。

プロジェクトのグループ化およびソリューションの構造の詳細については、「第 3 章 ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

単体テストの保存

次に示すように、**Source** と同じレベルの **UnitTests** というフォルダに、単体テストを保存することができます。

…

/MyApp1 →□MyApp1.sln ファイルが含まれている

/Source →□すべてのソースのフォルダが含まれている

/ClassLibrary1 →□ClassLibrary1.csproj が含まれている

/MyApp1Web →□Default.aspx が含まれている

/UnitTests →□単体テスト用のフォルダが含まれている

/ClassLibrary1Tests →□テスト プロジェクトおよびコードが含まれている

/MyApp1WebTests →□テスト プロジェクトおよびコードが含まれている

このシナリオでは、単体テストを "一級市民" として扱っていますが、プロジェクト レベルの分岐の適合性が犠牲になっています。次に、代わりの構造を示します。

/MyApp1

/Source

/ClassLibrary1

/ClassLibrary1Tests

/MyApp1Web

/MyApp1WebTests

各アプローチには次に示す利点と欠点があります。

Source フォルダと同レベルの単体テストフォルダ

- 利点 ： 単体テストが 1 か所にあります。
- 利点 ： 出荷するコードと出荷しないコードが区別されます。
- 利点 ： ビルド プロセスはすべてのプロジェクトにおいてすべての単体テストを簡単に実行できます。
- 欠点 ： 開発者は自身のプロジェクトのみで単体テストを実行するのは困難です。

- 欠点 ： ソースを分岐する場合には、単体テストは含まれません。

各プロジェクトの単体テスト

- 利点 ： 開発者は単一のプロジェクトで単体テストを簡単に実行できます。
- 利点 ： 分岐すると、分岐したフォルダには単体テストが含まれるため、各分岐のソースに対して単体テストが強固に関連付けられます。
- 欠点 ： ソース フォルダ内で出荷するコードと出荷しないコードが混在します。
- 欠点 ： 通常、すべてのプロジェクトでビルド時にすべての単体テストを同時に実行することは困難です。

ドキュメントの保存

Documentation フォルダは、製品に関連するドキュメントのための場所です。どのドキュメントを TFS のソース管理に保存するか、Microsoft Windows SharePoint® のチーム サイト上のドキュメント ライブラリに何を保存すればよいかを決定する場合は、以下のことについて考えるようにします。

- ユース ケース、シナリオ、要件のドキュメント、設計ドキュメントなどの内部のチーム ドキュメントに対して SharePoint を使用する。
- お客様に提供する、製品関連のドキュメントに対しては TFS のソース管理を使用する。これには、インストールおよび配置のガイド、操作ガイド、およびヘルプ ファイルを含めることもできます。

ほとんどのドキュメントはバイナリ ファイルのため、手動でマージしないよう排他的なロックの使用を検討します。このようにすると、ファイルを使用中のときに通知が行われ、手動マージをしなければならない状況を回避するうえで役に立ちます。

SharePoint を使用する場合には、ドキュメントのバージョンを厳密に管理する必要があるため、注意が必要です。TFS のソース管理に比べて、SharePoint では変更を簡単に上書きできます。SharePoint では、ファイルをアップロードするときに既定で [既存のファイルを上書きする] オプションが選択されています。

クライアント側の構造

開発ワークステーションのローカル フォルダは、サーバーのフォルダ構造と同じでなければなりません。すべてのチーム プロジェクトのすべてのソースは、C:¥DevProjects などの 1 つのルート フォルダの下にまとめて配置し、ソースの成果物をきちんと整理しておきます。次の例に示すように、各チーム プロジェクトに対して 1 つのサブフォルダを作成します。

C:¥DevProjects →□すべてのチーム プロジェクトに対するルート コンテナ
¥MyTeamProject1 →□TeamProject1 のコンテナ フォルダ
¥MyTeamProject2 →□TeamProject2 のコンテナ フォルダ

各チーム プロジェクトのフォルダの下では、次の例で示されているように、ソース管理サーバーで使用されているアプリケーション フォルダ構造のコピーを使用します。

¥MyTeamProject1 →□TeamProject1 のコンテナ フォルダ
¥Main →□複数のプロジェクトにまたがる .sln ファイルが含まれている
    ¥Source
¥MyApp 1 →□MyApp1.sln が含まれている
¥Source
¥ClassLibrary1 →□ClassLibrary1.csproj が含まれている
¥MyApp1Web →□Default.aspx が含まれている
¥UnitTests →□単体テスト プロジェクトとソースが含まれている
¥ClassLibrary1Tests
¥MyWinApp1Tests
¥SharedBinaries →□共有しているバイナリ (ライブラリなど)
¥SharedSource →□共有しているソース コード
¥Docs →□製品ドキュメントが含まれている
¥Tests →□テスト用のコンテナ
¥FunctionalTests
¥PerformanceTests

¥SecurityTests

注意:アプリケーションのルートからローカル コンピュータへワークスペースのマッピングを作成する場合には、クライアント側の構造は、サーバー側の構造を自動的にミラーリングします。ただし、非常に規模の大きいプロジェクトでは、この処理によってワークスペースのロード時間が長くなります。大規模なプロジェクトでアプローチを最適にするには、ルートの下でワークスペースのマッピングを作成し、開発に必要なファイルのみを取得するようにします。

分岐フォルダ

分岐を伴う開発の分離をサポートするには、**Main** の "兄弟" として別のフォルダを作成します。また、追加の Visual Studio ソリューション (.sln) ファイルを **Main** の中に配置すると、開発者は、プロジェクトをグループ化したものをまとめて作業することができます。**Main** ソース フォルダから作成された分岐を使用して、製品リリースの継続保守、または並行した開発ストリームをサポートできます。

次の構造例では、**Main** ルート フォルダの他に、(**Main** から分岐した) **Development** フォルダを使用して、機能またはチームに対して分離を行っています。リリースの分岐に対するコンテナである **Releases** フォルダ (これも **Main** から分岐しています) は、継続保守および対象リリースのロックダウンに必要な、リリース済みのビルドに対する分離を実現します。

$MyTeamProject1
/Development
/FeatureBranch1
/Source
/MyApp
/FeatureBranch2
/Source
/MyApp
/Main
/Source

/Releases
/Release1 – 保守
/Source
/MyApp
/Release2 – 保守
/Source
/MyApp
/Release3 – 対象リリースのロックダウン
/Source
/MyApp

注意:必要がない場合は、分岐を行わないでください。必要な場合は、リリースにラベルを付けて後で分岐することができます。

プロジェクトのグループ化およびソリューションの構造の詳細については、「第 3 章 ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

分岐のシナリオ、および関連するフォルダ構造の詳細については、「第 5 章 分岐とマージの方針の定義」を参照してください。

ワークスペースの説明

TFS のワークスペースは、TFS のソース管理におけるファイルとフォルダのクライアント側のコピーです。ワークスペースは、ソース管理フォルダをローカル ファイル システムのディレクトリへマップします。ローカル コンピュータのワークスペース内でファイルを変更すると、これらの変更がアトミック単位としてサーバーにチェックインされるまで、ローカルな変更 (保留中の変更とも呼ばれます) はワークスペース内で分離されます。バッチとしてチェックインされた変更のまとまりは、変更セットと呼ばれます。

1 つのワークスペースに、複数のチーム プロジェクトに対する参照を含めることができます。また、

複数のワークスペースを使用して、自身だけで使用する目的でファイルまたはバージョンを分離することができます。ワークスペースは、コンピュータごと、ユーザー アカウントごとに設定されます。したがって、使用するそれぞれのコンピュータに対して、ワークスペースの異なる定義を持つことができます。また、シングル ユーザーとして、1 つのコンピュータ上に複数のワークスペースを持つことも可能です。

注意:ローカル ファイル システム上の各物理ロケーションは、単一のワークスペースによってのみマップされます。ただし、各サーバー ディレクトリは、異なるワークスペースを使用することで、まったく異なるローカル ディレクトリにマップすることができます。

新しいワークスペース マッピングの作成

マッピングは再帰的であるため、新しいワークスペース マッピングを作成して、そのワークスペースのルートで **[最新バージョンの取得]** の操作を実行すると、ローカル フォルダ構造の全体が自動的に作成されます。新しく作成されたローカル フォルダの構造は、サーバーの構造と一致しています。

新しいワークスペースのマッピングを作成する場合には、次の推奨事項に留意してください。

- プロジェクトの所有者は、ソース管理へ最初にソリューションを追加する前に、ローカルで正しいフォルダ構造が使用されていることを必ず確認します。
- チーム プロジェクトに対するワークスペースのマッピングが最初に確立され、**[最新バージョンの取得]** の操作を実行する場合には、ルートのチーム プロジェクト フォルダを、C:¥DevProjects¥TeamProject1 などの適切なローカル フォルダへマップしてください。

ワークスペースのマッピングを格納する場所

ワークスペースの情報は、クライアントとサーバーの両方で保持されます。クライアントでは、ワークスペースの情報は、次のフォルダにある VersionControl.config に保持されます。
¥Documents and Settings¥[user]¥Local Settings¥Application Data¥Microsoft¥Team Foundation¥1.0¥Cache.

VersionControl.config ファイルは、ワークスペースの名前をコンピュータ上のローカル ディレクトリへマップします。このファイルには、個々のソース管理フォルダとユーザーのローカル ディレクトリ間のマッピングは保持されません。この情報は、サーバー上で TfsVersionControl データベース内のいくつかのテーブル (tbl_Workspace や tbl_workingfolder) で保持されます。

クローキング

ソース管理ツリーの一部を取得できないようにする場合は、パフォーマンスの最適化としてクローキングを使用できます。クローキングを使用する場合の一般的なシナリオは以下のとおりです。

- プロジェクトをローカルにビルドしたいが、ビルドにフォルダが必要ない場合 (ドキュメント フォルダなど)。
- 大規模なチーム プロジェクトの一員で、プロジェクトの一部のみを取得したい場合。

これらのいずれのシナリオでも、クライアントがこれらのフォルダを取得しないようにするために、フォルダをクロークすることができます。ワークスペースを編集し、作業フォルダのステータスを "アクティブ" から "クローク" へ変更して、クライアントのフォルダをクロークします。

クロークする場合には、次の推奨事項に留意してください。

- 個々のファイルはクロークしないでください。このようにすると、後でメンテナンスの問題がプロジェクトで発生することが多くなります。
- 大規模なプロジェクトでは、プロジェクトに対して複数のワークスペースを作成するのではなく、ルート フォルダをマップし、サブ フォルダをクロークしてください。

バージョン管理の対象となるファイル

ソース管理に追加する必要がある重要なファイルの種類について、次の一覧に示します。これらのファイルの種類は、**[ソリューションをソース管理に追加]** をクリックしたときに追加されます。

- **ソリューション ファイル (*.sln)。**ソリューション ファイルは、構成要素のプロジェクト、依存関係の情報、ビルド構成の詳細、およびソース管理のプロバイダの詳細の一覧を保持しています。
- **プロジェクト ファイル (*.csproj または *.vbproj)**。プロジェクト ファイルには、アセンブリのビルド設定、参照先のアセンブリ (名前およびパス)、ファイルの一覧が含まれています。
- **Visual Studio のソース管理プロジェクト メタデータ (*.vspscc)。**これらのファイルは、プロジェクトのバインディング、除外リスト、ソース管理のプロバイダ名、および他のソース管理メタデータを保持しています。
- **アプリケーションの構成ファイル (*.config)**。Extensible Markup Language (XML) の構成ファイルには、プロジェクトおよびアプリケーションに特有の詳細が含まれています。これを使用して、アプリケーションの実行時の動作を制御します。Web アプリケーションは Web.config というファイルを使用します。非 Web アプリケーションは App.config というファイルを使用します。

  注意:実行時に、Visual Studio のビルド システムは App.config をプロジェクトの Bin フォルダへコピーし、<YourAppName>.exe.config という名前に変更します。非 Web アプリケーションの場合は、構成ファイルは新しいプロジェクトに自動的に追加されません。必要な場合は、手動で追加します。これは必ず App.config という名前にして、プロジェクト フォルダの中に配置します。

- **ソース ファイル (*.aspx、*.asmx、*.cs、*.vb など)**。これらのファイルはソース コード ファイルで、アプリケーションの種類および言語によって異なります。
- **バイナリ依存 (*.dll)。**プロジェクトが、サードパーティの動的リンクライブラリ (DLL) などのバイナリ依存を利用している場合は、これらの依存を、ソース管理のプロジェクトへ追加することも必要です。依存関係の管理の詳細については、「第 6 章 Visual Studio Team System におけるソース管理の依存関係の管理」を参照してください。

ソース管理の対象にならないファイル

次のファイルは開発者に特有のものであるため、バージョン管理には追加しません。

- **ソリューションのユーザー オプション ファイル (*.suo)。**これらのファイルには、個々の開発者が Visual Studio IDE に対して行った個人的なカスタマイズが含まれています。

- **プロジェクトのユーザー オプション ファイル (*.csproj.user または *.vbproj.user)。**これらのファイルには、開発者特有のプロジェクト オプション、および Visual Studio が参照先のアセンブリを見つけるときに使用するオプションの参照パスが含まれています。
- **WebInfo ファイル (*.csproj.webinfo または *.vbproj.webinfo)。**このファイルは、プロジェクトの仮想ルートの場所を追跡します。これは、個々の開発者が、プロジェクトに関する作業コピーの仮想ルートを指定できるように、ソース管理には追加されません。この機能がある場合は、チームのすべてのメンバは、Web アプリケーションの開発時に、整合性のとれた (ローカルな) 仮想ルートの場所を使用することが推奨されます。
- **ビルドの出力。**これらの中には、アセンブリ DLL、相互運用機能アセンブリ DLL、および実行可能ファイル (EXE) が含まれます(ただし、サードパーティのバイナリなど、ビルド プロセスの一部としてビルドされないアセンブリは、前述のように、バージョン管理の下に置いてください)。

まとめ

TFS のソース管理におけるプロジェクトは、チーム開発を効率よく行えるように構造化します。**Main** というルートレベルのフォルダを使用して、Visual Studio のプロジェクトをグループ化します。**Main** フォルダには、ソース コード、テスト、ドキュメント、チームビルドの種類などプロジェクトのさまざまな資産を保存しておくための子フォルダがあります。

ユース ケースや設計のドキュメントなど、チームの内部ドキュメントに対しては SharePoint を使用します。お客様に提供する、製品関連のドキュメントに対しては TFS のソース管理を使用します。これには、インストールおよび配置のガイド、操作ガイド、およびヘルプ ファイルを含めることもできます。

フォルダ構成の違いによって生じる混乱を減らすために、サーバー側とクライアント側のフォルダ構造は、常に同期するようにします。大規模なプロジェクトでアプローチを最適にするには、ルートの下でワークスペースのマッピングを作成し、開発に必要なファイルのみを取得するようにします。

参考資料

- Team Foundation のソース管理の詳細については、http://msdn2.microsoft.com/en-us/library/ms364074(VS.80).aspx の「Using Source Code Control in Team Foundation」を参照してください。
- ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。

第 5 章 分岐とマージの方針の定義

目的

- 分岐をする場合、および分岐しない場合について理解する。
- プロジェクトに対する分岐およびマージの方針を選択する。
- 非常に規模の大きいチームをサポートするために、標準の分岐方針をどのように導入すべきかを説明する。
- さまざまな分岐シナリオに対して適切なフォルダ構造を認識する。

概要

この章では、一般的なシナリオの範囲での、分岐およびマージの方針を紹介します。通常は、リリースまたは並行開発のいずれかをサポートするために分岐が必要になります。
多くの簡単なシナリオでは、分岐は不要で、ビルドにラベリングするだけで十分です。たとえばラベルを使用して、将来任意のタイミングでビルドを再作成する、特定のビルドの作成でソース ファイルのどのバージョンが使用されたかを調べる、といったことが可能です。並行チームで、別に作業しているが一部の機能が重複している場合、または 1 つのリリースをサポートする場合に、分離が必要であれば、分岐を検討します。

この章の参照の仕方

この章を使用して、分岐をする場合および分岐しない場合について理解します。状況に対して分岐が適

切であると判断したら、この章を使用して、効果的に分岐およびマージする方法について学習します。

分岐およびマージの概要を知りたい場合は、この章を最初から最後まで読んで、分岐のさまざまな方針、および分岐が最適であるシナリオについて習得します。特定のシナリオについて調べたい場合は、そのセクションを参照してください。この章を最大限に利用するには、次のようにします。

- **シナリオの一覧を使用します。**シナリオ一覧を使用して、自身の状況に最も近いシナリオを見つけます。推奨事項を読んで、ニーズに適した分岐フォルダ構造を作成します。
- **付属のガイダンスを使用します。**分岐とマージのガイドラインの概要については、このガイドの「ソース管理のガイドライン」の中の分岐およびマージのガイドラインを参照してください。

分岐およびマージのシナリオ

次に、分岐を作成し、マージを行う必要があるシナリオの例を示します。

- ビルドの不具合の問題が常に生じる場合には、開発用の分岐を作成して、並行開発作業を分離する必要があります。
- 安定性の問題が生じる原因となる機能がある、または相互に安定性の問題を生じる原因となっているチームがある場合は、ソース管理の開発コンテナ フォルダの下に機能またはチームごとの分岐を作成します。

開発チームにとって必要ない場合は、分岐を行わないでください。分岐によって、ソース ツリーの保守およびマージのタスクが新たに発生します。基幹業務アプリケーションを構築しているような開発チームの多くは、短いリリース サイクルで作業しており、分岐が必要ありません。パッケージ アプリケーションをビルドする独立系ソフトウェア ベンダ (ISV) など、比較的長いリリース サイクルで作業している開発チームでは、開発プロセスの一環として分岐が必要になることが多くなります。

開発ストリームが 1 つである場合、つまり段階的に継続したリリースを実行する場合は、分岐を作成する必要はありません (ただし、開発作業が不安定になるような、変更による不具合が頻繁に発生する場合は例外です)。

実際の一般的なシナリオ

以下に、最も一般的な分岐のシナリオを示します。

- **シナリオ 1 – 分岐しない。**チームが、メインのソース ツリーのみから作業している場合。このような場合は、分岐を作成せず、分離する必要はありません。このシナリオは一般的に、チームまたは機能に対する分離を必要としない、またリリースに対する分離も必要としない小、中規模のチームに適用されます。
- **シナリオ 2 – リリース用の分岐。**チームが、継続リリースをサポートするために分岐を作成する場合。これは、2 番目に一般的なケースで、リリースを安定させるために分岐を作成する必要があります。この場合には、リリースを安定させるために、リリース前に分岐を作成し、ソフトウェアがリリースされた後で、リリースの分岐における変更をメインのソース ツリーへマージして戻します。
- **シナリオ 3 – 保守用の分岐。**チームが、古いビルドを保持するために分岐を作成する場合。この場合には、現在稼働しているビルドを安定させるように、保守作業のための分岐を作成します。保守の分岐における変更は、メイン ツリーへマージして戻す場合も、戻さない場合もあります。たとえば、一部の顧客に対する緊急措置を行い、その措置をメイン ビルドに含めないようにすることができます。
- **シナリオ 4 – 機能用の分岐。**チームが、機能に基づいて分岐を作成する場合。このような場合は、開発の分岐を作成し、開発の分岐内で作業し、メインのソース ツリーへマージします。並行して行う特別な機能での作業用に、別の分岐を作成することができます。
- **シナリオ 5 – チーム用の分岐。**サブ チームを分離するために分岐を作成し、これらのチームが変更の影響を受けずに作業したり、独自のマイルストーンに対して並行に作業したりできるようにします。

上記のうち、いずれかのシナリオを経験することがあるでしょう。これらのシナリオは、自身にとってどのガイダンスが適しているかを理解するための基準として使用してください。

フォルダの例と目的

以下に、分岐とマージのシナリオに対してソース ツリーを構造化するときに、Microsoft® Visual

Studio® Team Foundation Server (TFS) のソース管理に作成できるフォルダの例を示します。

- **Development** は **Main** から分岐したもので、現在の開発を分離するために使用します。Development の分岐は、**Main** に影響を及ぼさずに、リスクの高い変更を行うための一時的なものにすることも可能です。
- **Main** にはメインのソース ツリーが含まれています。他の分岐における変更は、ここに統合されます。
- **Releases** には、既に出荷したけれども、顧客のために保持する必要がある分岐が含まれています。これにより、開発の分岐に存在している現在の開発から、分岐することができます。これには、**Main** から分岐した対象リリースの分岐、およびリリース前にロックダウンしているバージョンも含まれています。この分岐内で、リリースのためにソフトウェアを準備し、他の開発者は、新しい機能のために **Development** 分岐で、引き続き作業することができます。

以下のセクションで、前述のシナリオにおける分岐の使用、および具体的な例を示します。

シナリオ 1 – 分岐しない

このシナリオは通常、分離した開発に関与しない、比較的規模の小さいチームに当てはまります。ビルドをラベリングすることによって、特定のビルドに対応するソースを取得することができます。Main フォルダから直接作業できるため、分岐の複雑さを導入する必要がありません。以下に、分岐なしのシナリオを表す図を示します。

My Team Project
Main
Source

シナリオ 2 – リリース用の分岐

このシナリオでは、分岐を作成してリリースを安定させ、ソフトウェアがリリースされた後で、リリースの分岐をメインのソース ツリーへマージして戻します。以下に、リリース用の分岐を表す図を示し

ます。

My Team Project

Main →□メインの統合分岐

Source

Releases

Release 1 →□リリースの分岐

Source

シナリオ 3 – 保守用の分岐

このシナリオには、現在作成しているビルドを安定させるために、保守作業のための分岐を作成します。以下に、保守用の分岐を表す図を示します。これは、リリース用の分岐と非常に似ていますが、この分岐は、リリースをサポートするために長い間保持されます。

My Team Project

Main →□メインの統合分岐

Source

Releases → 保守の分岐のコンテナ

Release 1 →□保守の分岐

Source

Other Asset Folders

Release 2 →□保守の分岐

Source

Other Asset Folders

シナリオ 4 – 機能用の分岐

このシナリオでは、開発の分岐を作成し、その分岐内で作業し、メインのソース ツリーへマージしま

す。開発の分岐は、製品の機能に基づいて編成します。次に、機能ごとに分岐している開発を表す、物理的な図を示します。

My Team Project
Development →□分離された開発の分岐コンテナ
Feature A →□機能の分岐
Source
Feature B →□機能の分岐
Source
Feature C →　□機能の分岐
Source
Main →□メインの統合分岐
Source

シナリオ 5 – チーム用の分岐

このシナリオは、前述の機能による分岐のシナリオに似ていますが、製品の機能ではなく、サブチームに従って開発の分岐を編成している点が異なります。このシナリオでは、チームと機能の間に 1 対 1 の対応が存在する場合もありますが、1 つのチームが複数の機能に対して作業する場合もあります。以下に、サブチームごとに分岐している開発を示す、物理的な図を示します。

My Team Project
Development →□分離された開発の分岐コンテナ
Team 1 →□チームの分岐
Feature A →□開発用に分離された分岐
Source
Feature B →□開発用に分離された分岐
Source
Team 2 →□チームの分岐
Feature A →□開発用に分離された分岐

Source

Feature B →□開発用に分離された分岐

Source

Main →□メインの統合分岐

Source

論理構造

論理構造は、それぞれの分岐の親/子の関係から構成されています。これは、ソース管理エクスプローラで参照できる物理構造とは異なる場合があります。たとえば、**Development** は実際には **Main** の子であるのに、前述の物理構造では **Development** と **Main** は同じレベルで現れます。

図 5.1 は、論理関係の例、および構造間での分岐とマージのフローを示しています。

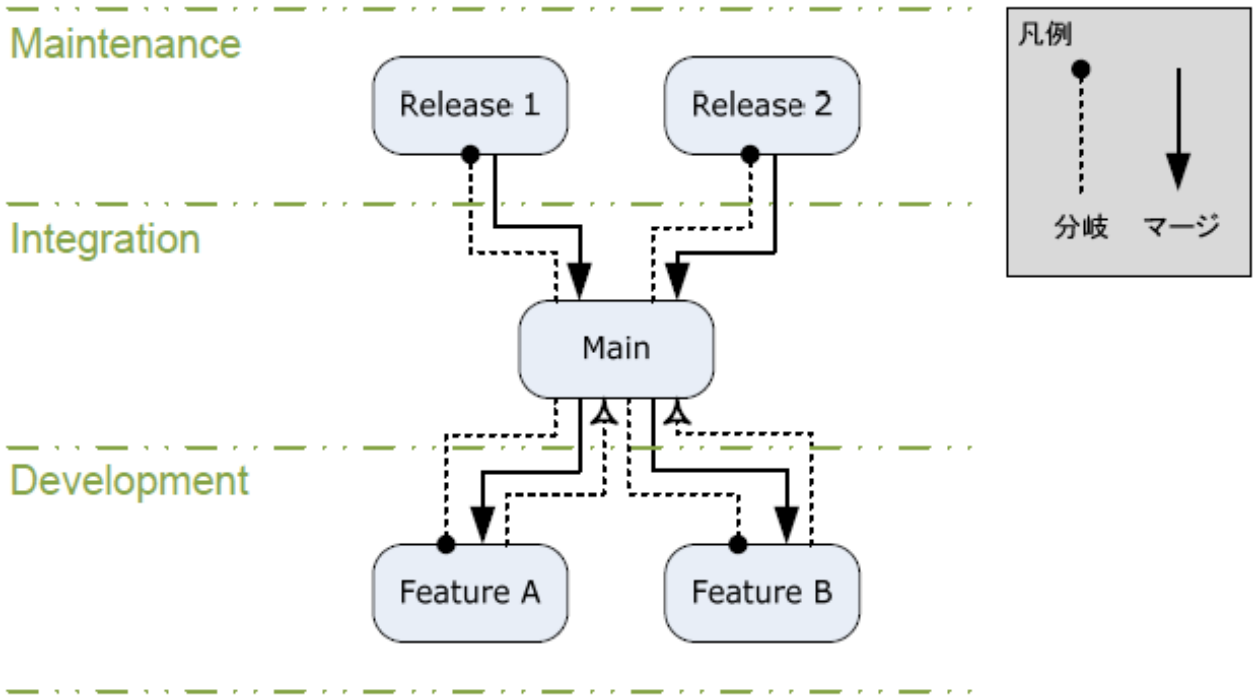


図 *5.1* 分岐およびマージのフローを示す論理関係

リリースのシナリオ

図 5.2 は、リリースに対して分岐するときの一般的なタイムラインを示しています。

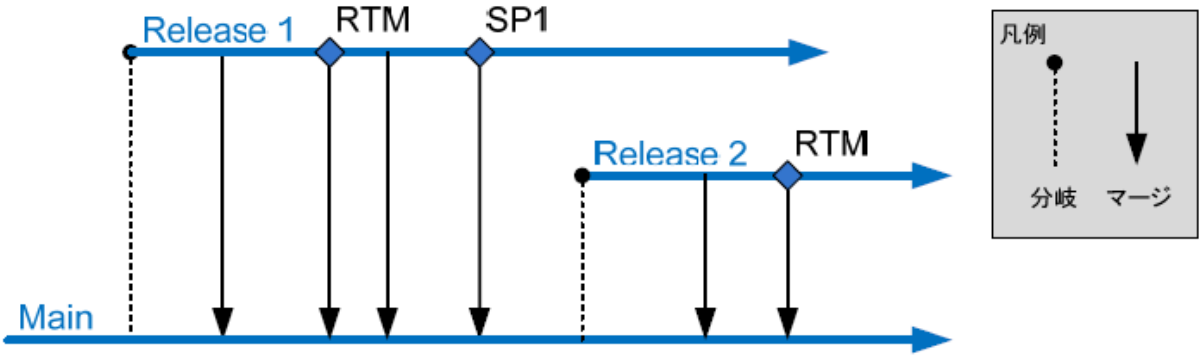


図 *5.2* リリースの分岐のタイムライン

イベントの順序は以下のようになります。

1. リリースをロックダウンする準備ができたら、**Main** から **Release 1** の分岐が作成されます。
2. **Main** に対する定期的なマージによって、リリースにおけるバグの修正がメインの統合分岐に確実に移行されます。
3. リリースされたビルドが **RTM** の分岐内でラベル付けされ、**Main** へマージされて戻されます。
4. サービス パックの SP1 がリリースされます。ビルドにラベルが付加され、変更が **Main** にマージされます。
5. **Release 1** の分岐は SP1 のサポート用に存続し、将来的なサービス パックが可能になります。

このプロセスを、将来のリリースに対して繰り返します。

注意:リリースの分岐をうまく使用するには、実装の前に、修正を適用しなければならない分岐はどれかを特定することが必要です。ホットフィックスやサービス パックとして修正をリリースする場合は、対象の **Release** 分岐で最初に変更を行い、次にそれを **Main** へマージして、将来のリリースに確実に適用されるようにします。

分離された開発のシナリオ

図 5.3 は、開発の分離に対して分岐するときの一般的なタイムラインを示しています。

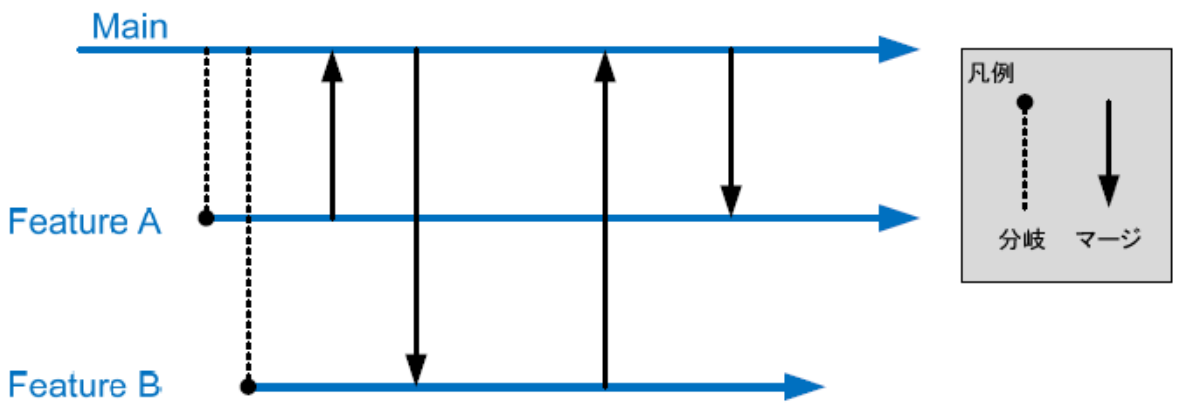


図 5.3 開発の分離に対する分岐のタイムライン

イベントの順序は以下のようになります。

1. Feature A に対する開発を分離するために、機能の分岐が作成されます。
2. Feature B に対する開発を分離するために、機能の分岐が作成されます。
3. 各チームは、機能のマイルストーンを達成したときに、それぞれの変更を **Main** にマージし、変更した内容をメイン ビルドに統合して、他のチームが使用できるようにします。
4. 定期的なスケジュールで、各チームは **Main** における最新の変更をマージし、他のチームの作業と常に同期をとれるようにして、マージ時の競合が減少するようにします。
5. 機能が完成したら、変更を完全に **Main** へマージして戻し、機能の分岐を消去します。

分岐に関する考慮事項

分岐を行う場合には、次のことに注意してください。

- 開発チームで、ファイルの同じセットで並行して作業する必要がなければ、分岐はしないでください。分岐が必要かどうかわからない場合は、ビルドにラベルを付けて、後でそのビルドから分岐を作成することができます。各分岐に多数の変更がある場合は特に、分岐をマージする処理は時間がかかるうえ、複雑になります。
- 分岐ツリーを構造化する場合には、階層全体ではなく、階層に沿って (つまり分岐ツリーの上下を対象として) マージすればよいように考慮します。階層全体を分岐するには、基点のないマージを使用しなければなりませんが、これは、多くの競合を手動で解決する必要があります。

- 分岐の階層は、分岐の親と分岐の子がベースになっています。これは、ディスク上のソース コードの物理構造とは異なる場合があります。マージを計画する場合は、ディスク上の物理構造ではなく、論理的な分岐の構造に留意してください。
- 分岐はあまり深くしないでください。各マージを実行して、競合を解決するには時間がかかるためです。分岐の構造を深くすると、子分岐の変更を親分岐に反映させるのに非常に時間がかかります。これはプロジェクトのスケジュールにマイナスの影響を及ぼし、バグを修正する時間が長くなります。
- 分岐は上位レベルで行い、その中に構成ファイルおよびソース ファイルを含めるようにします。
- 分岐構造は時間をかけて展開してください。
- マージでは、マージを実行し、競合を解決するために 1 人以上の開発者が必要です。ビルドを不安定にさせるような、不適切なマージを行ってしまうことも多いため、マージしたソースは完全にテストする必要があります。
- 分岐階層全体のマージは特に難しく、他の場合であれば自動で処理できるような多数の競合を、手動で処理しなければなりません。

分岐を作成するかどうかの判断は、競合をリアルタイムでマージするコストが、分岐間の競合をマージするための総コストより高いかどうか、という点に集約することができます。

大規模なプロジェクトでの考慮事項

規模が大きくて、開発サイクルの長い開発チームは、多くの場合、以下の点において規模の小さいチームとの違いが顕著になります。

- より複雑な分岐およびマージの構造が必要になる。
- 複数のソリューションおよびチーム プロジェクト間で依存関係を管理することが多くなる。
- コンポーネントおよびチームに対して複数のビルドを保持することが多くなる。

大規模なチームは、チームと機能の両方で分岐することが多くなります。また、外部の依存関係を統合することを意図して、1 つ以上の分岐を持つことも多くなります。このようにすると分岐の構造がより深くなるため、開発の分岐における変更と、その変更をメインの統合分岐へマージするまでの時間差が

大きくなります。この理由から、分岐を作成する場合には、マージの方針について慎重に検討する必要があります。

たとえば、スケジュールに基づいてマージするか、イベント ドリブンでマージするかについては、以下の内容を考慮します。

- 逆統合は、通常はスケジュールに基づいて行われます。たとえば、開発の分岐からメインの統合分岐へマージする場合です。
- メインの統合分岐から開発分岐へ統合を行うような前方統合は、通常、機能のマイルストーンを達成したなどのイベントに基づいて行われます。このイベントは、開発分岐のチーム担当者が、親分岐における変更をマージする準備ができたと思ったときに発生します。

分岐/マージの方針は、どのくらいの頻度でビルドを生成したいかによって論理的に理由付けします。分岐構造が深くなると、子分岐からメインの統合分岐へマージする時間が長くなります。開発チームにとって問題が発生しそうな兆候としては、次のものが考えられます。

- 分割されたビルドで、修正をメインの統合分岐へ反映させるのに長い時間がかかる。
- 機能をメインの統合分岐へ反映させるのに長い時間がかかるために、マイルストーンがない。
- それぞれの分岐における変更をマージするのに膨大な時間がかかる。

これらのことがチームで問題になる場合は、分岐構造の深さを減らすことを検討してください。

以下に、複雑な分岐構造の例を示します。

- **My Team Project**
    - o**Development** – 現在の開発分岐を分離するためのコンテナ
    - **Feature A** – 開発用に分離された分岐
    - **Source**
    - **Feature B** – 開発用に分離された分岐
    - **Source**

o**Main** – メインの統合およびビルドの分岐。すべての変更がここに収集される。

- **Source**
- **Other Asset Folders**

o**Releases** – 対象リリースおよび保守の分岐のコンテナ

- **Release 2**– 有効な保守の分岐
- **Source**
- **Other Asset Folders**
- **Release 3** – 有効な保守の分岐
- **Source**
- **Other Asset Folders**
- **Release 4 –** リリースのためにロックダウンされているコードが含まれている分岐
- **Source**
- **Other Asset Folders**

o**Safe Keeping**

- **Release 1** – 安全な保管のための以前のリリース
- **Source**
- **Other Asset Folders**

この構造には、次のものが含まれています。

- 複数のチームが分離して作業するための機能の分岐。
- すべてのチームにおける変更、およびリリースの分岐におけるバグの修正を収集するためのメインの分岐。
- 対象リリースのロックダウンで使用されるリリースの分岐。
- 現在も維持およびサポートの対象となっている以前のリリースのための、保守の分岐。
- 現在は維持されていない以前のリリースのための、安全管理の分岐。

まとめ

分岐は通常、リリースをロックダウンする、以前のリリースを保持する、または分離された並行開発を

サポートする、といった目的で作成されます。分岐するための相応な理由がない限り、分岐は行わないでください。
分岐を作成する場合は、階層全体ではなく、階層に沿って (つまり分岐ツリーの上下を対象として) マージすればよいように、分岐ツリーを論理的に構造化します。階層全体をマージすると、時間がかかりエラーが発生しやすくなります。
規模の大きいチームでは、分岐の構造が深くなるため、開発の分岐で発生する変更と、その変更がメインの統合分岐にマージされるまでの時間差が大きくなり、このための準備が必要になります。

参考資料

- 分岐とマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

第 6 章 Visual Studio Team System におけるソース管理の依存関係の管理

目的

- Microsoft® Visual Studio Team System のソース管理の依存関係を管理する。
- 同じチーム プロジェクトの異なるソリューションからプロジェクトおよびアセンブリを参照する。
- 他のチーム プロジェクトからプロジェクトおよびアセンブリを参照する。

- サードパーティのアセンブリを参照する。
- チーム環境で Web サービスの参照を管理する。
- チーム環境でデータベースの参照を管理する。

概要

この章では、Visual Studio のソリューション内、およびソリューション間の両方でソース管理の依存関係をどのように処理すればよいかについて説明します。ビルドの安定性を保持し、継続するソース管理の保守費用を低減するためには、チーム環境において依存関係を管理するための一貫したアプローチが必要です。

依存関係には、他のプロジェクト、外部のアセンブリ、Web サービス、およびデータベースが含まれます。時間が経過すると、依存関係は必ず変化し、その結果として、ビルドのプロセスおよびアプリケーションのビルド順序に影響を及ぼします。依存関係で適切な管理アプローチを使用すると、ビルドをできるだけシームレスにしたまま統合プロセスを改善できます。

この章の参照の仕方

この章を使用して、チーム環境における依存関係の管理について学習します。この章を最初から最後まで読むことも、特定の依存関係の管理要件に対処するセクションだけを読むこともできます。チーム環境における一般的な依存関係の管理シナリオを理解するには、「シナリオとソリューション」のセクションを使用します。このセクションは、それぞれの依存関係のシナリオを詳細に説明する以降のセクションへの出発点となります。

- **「プロジェクトの参照」セクション**を使用して、対象のチーム プロジェクトの内部および外部で、他のプロジェクトの依存関係を管理する方法について学習します。
- **「サードパーティのアセンブリの参照」セクション**を使用して、ソースを所有していないサードパーティのアセンブリに関する依存関係を管理する方法について学習します。
- **「Web サービスの参照」セクション**を使用して、チーム環境で共有している Web サービスを参照する方法について学習します。

- **「データベースの参照」セクション**を使用して、チーム環境で共有しているデータベースに対して参照および接続する方法について学習します。

シナリオとソリューション

依存関係を管理する場合には、以下のシナリオが一般的です。

1. 同じソリューション内の別のプロジェクトによって生成されたアセンブリを参照したい。
2. 別のソリューション内の別のプロジェクトによって生成されたアセンブリを参照したい。
3. 他のチーム プロジェクトに含まれているアセンブリを参照したい。
4. サードパーティのアセンブリを参照したい。

同じソリューションの他のプロジェクトで生成されたアセンブリを参照する

同じ Visual Studio ソリューションの他のアセンブリを参照する必要がある場合は、Visual Studio のプロジェクト参照を使用します。プロジェクト参照を使用すると、Visual Studio で、ビルドの構成 (デバッグ/リリース) の同期を保持する、アセンブリのバージョンが変わったときに、必要に応じてコンポーネントのバージョニングおよび再ビルドを追跡する、といったいくつかのことを自動的に行うことができます。

別のソリューションのプロジェクトで生成されたアセンブリを参照する

別の Visual Studio ソリューションのプロジェクトで生成されたアセンブリを参照する必要がある場合は、次の 2 つの方法のいずれかを使用します。

- バイナリ アセンブリをポイントするためのファイル参照を使用する。
- ソリューションに Visual Studio プロジェクト (プロジェクトおよびソース ファイル) を追加し、プロジェクト参照を使用する。

ファイル参照はプロジェクト参照より脆弱であり、ビルド構成に従いません。また、Visual Studio のビルド依存関係では追跡できません。したがって、参照していたアセンブリが変わっても、Visual Studio

のビルド システムは、再ビルドが必要であることを自動的に認識できません。

別の方法として、外部のプロジェクトを自身のソリューションへ分岐して、バイナリをビルドし、プロジェクト参照を使用することができます。この方法はより堅牢ですが、変更を取り入れるために、定期的にソースの分岐からマージする必要があります。

他のチーム プロジェクトからアセンブリを参照する

チーム プロジェクト全体でソースまたはバイナリを共有している場合は、次の 2 つの方法があります。

- **分岐。**このアプローチでは、他のチーム プロジェクトから自身の対象のソリューションへソースを分岐します。これにより、共有場所のソースとサーバー側のプロジェクトを統一する構成が作成されます。
- **ワークスペースのマッピング。**このアプローチでは、他のチーム プロジェクトから自身の開発コンピュータ上のワークスペースへソースをマップします。これにより、他のチーム プロジェクトのソースと、クライアント側のプロジェクトを統一する構成が作成されます。

分岐は、依存関係をソース管理のサーバーに保存するため、よく好まれるアプローチです。ワークスペースのマッピングはクライアント側のみのアプローチで、アプリケーションを正常にビルドするためには、すべての開発者が自身のコンピュータにマッピングを作成する必要があり、さらにビルド サーバー上にもマッピングを作成しなければならないことを意味します。

分岐ではマージのオーバーヘッドが別途かかりますが、分岐により、更新されたバイナリまたはソースを取り込むことを、より明白に決定することができます。

サードパーティのアセンブリを参照する

このシナリオは、チーム プロジェクト全体での参照と非常によく似ていますが、バイナリのみを共有しており、ソース コードを共有していない点が異なります。分岐とワークスペースの選択についても非常に似ています。ただし、サードパーティのアセンブリはあまり頻繁には変わらないため、オーバーヘッドが低減する傾向にあります。

プロジェクトの参照

Visual Studio のプロジェクト、たとえば、複数のチーム プロジェクトで使用している共有ライブラリコードが含まれているチーム プロジェクトを持っている場合には、自身のチーム プロジェクト内でプロジェクトを管理することも、共有プロジェクトのために別のチーム プロジェクトを特別に作成することもできます。

後者のアプローチを選択し、共通の共有プロジェクトを使用する場合は、Microsoft Visual Studio Team Foundation Server (TFS) のソース管理は図 6.1 のようになります。

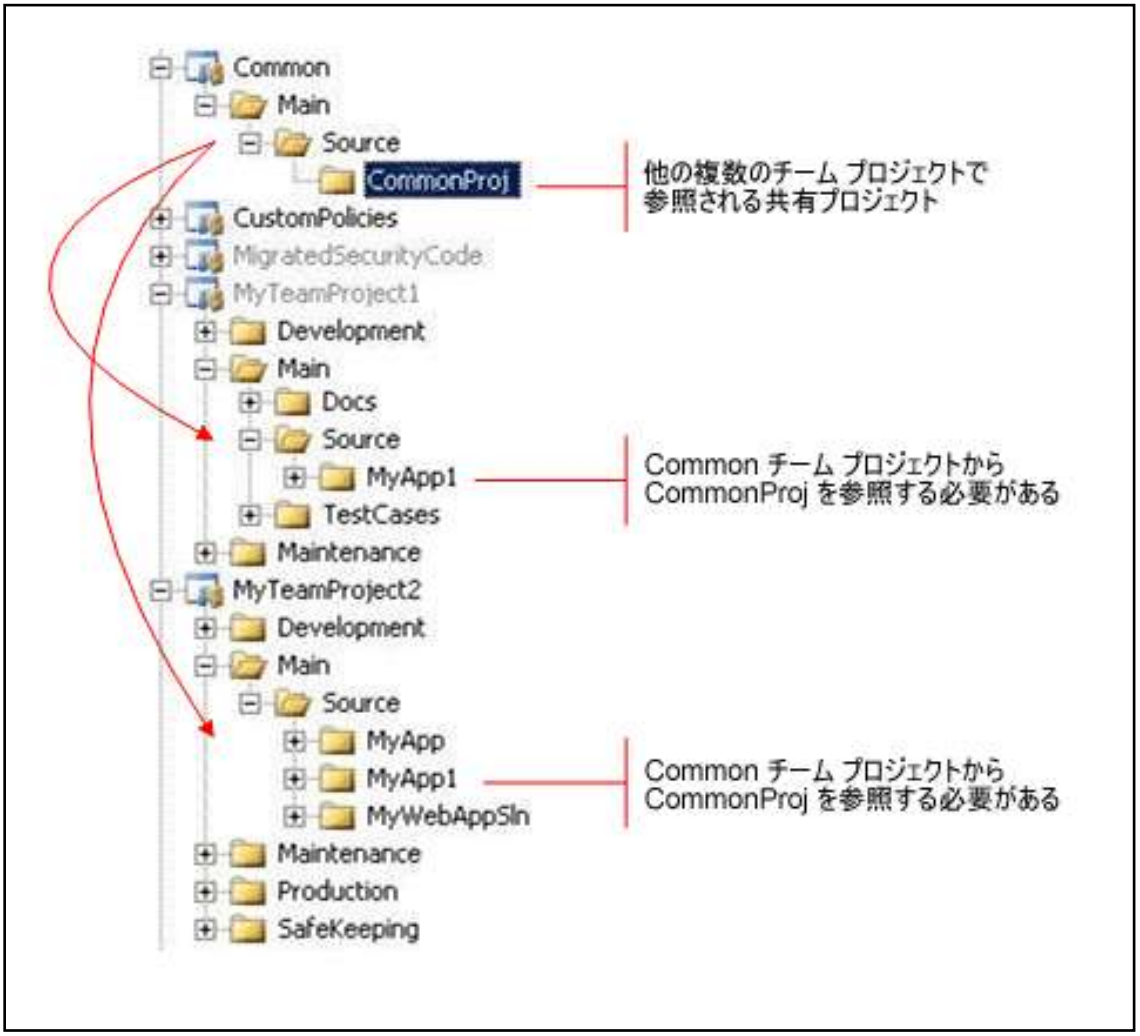


図 *6.1* 共通の共有プロジェクト フォルダ構造の使用

自身のチーム プロジェクトから共通の共有プロジェクトを使用するには、次の 2 つの方法があります。

- **分岐**
- **ワークスペースのマッピング**

分岐

ソースを共有しているほとんどのシナリオにとって、分岐は最も好まれる方法です。分岐によって、共有されているソースをプロジェクトへ取り込み、それを自身のチーム プロジェクトのソース管理にチェックインすることができます。このシナリオでは、共通の共有場所から自身のチーム プロジェクトへソースを分岐します。これにより、共有している場所のソースとサーバー側のプロジェクトを統一する構成が作成されます。

共有しているソースの変更は、分岐間のマージ プロセスの一部として取り込まれます。これにより、共有ソースにおいて変更を取り込むことをより明白に決定できるようになり、より簡単にテストできるようになりますが、オーバーヘッドが余分にかかります。また、このプロセスではチーム ビルドの使用がかなり簡単になります。これは、マッピングがサーバー側で行われ、ビルド サーバー上で複製する必要のある、クライアント側のマッピングがないためです。

たとえば、$TeamProject1 と $Common という 2 つのチーム プロジェクトを持っており、Common が共有プロジェクト ソースであるとすると、共有場所から、それを参照しているプロジェクトに対して分岐を作成します。TFS のフォルダ構造は、図 6.2 に示すようになります。

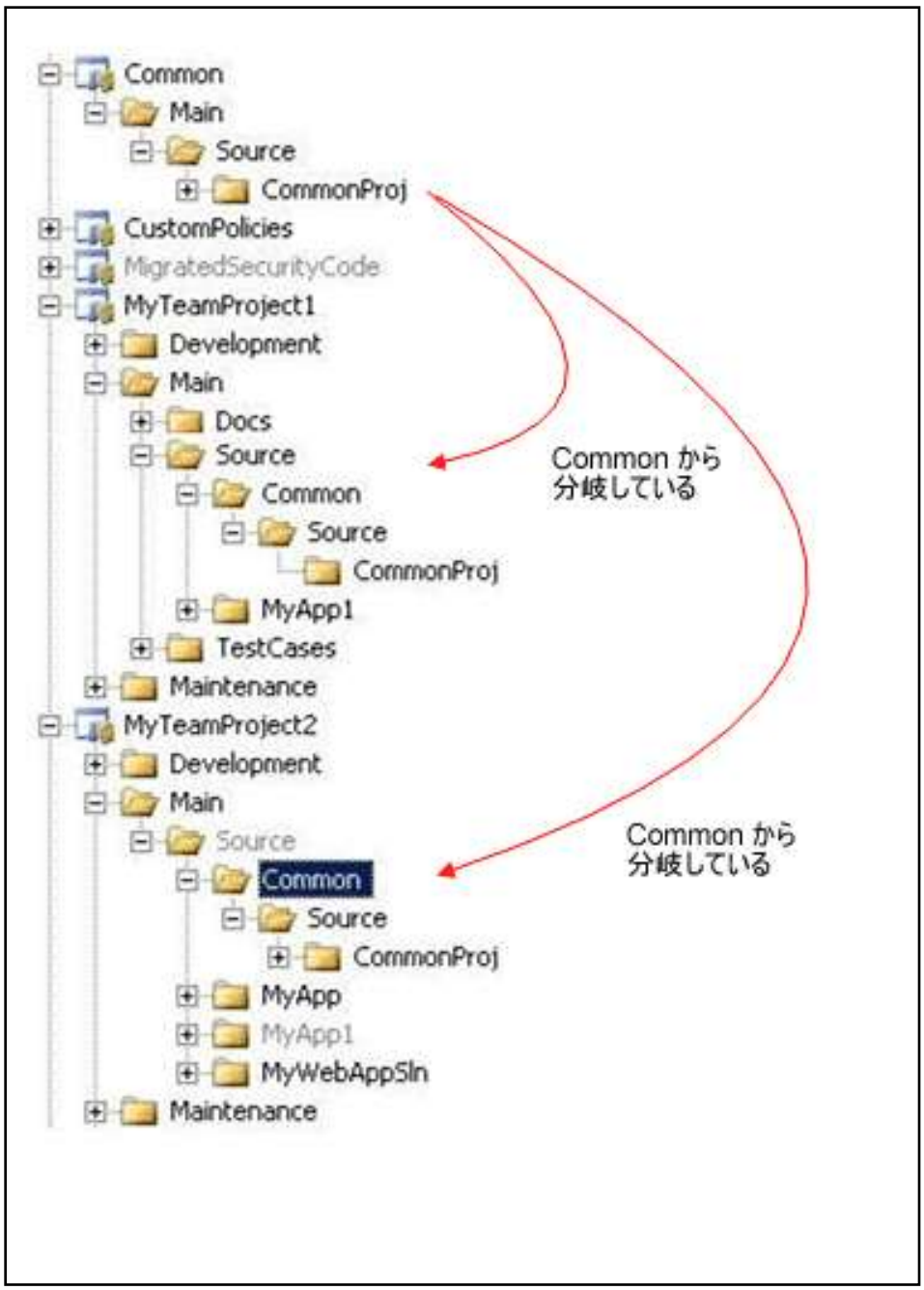


図 *6.2* 分岐の使用

ワークスペースのマッピングは以下のようになります。

ソース管理フォルダ	ローカル フォルダ
$/MyTeamProject1/Main/Source/	C:¥MyTeamProject¥Main¥Source

クライアント側のワークスペースのフォルダ構造は、図 6.3 に示すようになります。

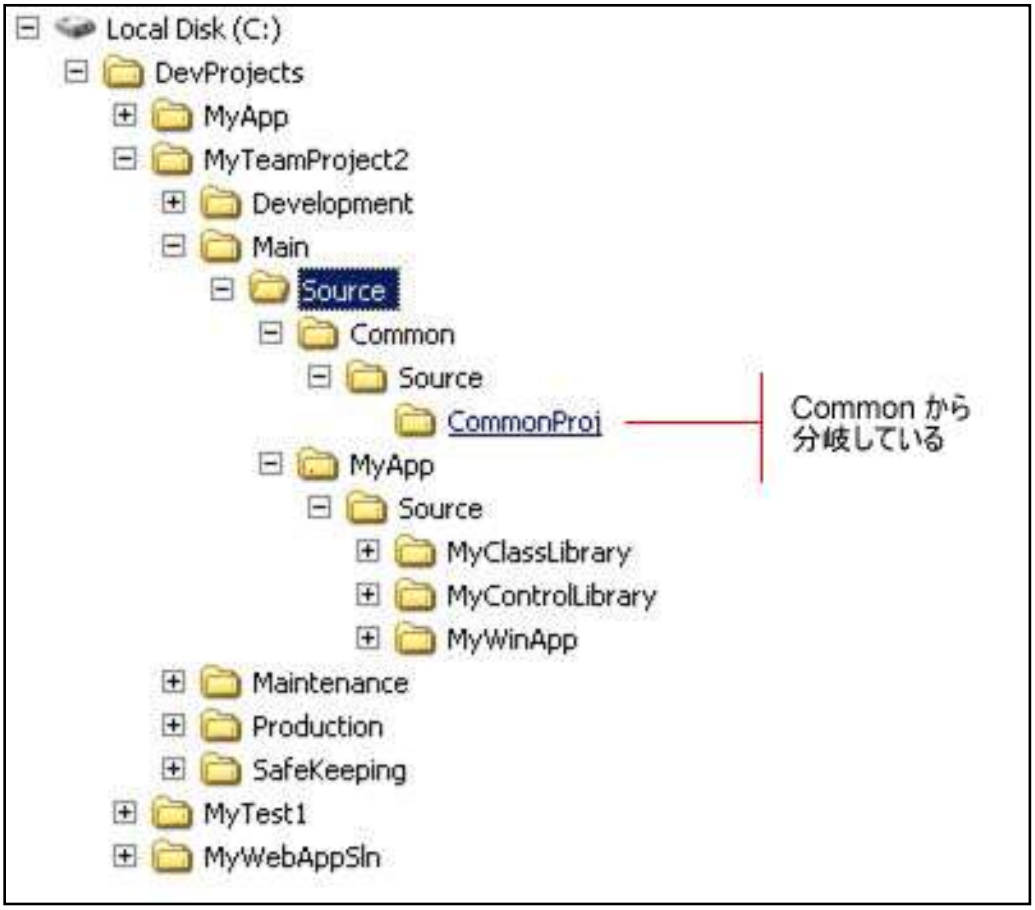


図 *6.3* クライアント側のワークスペース マッピング

ワークスペースのマッピング

分岐およびマージのオーバーヘッドを発生させずに、開発者が共有ソースからコードの変更をすぐに取り込めるようにしたい場合は、共通のプロジェクトから自身の開発コンピュータ上のワークスペースへ、共有ソースをマップすることができます。これにより、共有している場所のソースとクライアント側のプロジェクトを統一する構成が作成されます。

このアプローチの長所は、最新のソースをワークスペースに取り込むたびに、共有プロジェクトの変更も取り込まれることです。ただし、ワークスペースのマッピングはクライアント側の構築であるため、

チーム ビルドの使用はより複雑になります。

たとえば、$TeamProject と $Common という 2 つのチーム プロジェクトを持っており、Common が共有プロジェクト ソースであるとすると、共有場所からコードを参照するために、これらのプロジェクトは、クライアントのハード ディスク ドライブ上で共通のパスを共有します。クライアント側のワークスペースのフォルダ構造は、図 6.4 に示すようになります。

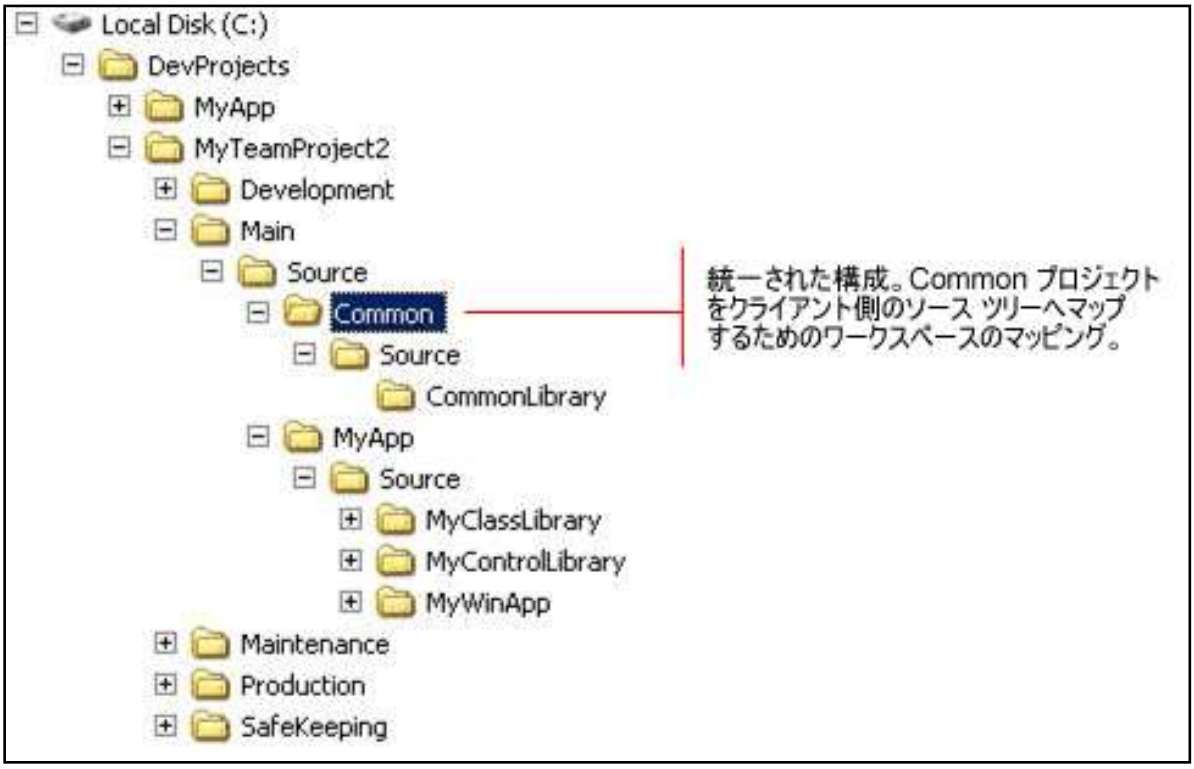


図 *6.4* ワークスペースのマッピングの使用

ワークスペースのマッピングは以下のようになります。

ソース管理フォルダ	ローカル フォルダ
$/MyTeamProject2/Main/Source/	C:¥DevProjects¥MyTeamProject2¥Main¥Source¥
$/Common	C:¥DevProjects¥MyTeamProject2¥Main¥Source¥ Common

詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の

「Working with multiple team projects in Team Build」を参照してください。

サードパーティのアセンブリを参照する

プロジェクト参照を使用できずに、対象のソリューションのプロジェクト セットの外部のアセンブリ (サードパーティのライブラリなど) を参照する必要があり、参照元のプロジェクトから自身のプロジェクトへ分岐を作成したくない、またはできない場合には、ファイル参照を設定する必要があります。

共有バイナリの管理は、共有プロジェクト ソースの管理と似ており、バイナリを保存したい場所、およびチームでどのようにしてバイナリにアクセスしたいかを決定する必要があります。複数のチーム プロジェクトで使用されているバイナリがある場合は、所有しているチームのチーム プロジェクト内でバイナリを管理することも、共有バイナリのために特別なチーム プロジェクトを作成することもできます。

共有バイナリを使用しているチームでは、プロジェクトの参照で使用できる 2 つの方法と同じものを、バイナリに対しても使用できます。

- **分岐**
- **ワークスペースのマッピング**

分岐

このシナリオでは、共通の共有場所から自身のチーム プロジェクトへバイナリを分岐します。これにより、共有している場所のバイナリとサーバー側のプロジェクトを統一する構成が作成されます。

ただし、新しいバージョンなど、バイナリに対する何らかの変更が、分岐間のマージ プロセスの一環として取り込まれる点が異なります。これにより、共有バイナリにおける変更を取り込むことを、より明白に決定できるようになります。

たとえば、$TeamProject1 と $Common という 2 つのチーム プロジェクトを持っており、

Common に共有バイナリが含まれているとすると、共有場所から、それを参照しているプロジェクトに対して分岐を作成します。TFS のフォルダ構造は、図 6.5 に示すようになります。

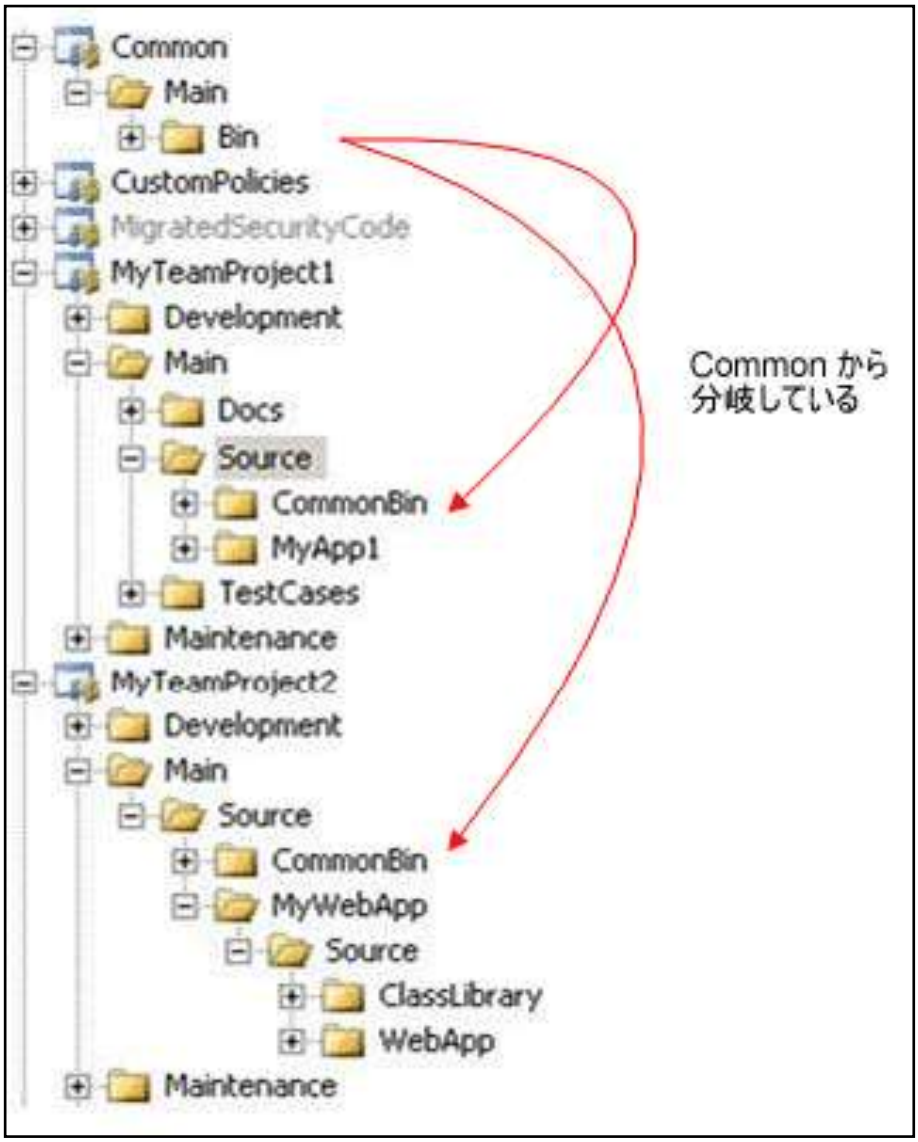


図 *6.5* *Common* からの分岐

ワークスペースのマッピングは以下のようになります。

ソース管理フォルダ	ローカル フォルダ
$/MyTeamProject1/Main	C:¥MyTeamProject1¥Main

クライアント側のワークスペースのフォルダ構造は、図 6.6 に示すようになります。

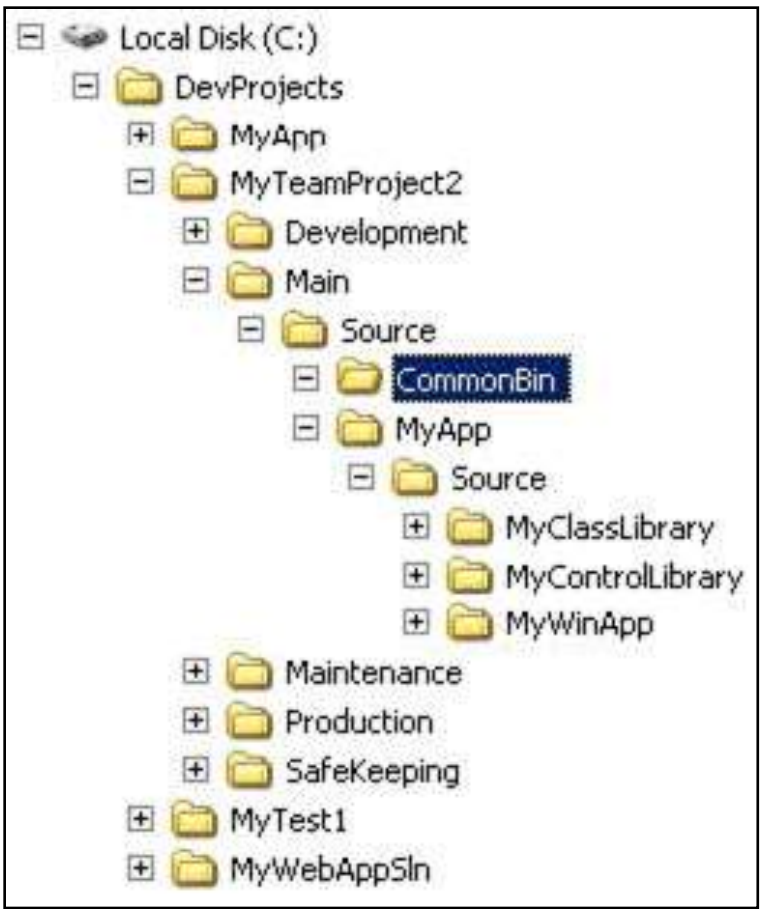


図 *6.6* クライアント側のワークスペース フォルダ構造

ワークスペースのマッピング

共有バイナリを使用するチーム プロジェクトでは、共有場所から、自身の開発コンピュータ上のワークスペースへバイナリを参照します。これにより、共有している場所のバイナリとクライアント側のプロジェクトを統一する構成が作成されます。

このアプローチの長所は、最新のソースをワークスペースに取り込むたびに、共有バイナリの変更も取り込まれることです。

たとえば、$TeamProject1 と $Common という 2 つのチーム プロジェクトを持っており、TeamProject1 が、Common にあるバイナリを参照している場合は、クライアント側のワークスペースのフォルダ構造は、図 6.7 に示すようになります。

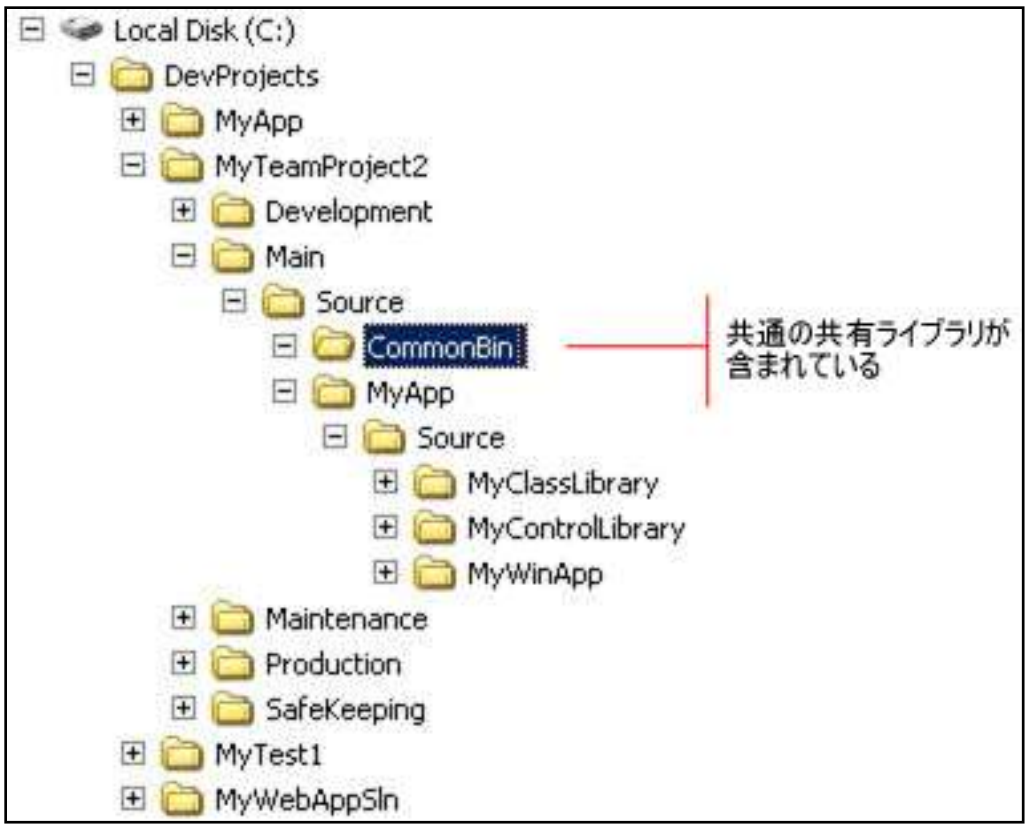


図 *6.7* 共通の共有ライブラリの保存

ワークスペースのマッピングは以下のようになります。

ソース管理フォルダ	ローカル フォルダ
$/MyTeamProject2/Main/	C:¥DevProjects¥MyTeamProject2¥Main¥
$/Common/Main/Bin	C:¥DevProjects¥MyTeamProject2¥Main¥Source¥CommonBin

詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の「Working with multiple team projects in Team Build」を参照してください。

プロジェクトおよびアセンブリの参照のガイドライン

ファイル参照は、次の 2 つの方法で設定することができます。

- .NET Framework アセンブリを参照するには、**[参照の追加]** ダイアログ ボックスの **[.NET]** タブに表示されている一覧からアセンブリを選択します。
- **[参照の追加]** ダイアログ ボックスで **[参照]** ボタンを使用できます。

System.XML.dll などのアセンブリは、グローバル アセンブリ キャッシュ (GAC) にあります。ただし、GAC 内のアセンブリは直接参照できません。かわりに、**[参照の追加]** ダイアログ ボックスの **[.NET]** タブのアセンブリを選択する場合は、実際には、%windir%¥Microsoft.NET¥Framework¥<version>¥ フォルダにあるアセンブリのコピーを参照します。

プロジェクト参照はファイル参照よりも好まれます。アセンブリの参照を管理する場合は、以下のガイドラインに留意してください。

- 可能な場合にはプロジェクト参照を使用する。
- 必要な場合のみファイル参照を使用する。
- プロジェクトおよびファイルの参照に対しては **Copy Local = True** を使用する。

詳細については、このガイドの「ソース管理のガイドライン」を参照してください。

依存関係の自動追跡

ローカル プロジェクトがビルドされるたびに、ビルド システムは、参照先のアセンブリ ファイルの日付と時刻を、開発ワークステーションの作業コピーと比較します。参照先のアセンブリの方が新しい場合は、新しいバージョンがローカル フォルダへコピーされます。このアプローチの長所の 1 つは、開発者によって確立されたプロジェクト参照は、サーバー上のアセンブリの動的リンク ライブラリ (DLL) をロックせず、ビルド プロセスに対して何も干渉しないことです。

Web サービスの参照

システムのすべてのプロジェクトが同じチーム プロジェクトに含まれている、より簡単なシステムでは、すべての開発者は最終的に、すべての Web サービスのローカルな作業コピーを使用します。これは、チーム プロジェクト内の Visual Studio のプロジェクトによって定義されているためです。ソー

ス管理から最初にソリューションをオープンすると、すべてのプロジェクト (Web サービスを含む) がローカルにインストールされます。同様に、他の開発者がソリューションに Web サービスを追加した場合は、ソース管理から次にソリューションを更新するときに、Web サービスをインストールします。このシナリオでは、チーム環境内の一元的な Web サーバーに Web サービスを発行する必要はありません。

より規模の大きいシステムでは、Internet Information Server (IIS) を通じて、一元的にアクセスされる Web サーバー上に Web サービスを発行することが可能で、すべての開発者がローカルに Web サービスをインストールする必要はありません。次に説明するように、開発者は Web サービスを適切に参照する必要がありますが、自身のクライアント プロジェクトから Web サービスへアクセスすることができます。

詳細については、このガイドの「ソース管理のガイドライン」および「ソース管理の実施要綱」を参照してください。

Web サービスの参照における動的な *URL* の使用

Web サービスを呼び出す場合には、最初に、プロジェクトに対して Web 参照を追加する必要があります。これにより、Web サービスとのやりとりで使用するプロキシ クラスが生成されます。プロキシコードには最初、Web サービスの静的な Uniform Resource Locator (URL) (http://localhost or http://SomeWebServer など) が含まれています。

重要事項:コンピュータ上で実行される対象のソリューション内の Web サービスに対しては、http://MyComputerName ではなく http://localhost を常に使用し、すべてのコンピュータ上で参照が正しく保持されるようにします。

プロキシの中に埋め込まれている静的な URL は、通常は、本番環境またはテスト環境で必要としている URL ではありません。一般的には、アプリケーションが開発からテストおよび本番へ移行されるときに、必要な URL が変わります。この問題に対処するには、次の 3 つの方法があります。

- プロキシ クラスのインスタンスを作成するときに、Web サービスの URL をプログラムで設定することができます。
- プロキシの中に URL をハードコーディングしないようにするためのより柔軟なアプローチは、Web サービス参照の **URL の動作**プロパティを「**ダイナミック**」に設定することです。これは、好まれるアプローチです。プロパティを「ダイナミック」に設定すると、アプリケーションの構成ファイル (Web アプリケーションの場合は Web.config、Windows アプリケーションの場合は SomeApp.exe.config) のカスタム構成セクションから Web サービス URL を取得するために、コードがプロキシ クラスに追加されます。
- WSDL.exe コマンド ライン ツールを使用し、/urlkey スイッチを指定して、プロキシを生成することもできます。これは、Web サービスの URL を取得するためにプロキシにコードを追加するという点では **URL の動作**プロパティの設定と同じように機能しますが、この場合には、URL が、アプリケーション構成ファイルの <applicationSettings> セクションに格納されます。

動的 URL のアプローチは、ユーザー構成ファイルを提供することもできます。このファイルは、メインのアプリケーション構成ファイルを上書きすることができます。これにより、別の開発者およびテスト チームのメンバが、Web サービスの参照を別の場所へ一時的にリダイレクトできます。

動的 URL およびユーザー構成ファイルの使用方法

開発環境と本番環境の両方で構成の柔軟性を最大限にするには、Web サービス参照の **URL の動作**プロパティを「**ダイナミック**」に設定します。Visual Studio の既定では、Web 参照が追加されたときに、このプロパティの値が「**ダイナミック**」に設定されます。この値がまだ「**ダイナミック**」に設定されていることを確認するには、次のようにします。

1. ソリューション エクスプローラで、[Web references] を展開します。
2. 一覧の中から各 Web 参照を選択します。
3. 各 Web 参照で、**[URLの動作]** プロパティの値が **[ダイナミック]** に設定されていることを確認します。

ユーザー構成ファイルで Web サービスの URL を指定するには、次のようにします。

1. 最初に Web 参照を追加すると、Visual Studio によって App.config ファイルが自動的に生成されます。このファイルには、Web サービスの参照が含まれており、App.config ファイルの構成の設定は、以下のようになっています。

```
<configuration>
<configSections>
<sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=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>
```

このファイルには、生成されたプロキシで使用される新しい構成セクションが含まれています。この構成セクションには、プロキシの生成時に Visual Studio が検出した Web サービスのアドレスが含まれています。

また Visual Studio は、URL のデフォルトの値を、このプロキシに対して生成されたコードへ配置します。この値は、Settings.Designer.cs というファイルの中に存続します。このファイルを参照するには、次のようにします。

1. **ソリューション エクスプローラ**で、Web サービスを右クリックします。
2. **[オブジェクト ブラウザで表示]** を選択します。オブジェクト ブラウザで、

[YourProject.Properties] というエントリを見つけます。ここで YourProject は Web サービス参照が含まれているプロジェクトの名前です。

3. **[YourProject.Properties]** を展開し、**[Settings]** をダブルクリックします。このようにすると、次のような行が含まれている Settings.Designer.cs ファイルが開きます。

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

これは、構成情報が見つからないときに、Web サービスの URL に対して使用されるデフォルトの値です。

呼び出し先の Web サービスの URL は、たいてい変更する必要があります。たとえば、コンピュータ上でローカルに稼働している Web サービスに対してテストしたり、テスト環境で稼働している Web サービスのバージョンに対してテストしたいことがあります。本番の Web サービスの URL が、開発で使用している URL と違うことはよくあることです。これらの URL をそれぞれ管理するには、URL の値をユーザー構成ファイルの中に指定する必要があります。これは、メインの App.config ファイルから参照されます。ユーザー構成ファイルの名前は任意です。以下の例では、ファイル名として User.config を使用して、ユーザーの構成が指している場所であることを明確にしています。

User.config ファイルを作成するには、次の手順を実行します。

1. ソリューション エクスプローラで、Web サービス参照が含まれているプロジェクトを右クリックし、**[追加]** をポイントしてから、**[新しい項目]** をクリックします。
2. **[アプリケーション構成ファイル]** を選択し、ファイル名を User.config に変更して **[追加]** をクリックします。
3. アプリケーション構成ファイル (App.config) から *<YourProject.Properties.Settings>* 要素を、User.config ファイルへコピーします。このファイルには、ランタイムがリダイレクトされる要素のみが含まれるようにします。<?xml> ディレクティブおよび <configuration> 要素がある場合には削除して、以下の例のようにします。

```
<YourProject.Properties.Settings>
```

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

個々の開発者は、User.config ファイルのコンテンツを、該当する URL の参照で必要とされるとおりに設定します。ここで、構成システムが、この構成データに対して、App.config ファイルではなく、この User.config ファイルを使用するように指定する必要があります。このようにするには、App.config ファイルを以下のように更新します。

1. メインのアプリケーション構成ファイルの *<YourProject.Properties.Settings>* 要素に **configSource="user.config"** 属性を追加します。このようにすると、このセクションから情報にアクセスしたときに、命名されたユーザー構成ファイルへランタイムが暗黙にリダイレクトされます。
2. *<YourProject.Properties.Settings>* 要素のコンテンツを削除します。

App.config は以下のようになります。

```
<?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="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>
```

この例では、*YourProject* は、Web サービス参照が含まれているプロジェクト名です。

重要事項:configSource 属性を使用する場合は、ユーザー構成ファイルが存在しなければなりません。このファイルには、*<YourProject.Properties.Service>* 要素のみを定義します。また、**configSource="user.config"** 属性を追加する場合には、*<YourProject.Properties.Service>* 要素から Extensible Markup Language (XML) のコンテンツを必ず削除します。

User.config のアプローチを使用する場合には、次のようにします。

- User.config ファイルは、必ずアプリケーション コードに従って配置します。ソリューション エクスプローラでこのようにするには、User.config ファイルを右クリックし、**[プロパティ]** オプションをクリックして、**[出力ディレクトリにコピー]** プロパティを **[新しい場合はコピーする]** に設定します。
- User.config ファイルをソース管理に追加しないでください。このようにすると、それぞれの開発者およびテスト チームが、User.config ファイルを使用して、特定の URL を明示的にバインドすることができます。ソース管理には、テスト用や本番用などの他の User.config ファイルが含まれていることがあります。これらのファイルは、テスト環境および本番環境の管理を担当しているユーザーが管理すべきです。これらのテストおよび本番用の User.config ファイルは、Web サービス プロジェクトの一部として保存せず、ソース管理システムの別のエリアに保存する必要があります。
- グローバルな User.config ファイルはソース管理に保存します。このファイルには、(<setting> 要素がない) ルート要素のみが含まれているか、または Web サービスのデフォルトの場所を指定することができます。User.config ファイルは、構成システムが機能するために不可欠なものです。

ヒント: 既定では、ソリューションを追加したときに、ユーザー構成ファイルがソース管理に自動的に追加されます。この処理を回避するには、ファイルを最初にチェックインするときに、User.config ファイルのチェック ボックスをクリアします。ソリューション エクスプローラでファイルを右クリックして **[保留中の変更を元に戻す]** オプションを選択し、ファイルがソース管理の対象にならないようにすることができます。

重要事項:URL を指定する User.config ファイルにルート要素のみが含まれており、User.config に

設定要素がない場合は、Web サービス プロキシは URL のデフォルトの値を使用します。このデフォルト値は、Settings.Designer.cs というファイルの中でプロキシにハード コーディングされています。この値は **DefaultValueSettings** 属性として指定され、以下のようになっています。

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

重要事項:ユーザー構成ファイルを使用する Web アプリケーションでは、ファイルに何らかの変更が行われると、デフォルトで Web アプリケーションが自動的にリサイクルされます。値が変わったときにアプリケーションをリサイクルしたくない場合は、構成セクションの定義に対して、以下のように **restartOnExternalChanges**="**false**" 属性を追加します。

```
<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" requirePermission="false"
restartOnExternalChanges="true"/>
</sectionGroup>
</configSections>
```

この属性を、Web.config ファイルの構成セクションに追加すると、外部の User.config 構成ファイルを変更しても、Web アプリケーションはリサイクルされません。ただし、これらの変更はアプリケーションに検知されます。

このメカニズムを使用している場合には、User.config ファイルが存在しなければならないことを理解することが重要です。本番用のリリース、およびテスト環境用のビルドを作成する場合には、その環境が正しいことを保証する責任者が必要です。このビルドのセットアップの一環として、適切な User.config ファイルがソース管理システムから取得され、MSBuild がそのファイルを見つけられるよう正しい場所へコピーする必要があります。

データベースの参照

接続文字列の形式のデータベース参照は、外部の構成ファイルを使用して管理することもできます。このアプローチの長所は、各開発者が自身の接続文字列を自身専用の User.config ファイルで簡単に指定できることです。単体テストの目的で、接続をローカル データベースへリダイレクトした場合など、ある開発者が何らかの変更を行っても、他の開発者には影響を与えません。

また、ユーザー構成ファイルを使用して、テスト環境で必要な設定など、環境特有の設定を制御することもできます。テスト環境では、テスト データベースを参照する User.config ファイルも使用できます。

この手順は、前述の Web 参照の例と似ていますが、Web 参照の例では、構成ファイルから Web サービスの URL を取得するためのコードが Web サービス プロキシに含まれている点が異なります。データベースの接続文字列では、接続文字列を読み込むためのコードを書く必要があります。

データベースの接続文字列に対するユーザー構成ファイルの使用方法

以下の手順は、ユーザー構成ファイル内でデータベース接続文字列を保存および参照する方法について説明しています。

ユーザー構成ファイルを使用してデータベース接続文字列を保存するには、次のようにします。

1. メインのアプリケーション構成ファイルの <connectionStrings> 要素に **configSource="user.config"** 属性を追加します。

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

2. メインのアプリケーション構成ファイルを上書きするには、(アプリケーションの構成ファイルと同じフォルダ内に) User.config ファイルを作成し、類似の <connectionStrings> エントリをファイルへ追加します。以下の接続文字列は、ローカル データベースを参照することに注意してくださ

い。

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

3. プロジェクト内で、以下のコードを使用して、ユーザー構成ファイルから接続文字列を取得します。このコードは、**System.Configuration.ConfigurationManager** クラスの静的な **ConnectionStrings** プロパティを使用します。WinForm アプリケーションでは、**System.Configuration.dll** に対して参照を明示的に追加する必要があります。

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

4. User.config ファイルは、必ずアプリケーション コードに従って配置します。ソリューション エクスプローラでこのようにするには、User.config ファイルを右クリックして **[プロパティ]** をクリックし、**[プロパティ]** ペインで **[出力ディレクトリにコピー]** プロパティを **[新しい場合はコピーする]** に設定します。

User.config ファイルをソース管理に追加しないでください。このようにすると、それぞれの開発者およびテスト チームが、User.config ファイルを使用して、接続文字列を明示的に指定することができます。ソース管理には、テスト用や本番用などの他の User.config ファイルが含まれていることがあります。これらのファイルは、テスト環境および本番環境の管理を担当しているユーザーが管理すべきです。これらのテストおよび本番用の User.config ファイルは、データベース プロジェクトの一部として保存せず、ソース管理システムの別のエリアに保存する必要があります。

ソース管理では、本番用やテスト用など、使用するそれぞれの環境に対して 1 つの User.config ファ

イルを定義するべきです。これらの構成ファイルは、データベースに対する接続文字列を指定します。User.config ファイルは、構成システムが機能するために不可欠なものです。

ヒント: 既定では、ソリューションを追加したときに、ユーザー構成ファイルがソース管理に自動的に追加されます。この処理を回避するには、ファイルを最初にチェックインするときに、User.config ファイルのチェック ボックスをクリアします。ソリューション エクスプローラでファイルを右クリックして **[保留中の変更を元に戻す]** を選択し、ファイルがソース管理の対象にならないようにすることができます。

このメカニズムを使用している場合には、User.config ファイルが存在しなければならないことを理解することが重要です。本番用のリリース、およびテスト環境用のビルドを作成する場合には、その環境が正しいことを保証する責任者が必要です。このビルドのセットアップの一環として、適切な User.config ファイルがソース管理システムから取得され、MSBuild がそのファイルを見つけられるよう正しい場所へコピーする必要があります。

まとめ

プロジェクトまたはサードパーティのアセンブリを管理する場合には、分岐またはワークスペースのマッピングのいずれかを使用できます。分岐は、依存関係をソース管理のサーバーに保存するため、よく好まれるアプローチです。分岐によって、更新されたバイナリまたはソースを取り入れることを意識的に決断することができます。

ワークスペースのマッピングはクライアント側のみのアプローチで、アプリケーションをビルドするためには、チームの各メンバが自身のコンピュータにマッピングを作成する必要があり、さらにビルド サーバー上にもマッピングを作成しなければならないことを意味します。更新されたバイナリまたはソースをビルドのときに簡単に取り入れたい場合には、このアプローチは最も一般的に使用されます。

Web サービスを参照する場合には動的 URL を使用し、Web サービスを管理する外部の構成ファイルを使用します。このアプローチの長所は、各開発者が自身の Web サービス参照を自身専用の User.config ファイルに簡単に指定できることです。外部の構成ファイルを使用して、接続文字列の形

式のデータベース参照を管理することもできます。このアプローチの長所は、各開発者が接続文字列を自身専用の User.config ファイルに簡単に指定できることです。

参考資料

- プロジェクト参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx 「プロジェクト参照」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- ワークスペースの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181383(VS.80).aspx の「ソース管理ワークスペースの使用」を参照してください。

第 3 部

ビルド

第 3 部の内容

- **チーム ビルドの説明**
- **チーム ビルドでの継続的な統合のセットアップ**
- **チーム ビルドにおけるスケジュール ビルドのセットアップ**

第 7 章 チーム ビルドの説明

目的

- Microsoft® Visual Studio® Team System のチーム ビルド アーキテクチャについて理解します。
- チーム ビルドを構成しているコンポーネントについて学習します。
- チーム ビルドが提供する機能について理解します。
- 適切なビルドの戦略を選択します。
- 大規模なプロジェクトで作業している場合に、ビルド戦略の変更が必要な状態を識別します。

概要

この章では、チーム ビルドを使用してビルド プロセスを自動化する方法について説明します。ここで

は、ビルドに関する多くの一般的な問題について概説し、日ごとのスケジュール ビルドから、継続的な統合ビルドまでのさまざまなアプローチについて比較します。
チーム ビルドは Microsoft Build Engine (MSBuild) の上に構築されています。これを使用して、ビルドに必要なソース コードを取得し、ソリューションをコンパイルし、(必要な場合は) ビルド プロセスの一環として単体テストおよび静的コード分析ツールを実行することができます。また、ビルドの出力を、特定の共有場所へ公開することもできます。
チーム ビルドは、特定のビルドに対して使用したソース コードにビルド番号を付けてラベリングするので、特定のビルドの生成に使用したソースを将来のある時点で抽出することができます。障害が発生した場合は、作業項目を作成し、発生したビルドの障害についてユーザーに通知するようチーム ビルドを構成できます。

この章の参照の仕方

この章を使用して、ビルド プロセスの自動化および管理のためにチーム ビルドが提供する機能について学習し、ビルドをスケジュールするためのさまざまな戦略について理解します。この章を最大限に利用するには、次のようにします。

- **「第 8 章 チーム ビルドでの継続的な統合のセットアップ」**を読んで、Team Foundation Server (TFS) での継続的な統合の使用の詳細について学習します。
- **「第 9 章 チーム ビルドにおけるスケジュール ビルドのセットアップ」**を読んで、スケジュール ビルドの使用の詳細について学習します。
- **付属の「方法」**を読んで、ビルド関連のタスクの実行に役立てます。
    - Visual Studio Team Foundation Server でチーム ビルドを使用してコード分析を自動的に実行する方法
    - Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法
    - Visual Studio Team Foundation Server でスケジュール ビルドをセットアップする方法

アーキテクチャ

このセクションでは、物理アーキテクチャの説明および論理的なワークフローを使用して、チーム ビルドのアーキテクチャについて紹介します。

物理アーキテクチャ

チーム ビルドの物理アーキテクチャは、次のコンポーネントから構成されます。

- **新しいチーム ビルドの種類の作成ウィザード** – クライアント側のこのコンポーネントはチーム エクスプローラからアクセスされ、これを使用して新しいビルドの種類を作成します。
- **チーム ビルド ブラウザ** – クライアント側のこのコンポーネントはチーム エクスプローラからアクセスします。これを使用してチーム ビルドのレポート、およびビルドの進捗情報をチーム エクスプローラから参照することができます。
- **ソース管理 Web サービス** – アプリケーション層のこのコンポーネントは、ビルド サービスがソース コードにアクセスするために使用します。
- **チーム ビルド Web サービス** – アプリケーション層のこのコンポーネントは、クライアントからの要求を受け取り、ビルドの実行ステップを調整します。
- **ビルド サービス** – このサービスは、チーム ビルド Web サービスからの命令に応答してビルドのステップを実行します。ビルド サービスは、別のビルド サーバー、またはアプリケーション層のサーバー上に置くことができます。
- **Team Foundation ビルド ストア** – データ層のこのコンポーネントを使用して、チーム ビルドプロセスに関する記録を保持します。
- **ソース コード データベース** – データ層のこのコンポーネントを使用して、ビルド プロセス中にビルド サービスによってアクセスされるソース コードを保存します。

論理ワークフロー

図 7.1 は、チーム ビルドの論理ワークフローを示しています。

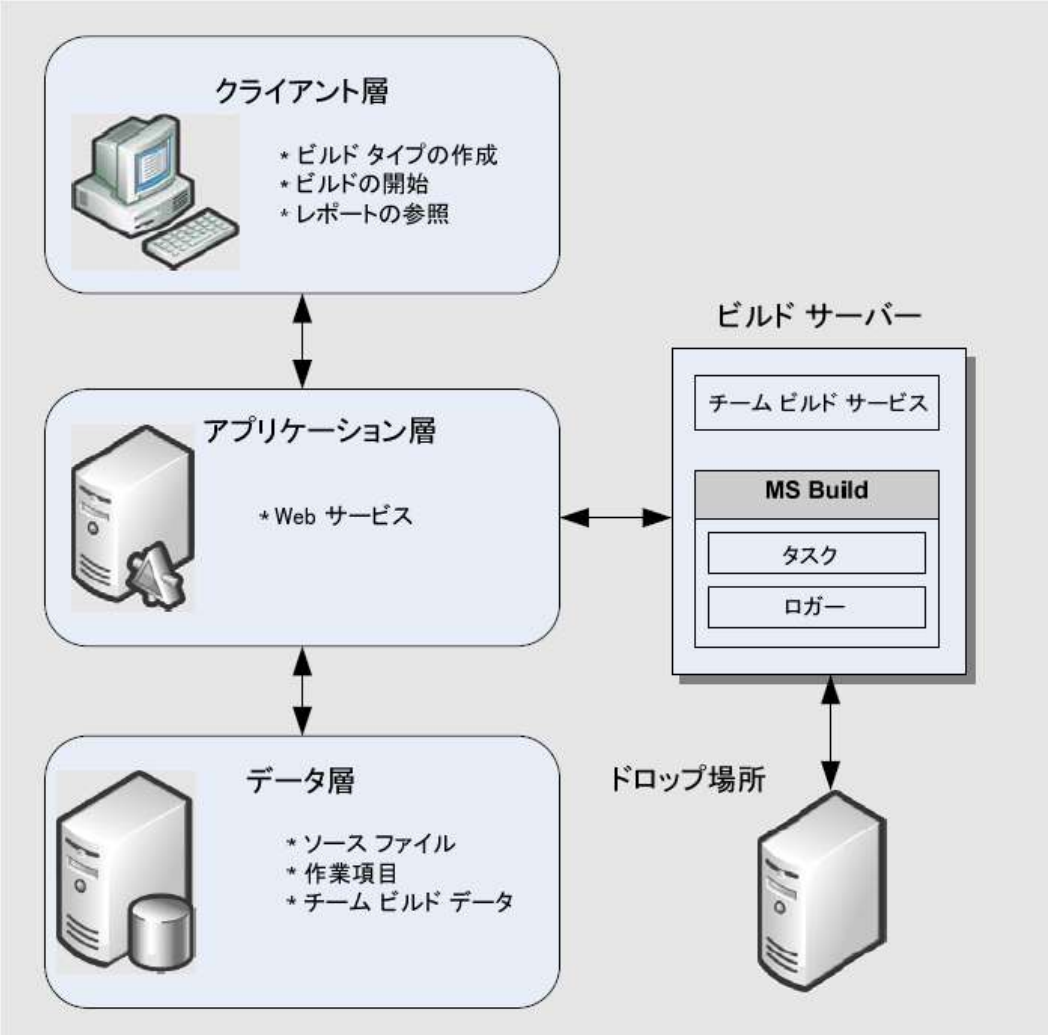


図 7.1 チーム ビルドの論理ワークフロー

チーム ビルドは、アプリケーション層を介して TFS に統合化され、作業項目、コード カバレッジ、コード分析、テスト ケースおよびレポーティングとやりとりします。

TFSBuild.proj ファイルは、ビルドの対象となるプロジェクト、構成、ドロップ場所、コード分析、および実行するテストを含めて、ビルドを制御します。このファイルは、ビルドが最初に作成されるときに、**チーム ビルドの種類の作成ウィザード**によって生成されます。このファイルは直接編集できます。

チーム ビルドは、TFS の中でイベンティング システムを使用します。チーム ビルドによって発生したイベントを使用して、カスタム ビルド ステップを作成したり、ビルド ステータスの変更またはビルドの完了についての通知を生成したりできます。

要点

チーム ビルドの物理アーキテクチャに関して、次の要点に留意してください。

- Team Foundation Build は MSBuild の上に構築されています。
- TFSBuild.proj ファイルには、ビルドのすべての設定が含まれます。**チーム ビルドの種類の作成ウィザード**を使用して、このファイルを生成します。このファイルは直接編集できます。
- TFSBuild.rsp ファイルには、MSBuild のコマンドライン オプションが含まれています。このファイルを修正して、ビルドをさらにカスタマイズできます。
- イベントの通知では、カスタム ビルド ステップを使用することも、**BuildStatusChangeEvent** および **BuildCompletionEvent** イベントによる通知を行うこともできます。
- チーム ビルドは、作業項目、コード カバレッジ、コード分析およびテストケースを統合します。

チーム ビルドの機能

チーム ビルドは、MSBuild ビルド システムの上位層である Team Build サービスで構成されています。MSBuild は ビルドそのものを行う中枢であり、Team Build サービス は、TFS のアプリケーション層との通信を行います。チーム ビルドは Visual Studio のクライアントで作成され、クライアントから、またはスケジューリングされているタスクなどによってビルド サーバー上のイベントから、またはコマンドラインから開始することができます。開始された後のチーム ビルドは、次の手順でプロセスが実行されます。

1. ソース管理からビルド ディレクトリへ、ソースを取り込みます。
2. ソースをコンパイルし、バイナリを生成します。
3. コード分析を実行します (オプション)。
4. ビルドが失敗した場合は、作業項目を作成します。
5. テストを実行します (オプション)。
6. コード カバレッジを計算します (オプション)。
7. ビルドの詳細を記録します。

8. ビルドをドロップ場所へコピーします。

ビルドが完了すると、次のものが使用できるようになります。

ビルドの詳細。クライアントまたはレポートから詳細を参照できます。
ビルドのバイナリ。バイナリはドロップ場所へ配置されます。
ビルドのログ。エラーおよび警告のログを確認できます。
作業項目。ビルドが失敗すると、ビルドの不具合を追跡するために作業項目が作成されます。

ビルド戦略の決定方法

次のステップを使用して、ビルドの戦略を決定します。

1. ビルドの利用者について検討します。
2. ソリューションのシナリオをレビューします。
3. 一般的な問題点について理解します。

ビルドの利用者に関する検討

ほとんどの開発チームには、次に示す 1 人または複数のビルドの利用者がいます。

- 開発チーム
- テスト チーム
- 内部のビルド利用者
- 外部のベータ版利用者
- 顧客

それぞれの利用者は、ビルドの質および頻度について異なるニーズを持っています。通常、ビルドの利用者は、予測可能なスケジュールされたビルドが必要な利用者と、迅速でイベントドリブンなビルドが必要な利用者の 2 つのグループに分類されます。スケジュール ビルドは最も一般的には毎日のビルド

になりますが、これよりも多く行われることも少なく行われることもあります。イベントドリブンなビルドは通常、チェックインによって開始され、開発チームに対して迅速なフィードバックを提供できるように設計されています。開発チームで、ビルドの不具合に関する問題がある場合は、ゲート付きのチェックインモデルの使用を検討します。このモデルでは、ソース ツリーに加えられる前にチェックイン対象物がテストされ、テストにパスしない場合はソース ツリーへの登録が拒否されます。

ソリューション シナリオのレビュー

シナリオの目的とビルドの利用者が、チームの状況にどの程度類似するかに基づいて、ソリューションのシナリオを選択します。迷う場合は、スケジュールされたビルドなど、最もシンプルなビルド シナリオを使用して、必要になった場合にだけさらに複雑なシナリオを追加します。

ビルドのシナリオ

次に、チーム ビルドの最も一般的なシナリオを示します。

- **チームで、毎日のビルドを 1 つ使用する。**ほとんどのチームは、テスト チームおよび他のユーザーに対して整合性のとれたバイナリを提供するために、スケジュール ビルドを利用します。
- **チームで、毎日のビルドと継続的な統合ビルドを使用する。**すべてのチェックインについて、開発者に迅速なフィードバックを提供するために、継続した統合ビルドを利用するチームもあります。比較的規模の小さいチームにとっては、単なる継続的な統合ビルドはうまく機能しますが、比較的規模の大きいチームは、チェックインの頻度が多くなり、ビルド時間が長くなるため、ビルド サーバーの負荷が高くなりがちです。このような場合は、ローリング ビルドを使用して、チェックインとビルドのフィードバックの間で時間をかけすぎずに、ビルドの頻度を少なくすることができます。
- **チームで、複数のビルド ラボを使用し、それぞれのラボで毎日のビルドと継続的な統合ビルドを使用する。**非常に規模の大きいチームでは、ビルドの品質および適時性を改善するために、より複雑なソリューションが必要になってきます。ゲート付きのチェックインを使用し、ソース管理に加える前に不正なチェックインを拒否することによって、ビルドの不具合を少なくすることができます。分岐を使用するチームは、ビルドがそれぞれの分岐の目的に適合するよう、複数の (分岐ごとに 1 つの) ビルド サーバーを使用することができます。たとえば、統合分岐は統合ビルドを作成

し、開発分岐は開発ビルドを作成します。

スケジュール ビルド

スケジュール ビルドは、定期頻度に基づく、時間ベースのビルドです。スケジュール ビルドの目的は、ビルドの品質に関するフィードバックを得るために、整合性のとれた信頼できるビルドを生成することです。通常、スケジュール ビルドは毎日のビルドですが、ニーズに応じてさらに多い、または少ない頻度で行うこともできます。

一般的に、スケジュール ビルドの利用者は次のようになります。

- テスト チーム
- 内部のビルド利用者
- 外部の利用者

スケジュール ビルドを作成するには、次の手順を使用します。

1. コマンドライン ビルドを使用してバッチ ファイルを作成します。
2. Windows スケジュール タスクを使用し、スケジュールに基づいてバッチ ファイルを実行します。

詳細については、「第 9 章 チーム ビルドにおけるスケジュール ビルドのセットアップ」を参照してください。

継続的な統合ビルド

継続的な統合ビルドは、チェックインが発生したときにいつでもトリガされます。継続的な統合ビルドの目的は、チェックインの後できるだけ早く、ビルドの品質に関する迅速なフィードバックを得ることです。開発チームは典型的な継続的な統合ビルドの利用者です。

チームの規模、ビルドの長さ、およびチェックインの数によって、次のいずれかの方針を選択します。

1. 各チェックインの直後に継続的な統合ビルドを使用する。
2. 特定回数のチェックインの後、または一定期間の後 (いずれか早い方) にローリング ビルドを行う。

詳細については、「第 8 章 チーム ビルドにおける継続的な統合のセットアップ」を参照してください。

ゲート付きのチェックイン

ゲート付きのチェックインのプロセスでは、ソース ツリーに対して追加を許可する前に、各チェックインがいくつかの品質規準に合格しなければなりません。ゲート付きチェックインの目的は、各チェックインを承認する前に一連のテストを行うことによって、ビルドの不具合の数を減らし、ビルドの品質を改善することです。

一般的な問題点について

一般的なシナリオの中には、チーム ビルドが既定では完全にサポートできないものもあります。次のシナリオをよく読んで、自分のチームがどのシナリオに当てはまりそうかを検討してください。

- **外部の依存関係を持つプロジェクトをビルドする。**自分のチーム プロジェクトに外部の依存関係を取り込むために分岐を使用している場合は、プロジェクト参照を使用すると、追加の手順なしでビルドがビルド サーバー上で機能します。外部の依存関係を参照するためにクライアント側のワークスペース マッピングを使用している場合は、ビルドを正常に完了するために、ビルド サーバー上にマッピングを保持する必要があります。詳細については、「第 6 章 Visual Studio Team System におけるソース管理の依存関係の管理」を参照してください。
- **セットアップ プロジェクトをビルドする。**チーム ビルドは、既定ではセットアップ プロジェクトをサポートしていません。ビルド後のカスタム ステップを使用して、セットアップ プロジェクトをコンパイルし、ビルドのドロップ場所へバイナリをコピーします。詳細については、http://msdn2.microsoft.com/ja-jp/library/ms404859(VS.80).aspx の「チュートリアル : Visual Studio セットアップ プロジェクトをビルドするために Team Foundation ビルドを構成する」を参照してください。

- **.NET 1.1 アプリケーションをビルドする。**チーム ビルドは、既定では .NET 1.1 アプリケーションをサポートしていません。MSBuild Extras – Toolkit for .NET 1.1 (MSBee) は .NET 1.1 ビルドをサポートしますが、プロジェクトおよびソリューションを Visual Studio 2005 にアップグレードする必要があります。Visual Studio 2005 のプロジェクトおよびソリューションにアップグレードできない場合は、ビルド後のカスタム ステップを使用して、.NET 1.1 アプリケーションをコンパイルすることができます。MSBee をダウンロードするには、http://www.codeplex.com/MSBee へアクセスします。カスタム ビルドステップの作成による .NET 1.1 アプリケーションのコンパイルの詳細については、http://blogs.msdn.com/nagarajp/archive/2005/10/26/485368.aspx の Nagaraju のブログのエントリ、または http://www.codeplex.com/sdctasks の「Microsoft SDC DevEnv task」を参照してください。
- **Microsoft Visual Basic® 6.0 のアプリケーションをビルドする。**チーム ビルドは、既定では Visual Basic 6.0 のアプリケーションをサポートしていません。ビルド後のカスタム ステップを使用して、Visual Basic 6.0 のアプリケーションをコンパイルします。ビルド後のカスタム ステップとして、類似している .NET 1.1 アプリケーションのコンパイルに使用できるステップを参照するには、http://blogs.msdn.com/nagarajp/archive/2005/10/26/485368.aspx の Nagaraju のブログを参照するか、または http://freetodev.spaces.live.com/blog/cns!EC3C8F2028D842D5!261.entry の「MSBuild task to build VB6」を参照してください。

大規模なプロジェクトでの考慮事項

大規模な開発チームで作業している場合は、その他にも留意することがあります。大規模な開発チームは通常、次の点において小規模のチームとの違いがあります。

- より複雑な分岐およびマージの構造が必要になります。
- 複数のソリューションおよびチーム プロジェクト間で依存関係を管理することが多くなります。
- コンポーネントおよびチームに対して複数のビルドを保持することが多くなります。

大規模な開発チームで作業する場合には、次のことに留意する必要があります。

- 大規模なチームでは、ビルドの時間が長くなる傾向にあります。ビルド サーバーのキューが長くなりすぎないように、および負荷が高くなりすぎないように、継続的な統合ビルドの頻度は、ビルド時間より少なくなるようにします。
- チームが分岐している場合は、分岐ごとにビルド サーバーをセットアップできます。このようにすると、各ビルドが分岐の目的 (統合や開発など) に適合するようにできます。
- 統合の分岐は、スケジュール ビルドのみを持つことが多くなります。開発の分岐は、継続的な統合ビルドおよびスケジュール ビルドを持つ場合があります。

ビルドのカスタマイズ

ビルド サーバー、ドロップ場所、ビルド ディレクトリなど、ビルドに関する情報を修正するには、TFSBuild.proj ファイルを編集することができます。

TFSBuild.proj ファイルには、チーム ビルドの実行に必要な多数の情報が含まれています。この情報には、ビルドの場所、およびビルドが静的なコード分析と単体テストを実行すべきかどうか、が含まれています。ビルドを修正するには、TFSBuild.proj ファイルを編集します。ファイルを編集するには、次のようにします。

1. ソース管理からファイルをチェックアウトします。
2. ファイルのビルド情報を更新します。
3. ファイルをチェックインして戻し、変更をコミットします。

ビルドを次に実行するときには、修正されたビルド データが使用されます。
ビルドのカスタマイズの詳細については、このガイドの「Build Guidelines」および「Build Practices」の「カスタマイズ」セクションを参照してください。

まとめ

チーム ビルドは MSBuild の上に構築されています。チーム ビルドは、アプリケーション層を介して

TFS に統合化され、作業項目、コード カバレッジ、コード分析、テスト ケースおよびレポーティングの機能とやりとりします。

ビルドの戦略を決める場合には、ビルドの利用者およびビルドの要件について考慮する必要があります。一般的なビルドの方針には、スケジュール ビルドを使用するものと、継続的な統合ビルドを使用するものとがあります。スケジュール ビルドは、信頼性の高い一貫したビルドのためのもので、テスト チームおよび他のユーザーに使用され、ビルドの品質に関するフィードバックを提供できます。継続的な統合ビルドは、開発チームに対して、ビルドの品質についての迅速なフィードバックを提供します。

参考資料

- チーム ビルドの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181710(vs.80).aspx の「Team Foundation ビルドの概要」を参照してください。

第 8 章 チーム ビルドでの継続的な統合のセットアップ

目的

- 継続的な統合ビルドの目的を理解する。
- Microsoft® Visual Studio® Team System のチーム ビルドを使用して継続的な統合をセットアップする。
- ボトルネックを減らすよう継続的な統合ビルドを最適化する。

概要

この章では、チーム ビルドおよび Microsoft Visual Studio Team Foundation Server (TFS) を使用して、継続的な統合ビルドをどのようにセットアップできるかについて説明します。チェックインの後できるだけ早く、ビルドの品質に関する迅速なフィードバックを得るために、継続的な統合ビルドを使

用します。チーム開発では、特にチェックインの内容が他の開発者による変更と結びついており、ビルドの不具合を生じる場合には、チェックインの品質に関する迅速なフィードバックを得ることが重要です。継続的な統合ビルドによって、コードを迅速に修正して、他のチーム メンバーの障害を取り除くことが可能になり、つまりビルドの整合性および品質が改善されます。

Team Foundation Server は、既定では継続的な統合ビルドをサポートしていません。Microsoft が提供する拡張機能を使用してビルド エンジンを拡張し、継続的な統合をサポートすることは可能です。

この章の参照の仕方

この章では、継続的な統合の戦略と、チーム ビルドによる継続的な統合ビルドのセットアップおよび構成方法について学習します。継続的な統合ビルドのセットアップについて、順を追って理解するには、「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。

TFS およびチーム ビルドを初めてお使いになる場合、またはビルドの自動化およびスケジューリングで使用できるオプションの詳細について学習したい場合は、この章を読む前に、「第 7 章 チーム ビルドの説明」をお読みください。

継続的な統合ビルドの方針

継続的な統合 (CI) は、開発者がソース管理にコードをチェックインするたびに、ビルドを作成するプロセスです。次の一覧は、継続的な統合ビルドで使用されるさまざまな方針を示しています。

- チェックインのたびにビルドします。
- 特定回数のチェックインの後でローリング ビルドを行います。
- 一定時間の後でローリング ビルドを行います。
- 特定回数のチェックインの後、または一定時間の後でローリング ビルドを行います。

チェックインのたびにビルドする

各チェックインの直後にビルドすることは最も簡単な継続的な統合の戦略であり、一般に最速のフィードバックを提供します。ただし、ビルド サーバーで問題が発生するほどチェックインが頻繁になる場合は、ローリング ビルドのアプローチを使用する必要があります。このアプローチでは、特定回数のチェックインの後、または一定期間の後でビルドを行います。ローリング ビルドを使用する必要があるかどうか決定するには、次の内容について判断します。

- チーム ビルドの長さ （分)。
- チェックインの平均頻度 (分)。
- 頻繁なチェックインが発生する時間枠。

ビルドの長さがチェックインの平均頻度よりも長い場合は、最初のビルドが完了しないうちに、他のビルドを開始する 2 番目のチェックインが発生するため、ビルドはキューに並び始めます。ビルドのキューが非常に長くなると、ビルド サーバーのパフォーマンスに影響を与えることがあり、スケジュール ビルドなどの他のビルドが開始できなくなることがあります。頻繁なチェックインが発生している時間枠を見直して、最もビジーなチェックイン期間が終了した後で、キューをクリアする時間があるかどうか判断します。

詳細については、「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。

特定回数のチェックインの後にローリング ビルドする

チェックインのたびにビルドを作成したためにビルド サーバーが過負荷になった場合は、特定回数のチェックインが完了した後でのみビルドすることができます。これは最もシンプルなローリング ビルドのソリューションですが、大きな欠点があります。ビルド サーバーはビルドを開始する前に特定回数のチェックインを確認しなければならないため、その日の最終チェックインは事実上ビルドしないことが保証され、したがってビルドのフィードバックが遅れます。

一定時間の後でローリング ビルドする

特定回数のチェックインの後にビルドを行うことの改良として、それぞれのチェックインの後で、一定の時間が過ぎたときだけビルドを生成することができます。このアプローチでは、各チェックインが行われた時間に対して、一定の期間内にビルドが必ず生成されます。それぞれのビルドに関連するチェックインの数は、さまざまであることに注意してください。つまり、1 回のチェックインだけが含まれるビルドもあれば、それ以上が含まれるものもあります。各ビルドのチェックインの数が多くなるほど、不具合を引き起こす変更をどのチェックインが生成したかを決定することがより困難になります。

特定回数のチェックインの後、または一定時間の後でローリング ビルドする

特定の時間間隔または特定回数のチェックインの後 (いずれか早い方) にビルドを生成すると、ビルド サーバーの負荷を低減させつつ最も整合性のとれたローリング ビルドが実現できます。継続的な統合ビルドで非常に長いビルド キューが生成され、その結果、チェックインからかなり時間が経ってからビルドが発生する場合は、ローリング ビルドを使用します。各ビルド間のチェックインの数を指定するには、チェックイン間隔を使用します。追加のチェックインがなくても、ビルドを確実に行うには、タイムアウト期間を使用します。

ローリング ビルドの間隔の決定

理想的なローリング ビルドの間隔を決定するには、チェックインの平均頻度をビルドの長さで除算します。たとえば、10 分かかるビルドがあり、チェックインを 5 分おきに平均化している場合は、チェックインの間隔を 2 に設定し、タイムアウト期間を 10 分に設定できます。このようにすると、次のビルドが始まるまでに、前のビルドは確実に終了します。ビルド サーバーの負荷が過剰になっていることに気付いた場合は、これらの値を増やすことができます。

Team Foundation Server における継続的な統合ビルド

Team Foundation Server 2005 には、すぐに使用できる継続的な統合ソリューションは提供していませんが、独自の継続的な統合ビルド ソリューションを実装するためのフレームワークを用意しています。

TFS における継続的な統合ビルドのセットアップの詳細については、「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。この方法では、Visual Studio Team System の開発チームによって提供されるソリューションを使用します。このソリューションは、TFS サーバーに対するアクセス権を持つアカウントのもとで実行する Web サービスをインストールします。Team Foundation Server は、特定のイベントが発生したときに電子メールのメッセージを送信したり、Web サービスを呼び出したりできます。このイベントのメカニズムは、継続的な統合ソリューションが **CheckinEvent** に対応した Web サービスを登録するために使用されます。このようにすると、チェックインが発生するたびに、Web サービスがチーム ビルドを開始します。

まとめ

開発者がソース管理にコードをチェックインするたびにビルドを開始するには、継続的な統合ビルドを使用します。チーム ビルドは、すぐに使用できる継続的な統合ソリューションは提供していませんが、ビルドをカスタマイズして、独自の継続的な統合ビルド ソリューションを実装することができます。

特有のプロジェクト要件によって、継続的な統合ビルドを次のように設定することができます。

- チェックインのたびに継続的な統合ビルドを行う。
- 特定回数のチェックイン、または特定の時間間隔の後 (いずれか早い方) で、ビルド サーバーの負荷を軽減するためにローリング ビルドを実行する。

参考資料

- Visual Studio Team System の継続的な統合ソリューションの使用方法の詳細については、http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx 「Continuous Integration Using Team Foundation Build」を参照してください。
- Visual Studio Team System の継続的な統合ソリューション MSI をダウンロードするには、http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi へアクセスしてください。

- Team Foundation Server におけるアジャイル開発および継続的な統合の詳細については、http://www.microsoft.com/japan/msdn/msdnmag/issues/06/03/TeamSystem/default.aspx の「常時結合を可能にするための Team Foundation Server の拡張」を参照してください。

第 9 章 チーム ビルドにおけるスケジュール ビルドのセットアップ

目的

- スケジュール ビルドの目的を理解する。
- Microsoft® Visual Studio® Team System のチーム ビルドを使用してスケジュール ビルドをセットアップする。

概要

この章では、チーム ビルドおよび Microsoft Visual Studio Team Foundation Server (TFS) を使用して、スケジュール ビルドをどのようにセットアップできるかについて説明します。スケジュール ビルドの目的は、一貫したスケジュールに基づいて、信頼性の高いビルドを作成するプロセスを自動化することです。これは、テスト チーム、内部の利用者、および外部のベータ ユーザーがもっともよく使用するビルドです。

スケジュール ビルドは、ビルドの自動化における最も簡単な形式です。スケジュール ビルドは、時間、日、週の単位、またはチームにとって最適に機能する任意の間隔で実行することができます。

この章の参照の仕方

この章を使用して、スケジュール ビルドの方針について、およびチーム ビルドを使用してスケジュール ビルドをセットアップし、構成する方法について学習します。スケジュール ビルドのセットアップについて、順を追って理解するには、「Visual Studio Team Foundation Server でスケジュール済みのビルドをセットアップする方法」を参照してください。

TFS およびチーム ビルドを初めてお使いになる場合、またはビルドの自動化およびスケジューリングで使用できるオプションの詳細について学習したい場合は、この章を読む前に、「第 7 章 チーム ビルドの説明」をお読みください。

開発チームがチェックインするコードの品質によってビルドが不安定になることを懸念する場合は、継続的な統合ビルドの使用を検討してください。継続的な統合の詳細については、「第 8 章 チーム ビルドにおける継続的な統合のセットアップ」を参照してください。

スケジュール ビルドの頻度の戦略

ビルドの頻度は、スケジュール ビルドを作成するときに決定する最も重要な事項の 1 つです。ビルドは、時間ごと、夜ごと、または週ごとのいずれかでスケジュールすることができます。

時間ごとのビルド

1 時間以内に多数の変更を生じさせる程のチェックインがあるプロジェクトで作業しており、かつ継続的な統合ビルドを採用しない場合は、1 時間ごとのビルド頻度を選択することができます。1 時間ごとのビルドでは、開発者に対して迅速なフィードバックが提供され、テスト担当者および他のチームのメンバーもフィードバックを要求できるようになります。

夜ごとのビルド

このビルドは最も一般的なスケジュール ビルドの頻度です。なぜならば、前日の変更点を取り込み、かつテストの準備が整った新しいビルドを、毎朝テスト チームと開発チームに提供するからです。

週ごとのビルド

ビルド期間が何日もかかるような大規模で複雑なプロジェクトで作業している場合には、週ごとのビルドを選択します。これは、前の週からの変更点を取り込み、かつテストの準備が整ったビルドが、テスト チームに毎週の初めに提供されることを保証します。

Team Foundation Server におけるスケジュール ビルド

TFS におけるチーム ビルドの機能は、ユーザー インターフェイスからのスケジュール ビルドをサポートしていません。代わりに、Microsoft Windows® タスク スケジューラ を使用して TFSBuild のコマンド ユーティリティを実行し、あらかじめ決められた時間にビルドを開始することができます。

スケジュール ビルドを作成するには、次の手順を使用します。

1. TFSBuild のコマンドラインを作成します。

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

2. バッチ ファイルの中にコマンドラインを配置します。Windows のコマンド プロンプトから実行できるように、TFSBuild.exe ファイルに対するフル パスを必ず指定してください。バッチ ファイルで使用するコマンドの例は次のようになります。

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

3. 望ましい間隔でバッチ ファイルを実行する、Windows スケジュール タスク を作成します。

詳細については、「Visual Studio Team Foundation Server でスケジュール ビルドをセットアップする方法」を参照してください。

まとめ

テスト チーム、またはビルドの品質に関するフィードバックを提供可能な他の利用者に対して整合性のあるビルドを提供するには、スケジュール ビルドを使用してください。Team Foundation Server は、ユーザー インターフェイスからのスケジュール ビルドをサポートしていません。代わりに、Windows タスク スケジューラ を使用して TFSBuild のコマンド ユーティリティを実行し、あらかじめ決められた時間にビルドを開始することができます。

スケジュール ビルドは、時間、日、週の単位、またはプロジェクトの要件にあった任意の間隔で実行

することができます。

参考資料

- スケジュール ビルドのセットアップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181727(VS.80).aspx の「方法 : スケジュールされたビルドを構成する (コマンド ライン)」を参照してください。

第 4 部

大規模なプロジェクトでの考慮事項

第 4 部の内容

- **大規模なプロジェクトでの考慮事項**

第 10 章 大規模なプロジェクトでの考慮事項

目的

- Microsoft® Visual Studio® Team System Team Foundation Server (TFS) における大規模なプロジェクトのワークフローを理解します。
- 大規模なチームに対するソース管理およびビルドを最適化する方法について学習します。
- 大規模なプロジェクトがソース管理にどのような影響を与えるかについて学習します。
- 大規模なプロジェクトに関係している場合に、分岐およびマージの方針をどのように変更しなければならないかを学習します。
- 大規模なプロジェクトがビルド戦略にどのような影響を与えるかについて学習します。

概要

この章では、TFS を使用した大規模な開発に対する特別な考慮事項について説明します。大規模プロジェクトは一般に、次の点で小規模プロジェクトと異なります。

- より複雑な分岐およびマージの構造が必要になります。
- 複数のソリューションおよびチーム プロジェクト間にわたる多数の依存関係を扱わなければなりません。
- コンポーネントおよびチームに対して複数のビルドを保持する可能性が高くなります。

たとえば、大規模なプロジェクトでは、複数の機能チームの並行開発をサポートするために、複数の分岐をサポートしなければならない場合があります。このようなシナリオでは、複数のソリューションおよびチーム プロジェクト間にわたって依存関係を管理し、共通の Web サービスおよびデータベースを共有しなければならないことが多くなります。各サブチームは、自身のビルド サーバーを保持し、特定のドロップ ポイントに対してビルドを出力しなければならないことがあります。

この章の参照の仕方

大規模な開発プロジェクトについて管理、サポート、または関与する場合は、この章を使用します。第 3 章、5 章、および 7 章にも、個別の「大規模なプロジェクトでの考慮事項」を説明しているセクションがあります。大規模なプロジェクトの考慮事項のすべてをまとめてレビューするには、この章を使用してください。

大規模なプロジェクトの論理ワークフロー

図 10.1 は、複数のサブチームが共同で作業し、1 つの複雑なアプリケーションを作成する場合の一般的なシナリオについて示しています。

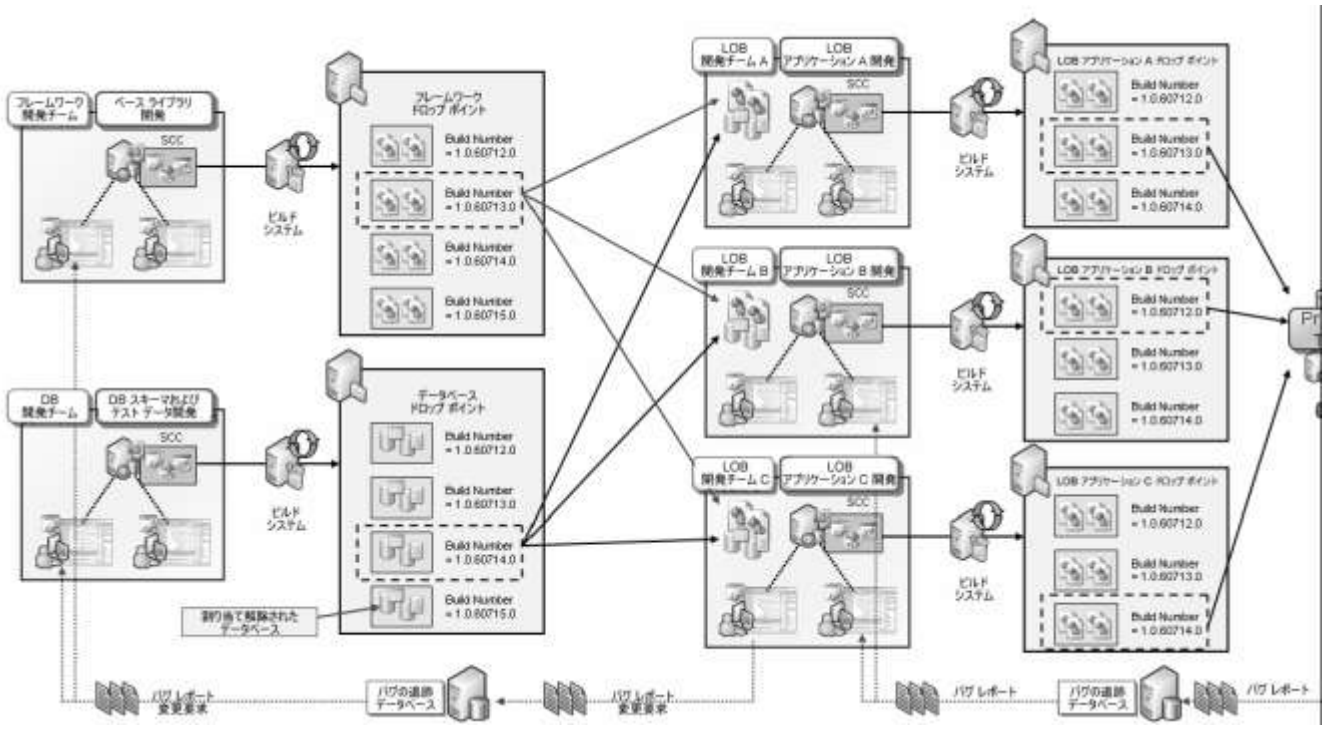


図 *10.1* 大規模なプロジェクトのシナリオ

図 10.1 は、大規模なチーム プロジェクトの運用方法に関するいくつかの重要なポイントを強調しています。

- 各サブチームは、自身のビルド サーバーを保持し、ドロップ ポイントに対してビルドを出力します。
- アプリケーション チームはコンポーネント チームからビルドを取得し、ソリューションの一部としてそれらを再利用して、自身のビルドの中に取り込みます。
- コンポーネント テストおよび統合テストは各ビルドで発生します。バグは、適切なバグ データベースへファイルされます。

ソース管理の考慮事項

数十ものプロジェクトが必要な大規模なソリューションで作業している場合に、Visual Studio ソリューション (.sln) の拡張性の限界に直面することがあります。このため、ソース管理の構造はサブシステムごと、またはサブアプリケーションごとに編成して、複数のソリューション ファイルを使用する必要があります。アプリケーションは複数のソリューションに分割しますが、全体のアプリケーション

に対して 1 つのマスター ソリューションは作成しないでください。図 10.2 は、大規模なチームで一般的に使用される複数ソリューションのアプローチを示しています。

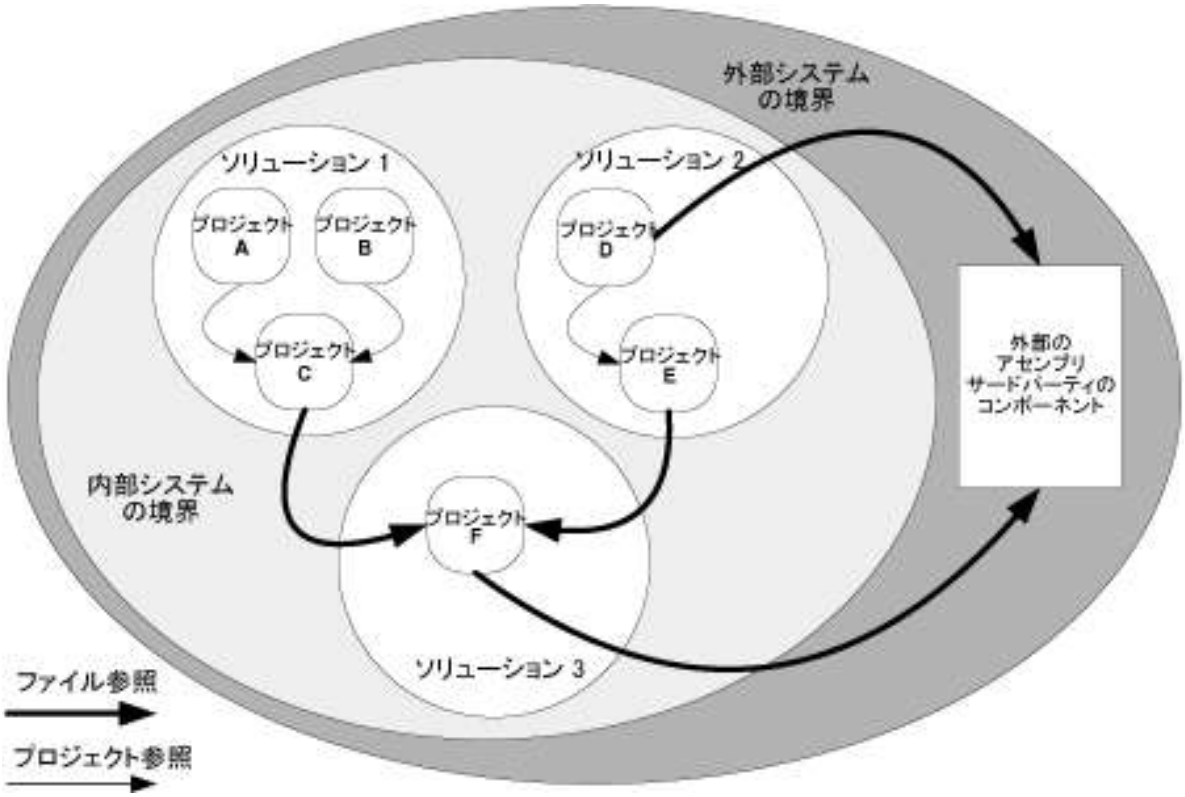


図 *10.2* 複数ソリューションのアプローチ

このシナリオでは、各ソリューション内のすべての参照はプロジェクト参照です。各ソリューションの外部のプロジェクトに対する参照 (サードパーティのライブラリや、別のサブソリューションのプロジェクトなど) はファイル参照です。これは、このような場合には "マスター" ソリューションがあり得ないことを意味しています。代わりに、ソリューションをビルドする必要がある順序を示すスクリプトを使用する必要があります。

複数ソリューションの構造に関するメンテナンス タスクの 1 つは、開発者がソリューション間で循環参照を不注意に作成していないことを確実にすることです。この構造では、複雑なビルド スクリプト、および依存関係の明示的なマッピングが必要です。この構造では、Visual Studio でアプリケーション全体をビルドすることはできません。代わりに、TFS チーム ビルドを使用します。

この複数ソリューションのアプローチの詳細については、「第 3 章 ソース管理におけるプロジェクト

とソリューションの構造化」を参照してください。

分岐とマージに関する考慮事項

大規模なチームは、チームと機能の両方で分岐することが多くなります。また、外部の依存関係を統合することを意図して、1 つ以上の分岐を持つことも多くなります。このようにすると分岐の構造がより深くなるため、開発の分岐における変更と、その変更をメインの統合分岐へマージするまでの時間差が広がります。この理由から、分岐を作成する場合には、マージ戦略について慎重に検討する必要があります。

たとえば、スケジュールに基づいてマージするか、イベント ドリブンでマージするかについては、次の内容を考慮します。

- 逆統合は、通常はスケジュールに基づいて行われます。たとえば、開発の分岐からメインの統合分岐へマージする場合です。
- メインの統合分岐から開発分岐へ統合を行うような前方統合は、通常、機能のマイルストーンを達成したなどのイベントに基づいて行われます。このイベントは、開発分岐の後のチームが、親分岐における変更をマージする準備ができたと思ったときに発生します。

分岐/マージの戦略は、どのくらいの頻度でビルドを生成したいか、によって論理的に理由付けします。分岐構造が深くなると、子分岐からメインの統合分岐へマージする時間が長くなります。開発チームにとって問題が発生しそうな兆候としては、次のものが考えられます。

- 分割されたビルドで、修正をメインの統合分岐へ反映させるのに長い時間がかかる。
- 機能をメインの統合分岐へ反映させるのに長い時間がかかるために、マイルストーンがない。
- それぞれの分岐における変更をマージするのに膨大な時間がかかる。

これらのことがチームで問題になる場合は、分岐構造の深さを減らすことを検討してください。

次に、大規模なチームの分岐構造の例を示します。

- **My Project**
  - o **Development** – 現在有効な開発分岐を分離するためのコンテナ
  - •□**Team 1**
  - **Feature A** – 開発用に分離された分岐
  - o **Source**
  - **Feature B** – 開発用に分離された分岐
  - o **Source**
  - •□**Team 2**
  - **Feature A** – 開発用に分離された分岐
  - o **Source**
  - **Feature B** – 開発用に分離された分岐
  - o **Source**
  - o **Main** – メインの統合およびビルドの分岐。すべての変更がここに収集される。
  - •□**Source**

•□**Other Asset Folders**

- o **Releases** – リリースの分岐のコンテナ
  - •□**Release 4 –** リリースのためにロックダウンされているコードが含まれている分岐
  - **Source**
  - •□**Release 2**– 有効な保守の分岐
  - **Source**
  - •□**Release 3**– 有効な保守の分岐
  - **Source**
  - o **Safe Keeping**
  - •□**Release 1** – 安全な保管のための以前のリリース
  - **Source**

この構造には、次のものが含まれています。

- 複数のチームが互いに分離して作業するための機能の分岐。

- すべてのチームにおける変更、およびリリースの分岐におけるバグの修正を収集するためのメインの分岐。
- 対象リリースのロックダウンで使用されるリリースの分岐。
- 現在も維持およびサポートの対象となっている以前のリリースのための、保守の分岐。
- 現在は維持されていない以前のリリースのための、安全管理の分岐。

ビルドに関する考慮事項

大規模なチームでは、ビルドの時間が長くなる傾向にあります。ビルド サーバーのキューが長くなりすぎないように、および負荷が高くなりすぎないように、継続的な統合ビルドの頻度は、ビルドの時間より少なくなるようにします。大規模なチームのビルド システムは通常、次のように編成されます。

- チームで、複数のチームまたは機能の分岐を使用している場合は、各分岐に対して 1 つのビルド サーバーを専用で使用します。このようにすると、チームで各ビルドを分岐の目的 (統合や開発など) に適合させることができます。
- ビルドの頻度は、チームのニーズに基づいて決定します。ビルドのインフラストラクチャは、これらのニーズをサポートするよう設計および構築されています。
- ビルドの頻度が最初に選択され、その目標に基づいてマージの頻度が決定されます。
- スケジュールされたビルドは通常、統合分岐に対して使用します。
- 継続的な統合とスケジュール ビルドの組み合わせは、開発分岐に対して使用します。統合ビルドは、統合テスト チームに対して配布されます。開発分岐のスケジュール ビルドからの出力は、関連するテスト チームへ渡され、開発分岐の継続した統合ビルドは、特定の開発チームに特有のフィードバックを提供します。

もう 1 つの重要な問題は、各統合ポイントにビルド、テスト、およびクオリティ ゲートを持たせるべきかということです。あるいは、これらのものをメインの統合分岐に持たせることもできます。理想的には、各分岐に対して 1 つのビルド サーバーを持っており、サポートしている開発およびテスト チームに従って 1 つのクオリティ ゲートがあるとよいでしょう。そうはいっても現実的になるべきで、この選択肢が取れない場合は、クオリティ ゲートを削除するか、または分岐を削除して、構造を簡単にした方がよいでしょう。

まとめ

数十ものプロジェクトが必要な大規模なソリューションで作業している場合に、Visual Studio ソリューション (.sln) の拡張性の限界に直面することがあります。このような場合は、ソース管理の構造はサブシステムごと、またはサブアプリケーションごとに編成して、複数のソリューション ファイルを使用する必要があります。

規模の大きいチームでは、分岐の構造が深くなるため、開発の分岐で発生する変更と、その変更がメインの統合分岐にマージされるまでの時間差が広がります。

大規模なチームでは、ビルドの時間が長くなる傾向にあります。ビルド サーバーのキューが長くなりすぎないように、および負荷が高くなりすぎないように、継続的な統合ビルドの頻度は、ビルドの時間より少なくなるようにします。このようにしても、ビルド サーバーのパフォーマンスでまだ問題がある場合は、ソース管理の構造で各分岐に対して 1 つのビルド サーバーを使用することを検討してください。

参考資料

- Team Foundation のソース管理の詳細については、http://msdn2.microsoft.com/en-us/library/ms364074(VS.80).aspx の「Using Source Code Control in Team Foundation」を参照してください。
- ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- 分岐とマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。

- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。
- チーム ビルドの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181710(vs.80).aspx の「Team Foundation ビルドの概要」を参照してください。

第 5 部

プロジェクト管理

第 5 部の内容

- **プロジェクト管理の説明**
- **作業項目の説明**

第 11 章 プロジェクト管理の説明

目的

- Microsoft® Visual Studio® Team System (VSTS) がプロジェクト マネージャに提供する機能について学習します。
- チーム プロジェクトの作成方針を選択します。
- Visual Studio Team System を使用してプロジェクトを作成および管理します。

概要

この章では、Visual Studio Team System が提供するプロジェクト管理の機能について紹介し、ソフトウェア開発プロジェクトの管理に関する多くの一般的な問題および課題の解決をこれらの機能がどのように支援するかを説明します。

Visual Studio Team System および Team Foundation Server (TFS) には、ソフトウェア開発プロ

セスの監視およびサポートを支援するためのツール、テンプレートおよびレポートが用意されています。結果として、チームの連携がよりスムーズになり、チームのメンバ間で必要な伝達が自動化され、タスクなどの作業項目の割り当ておよび追跡が簡単になり、そしてプロジェクトの状態および進捗を示すメトリックの監視が簡単になります。

この章の参照の仕方

この章を使用して、特にプロジェクト マネージャに関連するTFS および VSTS の詳細な機能を学習します。

- TFS のプロジェクト管理で解決しようとしている問題を理解するには、「プロジェクト管理の概要」および「従来のプロジェクト管理の問題」のセクションをお読みください。
- チーム プロジェクトの作成および編成の戦略を特定するには、「チーム プロジェクトの戦略」のセクションをお読みください。
- 残りのセクションを使用して、プロセス テンプレート、作業項目、およびプロジェクト管理のその他の機能について学習します。

プロジェクト管理の概要

通常、プロジェクトの計画には、次のような手順が含まれます。

1. **ビジョン ステートメントを生成する。**この手順では、プロジェクトの全関係者が共有する、プロジェクトの望ましい最終結果の見通しを作成します。
2. **シナリオを生成する。**この手順では、ソフトウェアの使用シナリオの初期セットを決定します。この中では、お客様からの入力を使用します。また、シナリオを検証し、シナリオがお客様にとって価値があることを確認します。
3. **シナリオをサポートする機能セットを生成する。**この手順では、シナリオをお客様に対する価値の詳細項目に細分化し、それらの項目に関するお客様の期待についてお客様に説明できるようにします。
4. **作業項目セットを生成する。**この手順では、シナリオおよび機能を詳細なタスクに分割します。言い換えると、作業項目が完了したときに、関連する機能またはシナリオが実装されます。
5. **タスクをエリアに分割する。**この手順では、タスクをエリアに分割します。これらの区分は機能上

のものであるか、または特定のチームの編成方法に基づくかのいずれかであることができます。

6. **作業をスケジューリングする。**この手順では、すべての作業を事前にスケジューリングするか、または作業をイテレーションに分割します。

従来のプロジェクト管理の問題

今日の大部分のプロジェクト管理者は、様々なツールを使用してソフトウェア開発プロセスを管理しており、それらのツールは通常、ソフトウェア開発者が自身の仕事に用いるツールと、統合されているとしてもそれはわずかです。このようにツールおよびチームの全体にわたって統合性および結束性が不足すると、プロジェクトマネージャは重大な問題に直面します。一般的な問題としては、次のものがあります。

- **異種の情報ソースを取り扱う。**通常、プロジェクト管理ツールは独立して使用されるため、容易に統合できない別々の情報ソースが生じます。また、有意義なメトリックを生成するために、プロジェクト管理ツールによる管理データを、別のバグ追跡システム内部で追跡したバグなどの他チームメンバーによる管理データと統合することは、通常は困難です。
- **プロジェクト関連のメトリックを取得するのが難しい。**プロジェクト関連のメトリックを取得することは、ステータスを追跡する、十分な知識に基づいた意思決定を行う、「プロジェクトは期限どおりに、予算内で実現できるか?」などの基本的な質問に答える、といったことにとって非常に重要です。これらの重大な問題に答えるために、プロジェクト マネージャは通常、Microsoft Office Project で取得したデータや、開発チームおよびテスト チームが使用しているバグ追跡システムから取得したデータを当てにします。これら異種システムのデータ統合は、困難で時間がかかり、エラーが発生しやすくなります。ツールで生成されたほとんどのメトリックは、保存またはアクセスの方法が統一されていません。レポートの作成は通常、異なるツールの情報をたくさんコピー アンド ペーストしなければならないため、非常に手間がかかります。
- **要件を満たしていることを保証するのが難しい。**開発チームに対して計画された作業と、お客様の要件、およびシステムに対して特定された機能以外の主要な要件の間には、しばしばギャップがあります。また、スケジュールした作業と実際の作業の間にも別のギャップがあります。不可欠な情報がこれらのギャップにおいて失われてしまい、要件が満たされない結果になります。
- **プロセスとプロセスの変化を管理する。**チームが従うべきプロセスを伝えることは、困難なタスク

です。組織的な問題に対処するために、チームの生産性に影響を与えずに変更を加えることはさらに困難です。

- **監査可能な伝達およびタスク追跡が不足している。**コラボレーションおよびチームの結束性は通常、チーム ミーティングを開催することにより、および開発者にタスク リストを割り当てて開発者が正しい優先事項に集中することを支援することで解決します。個々のタスクの進捗を追跡することは、困難です。また、プロジェクト マネージャはしばしば、さまざまな計画およびリストからステータス情報を収集するのに、多くの貴重な時間を費やします。チームのメンバも、ステータス レポートを完成させ、さまざまなドキュメントやフォームを更新するのに時間を費やします。
- **品質保証。**生成されているソフトウェアのバグの数および重大度を予測することは難しいため、スケジュールの見積もりおよびコストは通常、「最良の推測」となっています。従来、これらの見積もりでは、不測の事態の対策を考慮しており、その度合いは、プロジェクトの現在の状態について、プロジェクト マネージャがどのくらい自信を持っているかによって決まります。

VSTS は、プロジェクト マネージャが直面するこれら従来の問題の多くの解決を支援するよう設計されています。VSTS は、統合ツールのセットを提供して、チームがソフトウェア開発活動を改善することを支援し、かつプロジェクト マネージャがソフトウェア開発プロセスをより良くサポートすることを支援することで、これらを行っています。

Team Foundation Server におけるプロジェクト管理機能

Visual Studio Team System が提供するプロジェクト管理の主な機能は次のとおりです。

- **プロセス管理。**Team Foundation Server のプロセス管理には、Microsoft Solution Framework (MSF) のプロセス ガイダンス、およびプロセス テンプレートが含まれています。プロセス テンプレートは、作業項目の種類、レポート、プロジェクトの SharePoint ポータル、およびソース管理の設定を伴って新しいチーム プロジェクトをセットアップします。
- **セキュリティとアクセス許可。**新しいプロジェクトには、デフォルトのグループおよびアクセス許可が含まれており、これらは開発チームの一般的なロールへマップされています。
- **作業項目の一括管理。**バグ、リスク、タスク、シナリオ、サービス品質 (QoS) 要求を含む作業項目は、 TFS の作業項目データベースの中に一括して記録、管理および保持されます。それらを一括して格納することで、すべてのチーム メンバがそれらを簡単に参照およびアクセスすることが

できます。

- **Microsoft Office Excel® と Microsoft Office Project の統合。**プロジェクト マネージャは Office Excel と Office Project の統合機能を使用して、作業項目リポジトリおよびスケジュール情報に対して、使い慣れているツールでそのままアクセスすることができます。
- **メトリックとレポーティング。**TFS には、レポーティング サービスが用意されています。これは、作業項目、ビルド結果、テスト結果などの運用データを、TFS のデータ ウェアハウスに格納されるメトリックに変換するものです。あらかじめ定義されているレポートを使用して、さまざまなプロジェクトの状態および品質のメトリックを問い合わせできます。
- **プロジェクト ポータル。**すべてのチーム プロジェクトに対して、TFS は Microsoft Windows SharePoint® Services を使用する関連プロジェクト ポータルを作成します。このポータルを使用して、プロジェクト関連のドキュメントを管理すること、主要なレポートをすばやく参照すること、プロジェクトの現状を評価することができます。

利点

TFS のプロジェクト管理機能を使用すると、次の利点があります。

- 一括管理
- 高度なトレーサビリティ
- プロジェクト計画とスケジュールの統合
- プロセス制御の改善
- チームのコミュニケーションおよび結束性の改善
- 正確な進捗レポート

Team Foundation Server によるプロジェクトの作成および管理

次に、Team Foundation Server でチーム プロジェクトを作成するための一般的なアプローチを要約した手順を示します。

1. **プロセス テンプレートを選択します。**デフォルトのテンプレートをそのまま使用することも、テンプレートを自身でカスタマイズすることもできます。
2. **チーム プロジェクトを作成します。**チーム プロジェクトは、プロセス テンプレートがベースにな

ります。

3. **チーム プロジェクトに適切なグループおよびメンバを追加します。**
4. **プロジェクトに対するイテレーション サイクル期間を決定します。**これは、チーム プロジェクトにおける各イテレーションの作成を含みます。
5. **作業項目としてシナリオを TFS 内に取り込みます。**
6. **各イテレーションでどのシナリオを完成させる必要があるのか判断します。**
7. **サービス品質 (QoS) 要求を定義します。**サービス品質 (QoS) 要求をシナリオにリンクさせます。
8. **シナリオをいくつかのストーリーに分割して、開発者が管理できる作業項目を提供します。**開発者から、各作業項目に対する見積もりを入手します。
9. **プロジェクト スケジュールを作成します。**(Microsoft Project を使用して) プロジェクトのスケジュールを作成し、それを Team Project へ追加します。
10. **受け入れ基準を定義します。**作業項目に対して受け入れ基準を定義します (QoS 要求と相互に関連します)。
11. **レポーティングの要件を定義します。**プロジェクトの主要なパフォーマンス指標を定義し、レポーティングの要件を特定します。

チーム プロジェクトの作成および管理の詳細については、「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。

チーム プロジェクトに関する戦略

TFS で作業を開始するには、少なくとも 1 つのチーム プロジェクトが必要です。プロジェクト マネージャが新しいチーム プロジェクトを作成すると、関連するチーム プロジェクトの Web サイトが作成されます。このサイトには、ドキュメント ライブラリ、およびドキュメント テンプレートとストック レポートが含まれています。プロジェクトのすべての成果を追跡するために作業項目データベースが作成され、メソドロジ テンプレートがインストールされます。このテンプレートでは、ルール、ポリシー、セキュリティ グループを決定し、すべての作業成果に対するクエリを決定します。また、ソース管理に対してソース コードの分岐が作成されます。

チーム プロジェクトに対して選択する構造は、要件に従うべきものであり、ソフトウェア開発プロセスが進むにつれて変わる可能性があります。新しいチーム プロジェクトの作成に対しては、一般的に

次の 3 つの戦略があります。いずれか 1 つの戦略を使用することも、いくつかを組み合わせて使用することもできます。3 つの一般的な戦略は次のとおりです。

- **アプリケーションごとのチーム プロジェクト**
- **リリースごとのチーム プロジェクト**
- **チームごとのチーム プロジェクト**

アプリケーションごとのチーム プロジェクト

これは、チーム プロジェクトを作成するための最も一般的な方針です。このアプローチは、大規模なアプリケーションと小規模なアプリケーションの両方において、さらに並行に開発されている複数アプリケーションのリリースに対しても有用です。このアプローチを使用して、開発中の各アプリケーションに対して 1 つのプロジェクトを作成します。

リリースごとのチーム プロジェクト

このアプローチは、長期におよぶプロジェクトで作業する、大規模なチームにおいて有用です。それぞれのメジャー リリースの後で、新しいプロジェクトを作成し、新たにスタートします。このアプローチでは、作業項目を含む以前のリリースのお荷物を持ち越す心配がありません。また、このアプローチは、プロセス テンプレートを改善する機会、または新しく獲得した経験や知識に基づいて新しいテンプレートを使用する機会を与えます。

チームごとのチーム プロジェクト

このアプローチは、一括管理およびアクティビティの監視が重要な、複数のチームにまたがる大規模なプロジェクトにおいて有用です。このアプローチを使用して、各チームに対して 1 つのプロジェクトを作成します。このアプローチでは、チームと、TFS の作業項目の種類で定義されているワークフローを密に連携させて、チーム全体に対して統一したレポーティングを提供します。

プロセス管理

VSTS では、ソフトウェア開発ライフサイクルは、ソフトウェア プロジェクトの開発作業をサポートするツール選択の不可欠な部分となっています。ライフサイクル プロセスを VSTS に統合することによって、開発チーム内のやりとりに関するプロセスおよび伝達は、ツール選択により完全にサポートされ、多くの領域で自動化できます。

プロセス テンプレート

VSTS ではプロセス テンプレートを使用して、新しいプロジェクトをセットアップするための指示、および成果物 (プロセス ガイダンスのドキュメント、ドキュメント テンプレート、デフォルトの作業項目など) を定義します。プロセス テンプレートはソフトウェア開発を先導するためのメソドロジを開発チームに与える、必要なものが完備された指示セットです。プロセス テンプレートには次の要素が含まれています。

- **プロセス ガイダンス。**これは各テンプレートに対して提供されるもので、特別なアクティビティの追随およびその理解に対する支援を必要とするチーム メンバに対して、コンテキスト依存の情報、ヘルプ、および指示を与えます。プロセス ガイダンスは、Visual Studio のヘルプ システムに統合されています。
- **ドキュメント テンプレート。**これらのテンプレートを使用して、チーム メンバは仕様書、リスク評価およびプロジェクト計画を一貫した方法で作成することができます。
- **作業項目およびワークフロー。**作業項目には、それぞれ固有のフィールドおよびルール セットがあります。それらは、作業項目のワークフロー、およびチーム メンバが作業を割当てて実施できる方法を定義します。
- **セキュリティ グループ。**これらのグループを使用して、レポート、作業の成果物 (ソース コードやドキュメントなど)、および作業項目を制御し、操作できる人を定義します。プロジェクト マネージャは、Windows の管理者にならなくてもセキュリティ グループを管理することができます。
- **チェックイン ポリシー。**これらのポリシーを使用して、ソース管理にチェックインするすべてのコードに対してルールおよびクオリティ ゲートを施行させます。たとえば、チェックインしたコードが、会社のコーディング基準に合致している、単体テストに合格している、などの特定の基準に適合するようにできます。カスタム チェックイン ポリシーの作成および構成方法の詳細については、「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してください。

- **レポート。**レポートは、プロジェクトの現在のパフォーマンスおよび状態を監視するために使用されます。VSTS には、コード品質レポート、スケジュール進捗レポート、テスト有効性レポートなどの、あらかじめ定義されたレポートが付随しています。独自のレポートを作成することも、既存のレポートをカスタマイズすることもできます。

MSF for Agile Software Development プロセス テンプレートと *MSF for CMMI Process Improvement* プロセス テンプレート

そのまま使用できるものとして、次の 2 つのプロセス テンプレートがあります。

- **MSF for Agile Software Development**。この軽量プロセス テンプレートは小規模なソフトウェア、アジャイル ソフトウェア、または形式にこだわらないソフトウェア プロジェクト用のもので、シナリオおよびコンテキスト ドリブンのアクションをベースとして、プロジェクトおよび人間を中心にしています。
- **MSF for CMMI® Process Improvement**。このプロセス テンプレートは、より成熟したソフトウェア プロジェクト向けに設計されています。これは、監査、検証、および正式なプロセスのサポートを提供することによって、MSF for Agile Software Development プロセス テンプレートを拡張しています。これは、プロセスおよびプロセスへの準拠をよりどころとし、組織を中心にしています。

提供されたテンプレートが特有のプロセス要件およびメソドロジにぴったり合わない場合は、システムに新しいプロセス テンプレートを追加し、組織のニーズに合うようにテンプレートをカスタマイズすることができます。既存のテンプレートのカスタマイズの詳細については、「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

セキュリティとアクセス許可

TFS でプロジェクトを作成すると、選択したプロセス テンプレートに関係なく、そのプロジェクトに対して既定の 4 つのグループが作成されます。既定では、どのグループに対しても一連のアクセス権が定義されており、グループ内のどのメンバに対してどのような処理が許可されるかが規定されます。

- プロジェクト管理者 (Project Administrator)
- 貢献者 (Contributor)
- 閲覧者 (Reader)
- ビルド サービス (Build Services)

チーム プロジェクトのセキュリティ グループを作成して、組織のセキュリティ要件により忠実に準拠することができます。セキュリティ グループを作成すると、チーム プロジェクトのそれぞれのユーザー グループに対して一連のアクセス許可を効率よく付与できます。グループに対して必要最小限のアクセス許可のみを付与し、この新しいチーム プロジェクト グループに所属する必要のあるユーザーまたはグループのみを追加するように注意します。

チーム プロジェクト グループの作成が完了したら、新しいグループを追加し、そのグループに適切な許可を付与して、グループにメンバを追加します。既定では、チーム プロジェクト グループには、作成時には何も権限が与えられません。

作業項目の管理

作業項目は、チーム間のコミュニケーションおよびコラボレーションのための作業単位として使用されます。選択したプロセス テンプレートは、作業項目の種類の初期設定を提供します。プロジェクト マネージャは、開発プロジェクトで揃えておく必要がある追加の作業項目を作成し、管理します。作業項目は、タスク、リスク、シナリオ、バグ、またはサービス品質 (QoS) 要求を定義します。追跡しやすいように、いくつかの作業項目どうしをリンクすることができます。たとえば、特定の作業項目のタスクを、関連するシナリオ、またはその作業項目が関連する QoS 要求に関連付けることができます。

プロセス テンプレートには、各作業項目の種類について定義されている一連のフィールドなど、作業項目の定義が用意されています。したがってプロセス テンプレートの選択は重要です。なぜならば、プロジェクトの期間中はプロセス テンプレートを変更できないからです。必要な場合は、プロセス テンプレートをカスタマイズして、ベース テンプレートで提供された作業項目の種類以外に、追加の作業項目の種類を含めます。

事前に定義されている多数の作業項目が、新しいチーム プロジェクトを作成するときに MSF for Agile Software Development および CMMI Process Improvement の両方のプロセス テンプレートで生成されます。これらの作業項目には、ソフトウェア開発プロセスを始めるために完了する必要のあるタスクが含まれているため、作業項目を使用してプロセスの使用をすばやく始めることができます。

MSF for Agile Software Development プロセス テンプレート

このプロセス テンプレートで提供される作業項目の種類には、次のものがあります。

- **シナリオ：** ユーザーとアプリケーション システムとのやりとりを表現するのに使用します。シナリオには、目的の達成に必要な詳細な手順を記載します。考えられるパスがたくさんあるため、シナリオを記述する場合には具体的にする必要があります。
- **タスク：** 実行しなければならない作業単位を表現するのに使用します。各ロールには、タスクに対する独自の要件があります。たとえば、開発者は作業を割り当てるために開発タスクを使用します。
- **サービス品質要求：** パフォーマンス、負荷、可用性、ストレス、アクセシビリティ、有用性など、システムの特性を文書化するのに使用します。
- **バグ：** システムの潜在的な問題を伝達するのに使用します。
- **リスク：** プロジェクトの固有のリスクを特定および管理するのに使用します。

MSF for CMMI® Process Improvement プロセス テンプレート

このプロセス テンプレートで提供される作業項目の種類には、次のものがあります。

- **要件：** 要件の収集フェーズで定義された要件を取得するのに使用します。
- **変更要求：** 要件の収集以降に発生した変更要求を取得するために使用します。
- **問題：** プロジェクトで追跡する問題を取得するのに使用します。
- **タスク：** 実行しなければならない作業単位を表現するのに使用します。各ロールには、タスクに対する独自の要件があります。たとえば、開発者は作業を割り当てるために開発タスクを使用します。
- **レビュー：** プロジェクト内のレビューの作業単位 (コード レビュー、設計レビューなど) を表現するのに使用します。

- **バグ：** システムの潜在的な問題を伝達するのに使用します。
- **リスク：** プロジェクトの固有のリスクを特定および管理するのに使用します。

Microsoft Project の統合

VSTS またはチーム エクスプローラ アプリケーションをインストールすると、Microsoft Project に対する拡張機能が提供されます。多数のリソースが関与する大規模なプロジェクトでは、Microsoft Office Project を使用して、TFS 内のスケジュール データを操作することができます。

たとえば、Microsoft Project を使用して、作業を管理、計画、割り当て、均衡化および追跡し、他のチーム メンバが使用する準備ができたときに、更新内容を作業項目データベースへ戻して公開することができます。

詳細については、http://msdn2.microsoft.com/en-us/library/ms244368(VS.80).aspx の「Working with Work Items in Microsoft Project」を参照してください。

Microsoft Excel の統合

VSTS またはチーム エクスプローラ アプリケーションをインストールすると、Microsoft Excel に対する拡張機能が提供されます。大量の作業項目が含まれているプロジェクトでは、Excel の統合機能を使用して Excel のスプレッドシートで作業項目を作成してから、作業項目データベースへアップロードして、他のチーム メンバが使用することができます。

詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181694(VS.80).aspx の「Microsoft Excel で作業項目リストを操作する」を参照してください。

進捗とレポーティング

TFS で提供されるレポートは、チーム プロジェクトのステータス、開発中のソフトウェアの品質、およびプロジェクトの完了に対する進捗を迅速に評価できるよう支援します。これらのレポートは、TFS

のデータ ウェアハウスで保持されているデータから生成され、作業項目、ソース管理、テスト結果およびビルドから導出されるメトリックを要約します。

たとえばレポートを使用して、チームの実際のアクティビティに基づいて、チームがどのくらいのスピードで作業しているかを週ごとに調べることができます。TFS のレポーティング システムの目的は、VSTS コンポーネント全体からの統合データを提供して、それによりプロジェクト マネージャおよびチーム メンバがソフトウェア開発プロジェクトの状態を理解し、プロジェクトの成功を保証するための適切な措置を講じることを可能にすることです。

チーム プロジェクトを作成するために使用するプロセス テンプレートでは、既定でどのレポートが使用できるかを定義していますが、独自のカスタム レポートを追加することもできます。プロセス テンプレートで作成される各レポートの内容および使用方法は、そのテンプレートのプロセス ガイダンスで説明されています。Team Foundation Server は Microsoft SQL Server™ 2005 上に構築されており、SQL Server を使用して、作業項目、品質属性、テスト、テスト結果およびビルド結果に関するすべてのデータを格納しています。TFS は SQL Server Analysis Services を使用して、データを集約および分析し、レポートを作成します。プロセス テンプレートで作成されたレポート、または Microsoft Office Excel あるいは Visual Studio 2005 レポート デザイナ を使用して個々のチーム メンバによって作成されたレポートは、SQL Server 2005 Reporting Services およびチームの SharePoint ポータル サイトを介して利用できます。

レポートのカスタマイズの詳細については、「Visual Studio Team Foundation Server のカスタム レポートを作成する方法」を参照してください。

まとめ

Team Foundation Server には、Visual Studio で開発プロジェクトの管理機能を改善するために、作業項目の一括管理、プロセス管理、セキュリティおよびアクセス許可の管理、プロジェクト メトリック、レポーティングなどのプロジェクト管理機能が用意されています。

ソフトウェア開発ライフサイクルは、ソフトウェア プロジェクトの開発作業をサポートするツール選

択の不可欠な部分となっています。TFS には、MSF Agile プロセス テンプレートおよび MSF CMMI プロセス テンプレートが用意されていますが、これらは 2 つの大きく異なる開発方法をサポートします。提供されたプロセス テンプレートを修正することも、自身のチームの開発プロセス ニーズに合わせて最初からテンプレートを作成することもできます。

参考資料

- VSTS を使用したソフトウェア プロジェクトの開発の詳細については、http://msdn2.microsoft.com/en-us/library/aa302181.aspx の「Visual Studio 2005 Team System: Software Project Management」を参照してください。
- プロジェクト管理のタスクに対する Microsoft Office Excel の使用の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181694(VS.80).aspx の「Microsoft Excel で作業項目リストを操作する」を参照してください。
- プロジェクト管理のタスクに対する Microsoft Office Project の使用の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms244368(VS.80).aspx の「Microsoft Project で作業項目を操作する」を参照してください。

第 12 章 作業項目の説明

目的

- 作業項目の目的および構造について学習します。
- 作業項目のワークフローについて説明します。
- 自身のチーム特有のニーズに合わせて作業項目をカスタマイズします。

概要

この章では、作業項目について紹介し、作業項目をどのように使用してソフトウェア開発プロジェクト

の管理をサポートするかについて説明します。各作業項目は、開発チームで実行される作業単位を表しています。一連の作業項目の種類は、新しいチーム プロジェクトを作成するときに選択した、プロセス テンプレート内で定義されます。

プロジェクトが始まると、任意の利用可能な作業項目の種類を作成して作業を追跡することができます。デフォルトの作業項目の種類および動作はプロセス テンプレートで定義されていますが、チームの作業方法により良く適合するように、作業項目の種類の任意の側面を修正することができます。

この章の参照の仕方

この章を最大限に利用するには、次のようにします。

- 「作業項目の構造」セクションを読んで、あらかじめ定義されている作業項目の種類について、および作業項目のワークフローがどのように定義されているかについて学習します。
- 「作業項目のカスタマイズ」セクションを読んで、作業項目の種類をカスタマイズする方法と理由について学習します。

シナリオとソリューション

作業項目は、プロジェクト マネージャおよびチーム リーダーが、プロジェクトの残存作業と既に完了した作業を追跡するための主要な方法です。チーム メンバは作業項目を使用して、個人的なワーク キューを追跡し、バグやタスクなどの形式で作業を相互に割り当てます。

次に、チーム プロジェクトにおける作業項目の一般的な用途について示します。

- アプリケーションに対するユーザー要件またはサービス品質 (QoS) 要求を作成します。
- 要件に対する開発およびテストを追跡します。
- アプリケーションのコンポーネントおよび機能を実装するために完了すべき作業を表す開発タスクを作成します。
- アプリケーションのコンポーネントおよび機能の実装における障害を表すバグを作成します。
- バグおよびタスクの優先順位を決定し、チーム全体でそれらを適切に優先度付けて、バランスをとります。

- 開発タスクを追跡して、コードの完了ステータスに対する進捗を判断します。
- 他の品質メトリックと共にバグを追跡し、アプリケーションの品質、および出荷の準備状況を判断します。

作業項目の使用方法は、プロジェクトで定義されている作業項目の種類によって異なります。作業項目の定義は、最初にチーム プロジェクトを作成したときに、選択したプロセス テンプレートに格納されます。Microsoft® Solution Framework (MSF) for Agile Software Development (MSF Agile) または MSF for CMMI® Process Improvement (MSF CMMI) というデフォルトの 2 つのテンプレートのいずれかを選択するか、あるいは、自身のチームの特別なニーズおよびプロセスに合わせて作業項目をカスタマイズすることができます。

作業項目の構造

各作業項目の種類は次のように定義することができます。

- 各作業項目には、目的および意図される用途があります。たとえば、品質の欠陥の追跡にはバグが使用され、スケジュールされた作業の追跡にはタスクが使用され、セキュリティやパフォーマンスの要件など、機能以外の重要な側面を捉えるには QoS 要求が使用されます。
- 各作業項目には、状態と遷移で定義されるワークフローがあります。たとえば、「作業中」から「解決済み」、「終了」状態へステップが遷移します。
- 各作業項目には、一連のフィールドがあり、設定、クエリおよびレポートすることができます。たとえば、優先度、状態、およびイテレーションがあります。

作業項目の種類

MSF Agile プロセス テンプレートおよび MSF CMMI プロセス テンプレートはそれぞれ一連の作業項目を定義しています。これらの作業項目は、プロセス ガイダンスで定義されているロールおよびアクティビティへマップされます。

MSF Agile の作業項目の種類

MSF Agile には次の作業項目の種類が含まれています。

- **バグ。**アプリケーションにおける問題、または潜在的な問題を表します。
- **リスク。**プロジェクトにマイナスの影響を与える可能性のあるイベントまたは状況を表します。
- **シナリオ。**システムを介したユーザーとのやりとりにおける単一のパスを表します。
- **タスク。**チームのメンバが実行する必要がある作業を表します。
- **サービス品質要求。**システムがどのように機能すべきかについて課せられる要件を表します。

MSF CMMI の作業項目の種類

MSF CMMI には、次の作業項目の種類が含まれています。

- **バグ。**アプリケーションにおける問題、または潜在的な問題を表します。
- **変更要求。**アプリケーションに対して提示された変更を表します。
- **問題。**作業を妨害する可能性のある状況、または妨害している状況を表します。
- **要件。**お客様の問題を解決するためにアプリケーションで行うべき内容の説明を表します。
- **レビュー。**コード、設計、または配置のレビュー結果を表します。
- **リスク。**プロジェクトにマイナスの影響を与える可能性のあるイベントまたは状況を表します。
- **タスク。**チームのメンバが実行する必要がある作業を表します。

作業項目のワークフロー

各作業項目には、あらかじめ定義されている 1 つのワークフローがあり、作業項目がとり得る状態、および状態間の遷移を表します。それぞれの状態は、TFS のロールに自然に関連付けられています。たとえば、テスト担当者が MSF Agile で新しいバグをオープンすると、状態は**アクティブ**になります。開発者がバグを修正すると、状態は**解決済み**に変わります。テスト担当者が修正を確認すると、バグの状態は**終了**に変わります。

ワークフローの例

次の例は、一般的な 2 つの作業項目の種類に対するワークフローを示しています。

MSF CMMI のタスク

MSF CMMI のタスクでは、次の状態があります。

- **提案。**開発者、テスト担当者、または設計者によって提案されている、など。
- **アクティブ。**リーダーまたはマネージャによって承認されている、など。
- **解決済み。**開発者によって解決されている、など。
- **終了。**テスト担当者によってテストされ、終了している、など。

図 12.1 は、各状態、および状態間で考えられる遷移を示しています。

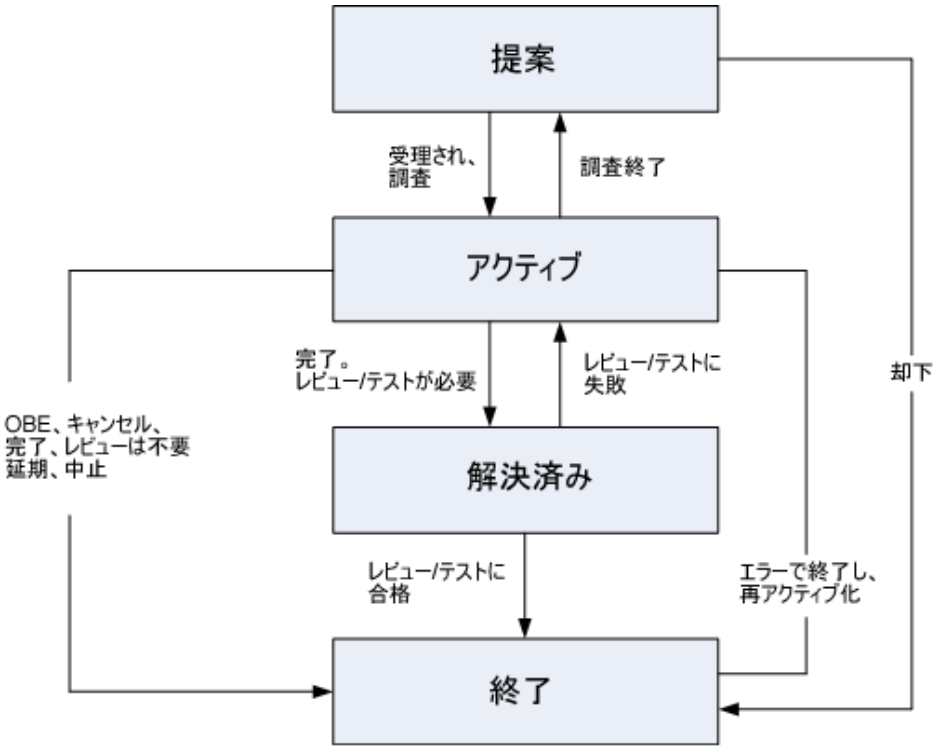


図 *12.1* *MSF CMMI* の遷移状態

MSF Agile のバグ

MSF Agile のバグでは、次の状態があります。

- **アクティブ。**テスト担当者によってオープンされている、など。
- **解決済み。**開発者によって解決されている、など。
- **終了。**テスト担当者によってテストされ、終了している、など。

図 12.2 は、各状態、および状態間で考えられる遷移を示しています。

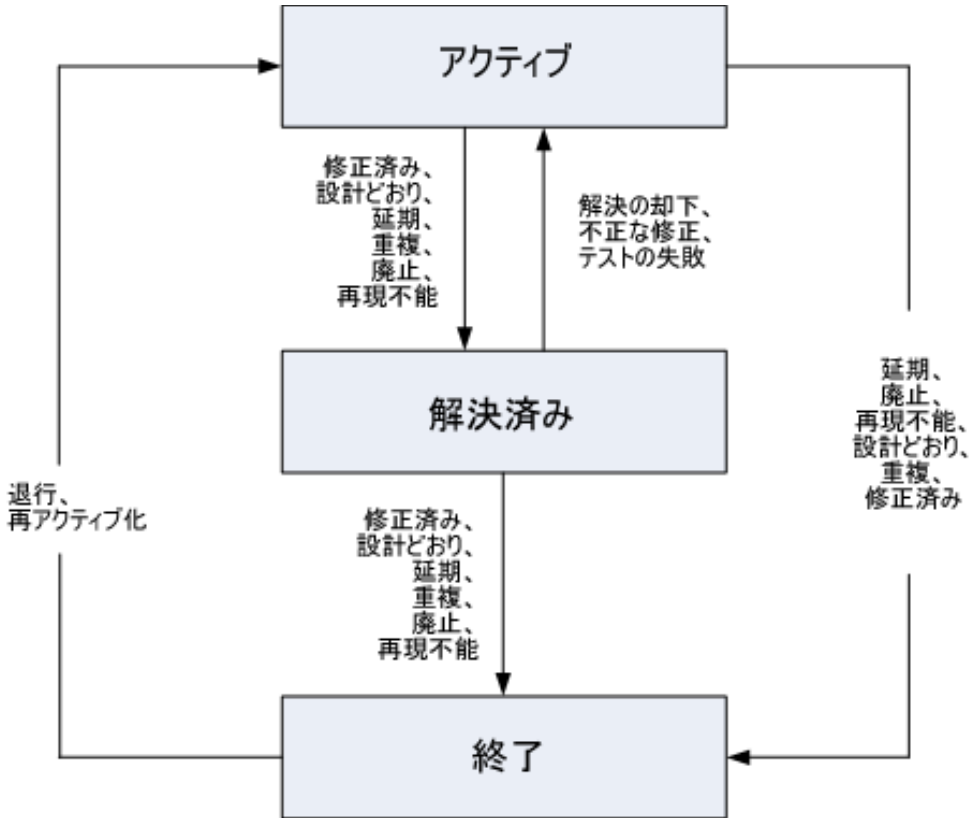


図 *12.2* *MSF Agile* の遷移状態

作業項目のカスタマイズ

いくつかのシナリオで、MSF Agile または MSF CMMI で定義されている作業項目の種類を修正したい場合があります。

- 作業項目で、開発プロセスにとって重要なフィールドが不足している場合。
- 作業項目のワークフローがチームの作業に合っていない場合。
- 新しい作業項目の種類が必要な場合。

これらのシナリオをサポートするには、TFS で次のようにすることができます。

- 作業項目の種類を追加/削除します。
- 既存の作業項目内のフィールドを修正します。
- 既存の作業項目内の状態および遷移を修正します。

作業項目のカスタマイズの詳細については、「Visual Studio Team Foundation Server でプロセス テ

ンプレートをカスタマイズする方法」を参照してください。

まとめ

作業項目は、プロジェクトで実行する作業を追跡するためにプロジェクト マネージャおよびチーム リーダーが使用します。作業項目を使用して、完了しなければならない作業を表す開発タスクの作成、実装における障害を表すバグの作成、ユーザー要件またはサービス品質 (QoS) 要求の作成を行います。また、作業項目を使用して、要件に対する開発およびテストを追跡し、アプリケーションの品質および出荷の準備状況について判断することができます。

MSF Agile および MSF CMMI のプロセス テンプレートには、一連のデフォルトの作業項目の種類が用意されています。これらのものはカスタマイズすることも、プロセス要件に合わせて新しい作業項目の種類を作成することもできます。

参考資料

- 作業項目の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181314(VS.80).aspx の「Team Foundation の作業項目の管理」を参照してください。
- MSF CMMI プロセス テンプレートをダウンロードし、使用できる作業項目の種類を確認するには、http://www.microsoft.com/downloads/details.aspx?FamilyID=12A8D806-BB98-4EB4-BF6B-FB5B266171EB&displaylang=en へアクセスしてください。
- MSF Agile プロセス テンプレートをダウンロードし、使用できる作業項目の種類を確認するには、http://www.microsoft.com/downloads/details.aspx?FamilyID=EA75784E-3A3F-48FB-824E-828BF593C34D&displaylang=en へアクセスしてください。

第 6 部

プロセス テンプレート

第 6 部の内容

- **プロセス テンプレートの説明**
- **MSF for Agile Software Development プロジェクト**

第 13 章 プロセス テンプレートの説明

目的

- プロセス テンプレートの目的、コンテンツおよび構造を理解します。
- Microsoft® Solution Framework (MSF) for Agile Software Development (MSF Agile) および MSF for CMMI® Process Improvements (MSF CMMI) のプロセス テンプレート間の主な違いについて学習します。
- チーム特有のニーズに合わせてプロセス テンプレートをカスタマイズします。

概要

この章では、Microsoft Visual Studio® 2005 Team Foundation Server (TFS) におけるプロセス テンプレートの役割について説明します。ここでは、提供される 2 つのプロセス テンプレート、つまり MSF Agile と MSF CMMI の主な機能および主な違いについて確認します。

ソフトウェア開発のプロセスは複雑で、相互に依存する多数のレベルのアクティビティが含まれています。これらのプロセスは一般に開発チームがドキュメントの形式で利用できますが、通常はツールにより強制されません。このようにツールによるサポートがないために、開発チームがプロセスについて習得し、それに忠実に従うことは困難です。プロジェクト マネージャは、プロジェクト管理、要件管理、バグの追跡またはレビュー管理のためにさまざまなツールを使用することはできますが、多くの場合には、あまり十分に統合されていません。このように統合が実現されていないことにより、複数のプロジェクト間で一貫した方法論を実行すること、および整合性のとれたレポートを生成して、プロジェクトの進捗と状態についてチームの共通した理解を得ることは、さらに難しくなっています。このように整合性が欠如すると、プロセスの分析は信頼性の低いものになり、長期にわたるプロセス改善箇所の特定および実施がさらに難しくなります。

Visual Studio Team System (VSTS) および TFS は、ソフトウェア開発プロジェクトに関わるプロセス アクティビティの大半をサポートする統合環境を提供します。TFS は、プロセス テンプレートを使用することにより、そのライフ サイクルの方法論を実装します。プロセス テンプレートとは、開発方法論のプロセスおよび成果物の仕様を規定した、一連の Extensible Markup Language (XML) ファイルのことです。この章では、プロセス テンプレートのアーキテクチャおよびそのコンポーネントについて説明し、提供されるプロセス テンプレートをどのように利用し、カスタマイズするかについて理解を深めます。

既存のプロセス テンプレートがチームの開発プロセスに合わない場合は、新しいプロセス テンプレートを作成したり、既存のテンプレートを修正したり、Microsoft Partners で提供されるプロセス テンプレートから選択したりできます。Microsoft Partners で提供されるプロセス テンプレートをレビューするには、http://msdn2.microsoft.com/en-us/teamsystem/aa718801 を参照してください。

この章の参照の仕方

この章を使用して、プロセス テンプレートのアーキテクチャ、構造、およびカスタマイズについて理解します。この章を最大限に利用するには、次のようにします。

- デフォルトのプロセス テンプレートについて、およびチームのニーズに最も適合するテンプレー

トの選択方法について理解するには、**「MSF Agile プロセス テンプレートおよび MSF CMMI プロセス テンプレート」のセクションを読んでください。**

- チームのニーズにさらに近づけるために既存のプロセス テンプレートをカスタマイズする方法について理解するには、**「プロセスのカスタマイズ ガイダンス」のセクションを読んでください。**

MSF Agile プロセス テンプレートおよび MSF CMMI プロセス テンプレート

Team Foundation Server には、MSF Agile と MSF CMMI という 2 つのプロセス テンプレートがあります。これらの 2 つのプロセス テンプレートは、2 つの異なるスタイルのソフトウェア開発を対象としたものです。ソフトウェアを作成するのにアジャイル方法論を採用している場合は、MSF Agile を使用します。MSF Agile はテスト ドリブンな開発その他のアジャイル プラクティスを促進します。Software Engineering Institute (SEI) Capability Maturity Model® Integration の方法論を採用している場合は、MSF CMMI を使用します。これは、既存の開発プロセスを改善することを目的とした正式なプロセスです。

これらのテンプレートは、それぞれ提供する内容が異なります。たとえば、テンプレートによって、デフォルトのレポート タイプ、および作業項目の種類が異なります。これらのテンプレートは、プロジェクトのニーズに合うように簡単に編集することができます。

プロセス ガイダンスのカスタマイズ

作成中のプロジェクトは、VSTS で提供されるプロセス テンプレートに合わないかもしれません。違う作業項目の種類が必要になったり、まったく別のプロセス方法論を使用している可能性もあります。たとえば、SCRUM を使用している場合に、対象のプロセス テンプレートの中で「スプリント」がまったく出てきません。このような場合には、チームが使用している方法論に合うように、既存のプロセス テンプレートを修正または置き換えます。

プロセス テンプレートのアーキテクチャ

プロセス テンプレートのアーキテクチャでは、次の 3 つが重要です。

- **プロセス テンプレート プラグイン**
- **XML プロセス定義ファイル**
- **新しいチーム プロジェクト ウィザード**

プロセス テンプレート プラグイン

プロセス テンプレート プラグインは、新しいチーム プロジェクトが作成されたときに実行されるコンポーネントです。プラグインは必要なファイルをセットアップし、特定領域のテンプレートに関するデータを構成します。TFS ですぐに使用できるプラグインには、次のものがあります。

- **分類** – プロジェクトの最初のイテレーションおよび区分を定義します。
- **グループおよびアクセス許可** – チーム プロジェクトの最初のセキュリティ グループおよびそのアクセス許可を定義します。
- **Windows SharePoint Services** – Microsoft Windows SharePoint® のサイト テンプレートに基づいて、チームのプロジェクト ポータルを定義します。テンプレート ファイルおよびプロセス ガイダンスも定義します。
- **作業項目の追跡** – プロジェクトの最初の作業項目の種類、クエリ、および作業項目のインスタンスを定義します。
- **レポート** – チーム プロジェクトの最初のレポートを定義し、レポート サイトをセットアップします。
- **バージョン管理** – チーム プロジェクトの最初のバージョン管理のセキュリティ許可、およびチェックイン メモを定義します。

それぞれのプラグインの定義ファイルを修正して、プロセス テンプレートをカスタマイズすることができます。分類のプラグインを除いては、プラグインの定義ファイルを削除してプロセス テンプレートをカスタマイズすることもできます。

XML プロセス定義ファイル

XML プロセス定義ファイルは ＸＭＬ のファイル セットで、プロセスに対して新しいチーム プロジェクトを正しく構成するために実行しなければならない一連のタスクを定義します。**新しいチーム プロジェクト** ウィザードを使用してチーム プロジェクトを作成する場合は、このファイルによって、必要なプラグインが実行されます。各プラグインは、対応する XML プロセス定義ファイルを読み込んで、実行しなければならないタスク リストを取得します。XML プロセス定義ファイルを使用して、プラグインが実装しなければならない構成および設定を指定します。次に、XML ファイルの詳細を示します。

- **作業項目の追跡 XML** – このプロセス定義ファイルは Workitems.xml という名前で、プロセス テンプレートのフォルダの階層の Work Item Tracking フォルダにあります。指定するタスクには、主に 3 つのタイプ、つまり作業項目の種類、作業項目のクエリ、および作業項目のインスタンスがあります。
  - **作業項目の種類** – タスク、バグ、要件など、チーム プロジェクトで追跡される作業項目のルール、フィールド、状態、および遷移を定義します。
  - **作業項目クエリ** – タスクや有効なバグなど、作業項目の特定のグルーピングを見つけるのに使用します。作業項目クエリは、プロセス テンプレートのフォルダ階層内の Work Item Tracking フォルダ下の Queries フォルダの中の作業項目クエリ (WIQ) ファイルで指定されます。
  - **作業項目インスタンス** – チーム プロジェクトを作成したときにデフォルトで作成される、作業項目インスタンスの最初のセット。
- **分類 XML** – このプロセス定義ファイルは Classification.xml という名前で、プロセス テンプレートのフォルダ階層内の Classification フォルダにあります。これは、イテレーションと区分という 2 つの部分で構成されています。
  - **イテレーション** – チームが、主なアクティビティ (計画、開発、テストなど) の特定の部分を何回繰り返すかを定義する場合に使用します。イテレーションは作業項目をグループ化するために使用されるため、作業項目クエリおよびレポートに影響を与えます。
  - **区分** – チーム プロジェクト内の作業を編成する場合に使用します。たとえば、チームでは、UI 区分、Application 区分、Database 区分など、製品または機能に基づいて作業を編成することができます。区分を使用して、クエリおよびレポートに対して作業項目をグループ化す

ることができます。

- **Windows SharePoint Services XML** – このプロセス定義ファイルは WssTasks.xml という名前で、プロセス テンプレートのフォルダ階層内の Windows SharePoint Services フォルダにあります。ここでは、指定する 3 つの主なタスク、つまりサイト テンプレート (どのサイト テンプレートを使用するか)、ドキュメント ライブラリ (どのドキュメント ライブラリを作成するか)、およびフォルダとファイル (どのフォルダとファイルをドキュメント ライブラリへコピーするか) があります。
  - **サイト テンプレート** – プロジェクト ポータルがベースとするサイト テンプレートを指定する必要があります。また、このサイト テンプレートは、TFS SharePoint ポータルで使用できるようにする必要があります。サイト テンプレートは、プロセス テンプレートには含まれません。
  - **ドキュメント ライブラリ** – プロジェクト ポータルが作成されると、追加のドキュメント ライブラリを作成することを指定できます。
  - **フォルダとファイル** – プロジェクト ポータルが作成されたら、作成する追加のフォルダを指定することができます。また、テンプレート ファイルなど、コピーするファイルも指定できます。
- **バージョン管理 XML** – このプロセス定義ファイルは VersionControl.xml という名前で、プロセス テンプレートのフォルダ階層内の Version Control フォルダにあります。これは、チーム プロジェクトの最初のバージョン管理のセキュリティ アクセス許可、チェックイン メモ、および排他チェックアウトが必要かどうかを定義します。
  - **チェックイン メモ** – チェックイン メモを含めるかどうかを指定します。チェックイン メモは、コードがチェックインされたときに開発者によって与えられ、コードの変更がチームプロセスに関係するか否か、またはコードの変更がどのようにチーム プロセスに関係するかを記述します。たとえば、チェックイン メモは、変更がセキュリティ レビューの一部であったかどうかを示すことが可能です。また、このメモにはセキュリティ レビューに関する変更の詳細を含めることができます。
  - **排他チェックアウト** – 複数のユーザーが同時に 1 つのファイルをチェックアウトできるかどうかを制御するために使用します。
  - **アクセス許可 –** アクティビティ セキュリティ グループおよび個人が、バージョン管理の項目において実行できる内容を定義します。

- **レポート XML** – このプロセス定義ファイルは ReportsTasks.xml という名前で、プロセス テンプレートのフォルダ階層内の Reports フォルダにあります。これは、チーム プロジェクトの最初のレポートを定義します。
  - **レポーティング サイト** – レポーティング サイトは、レポート サイトに対してリンクがあり、そのリンクは、プロジェクト ポータルのホーム ページ上で、Reports とラベル付けされています。
  - **フォルダ** – レポーティング サイトにフォルダを作成できます。作成するフォルダは、プロジェクト サイトで、チーム エクスプローラの Reports フォルダの下に表示されます。
  - **レポート** – .rdl ファイルを使用してレポートを追加する場合に使用します。
- **グループとアクセス許可 XML** – このプロセス定義ファイルは GroupsandPermissions.xml という名前で、プロセス テンプレートのフォルダ階層内の Groups and Permissions フォルダにあります。これは、チーム プロジェクトの最初のセキュリティ グループを定義します。
  - **グループ** – TFS の新しいセキュリティ グループを指定するのに使用します。
  - **アクセス許可** – 指定する各グループのアクセス許可を定義するのに使用します。

新しいチーム プロジェクト ウィザード

新しいチーム プロジェクト ウィザードを使用して、新しいチーム プロジェクトを作成します。このウィザードは、プラグインおよび XML プロセス定義ファイルを使用してプロジェクトを作成します。

カスタマイズのアプローチ

プロセス テンプレートをカスタマイズするには、次の手順を実行します。

1. TFS で提供されているプロセス テンプレートを調べて、組織のプロセスに最も近いものを選択します。
2. 選択したプロセス テンプレートをダウンロードします。
3. プロセス テンプレートのさまざまなコンポーネントをカスタマイズします。
4. カスタマイズしたテンプレートを TFS へアップロードします。
5. 変更が、プロセスの要件に合っていることを検証します。

この基本的なアプローチは、プロセス テンプレートをカスタマイズするための次のソリューションの一部として使用されます。

- **XML ファイルを手動でカスタマイズします。**手動によるカスタマイズではエラーが発生しやすくなりますが、プロセス テンプレートのカスタマイズをきめ細かくコントロールできます。詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243782(VS.80).aspx の「プロセス テンプレートのカスタマイズ」を参照してください。
- **Power ツールで使用できる Process Template Editor Tool。**TFS のエクスペリエンスを改善する一連の拡張機能、ツール、コマンドライン ユーティリティである Visual Studio 2005 Team Foundation Server Power Tool の最新バージョンには、プロセス テンプレートを参照およびカスタマイズするためのグラフィカル ツールが用意されています。TFS に接続すると、このツールを使用して、作業項目の種類の定義、およびアクティブなプロジェクトのグローバル リストをカスタマイズできます。詳細については、「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

一般的なカスタマイズ

次に、独自のカスタム プロセスを作成するために通常カスタマイズするコンポーネント セットについて説明します。

- **グループおよびアクセス許可。**既定の状態のプロセス テンプレートには、いくつかのグループ、およびそのグループに割り当てられているさまざまなアクセス許可があります。これらのテンプレートに関連付けられているデフォルトのグループおよびアクセス許可が自身のプロセス要件に対して不十分、または不適切な場合は、これらのグループを更新したり、新しいグループを作成したりできます。また、グループに別のユーザーを追加する、グループからユーザーを削除する、グループに対してアクセス許可を付与および破棄する、といったことができます。
- **ソース管理のチェックイン メモおよびポリシー。**すぐに使用できるプロセス テンプレートには、ソース管理のチェックイン メモおよびポリシーのセットがあります。デフォルトのチェックイン メモが自身のプロセス要件に対して不十分または不適切な場合は、チェックイン メモ フィールド

を追加または削除したり、いくつかのフィールドを必須にして、その他を不要にしたりできます。デフォルトのチェックイン ポリシーが不十分または不適切な場合は、個々のチェックイン ポリシーを追加、更新または削除できます。

- **区分とイテレーション。**既定の状態のプロセス テンプレートには、区分またはイテレーションに対する分類の構造がありません。特定のプロセス要件に従って、区分およびイテレーションをカスタマイズすることができます。このような場合は、プロジェクトのコンポーネントまたは機能に基づいて区分を作成する方法を推奨します。イテレーションには、（計画、開発、およびテストなどの） 主要なアクティビティの特定の組を繰り返す、時間に基づいたサイクルを使用できます。
- **チーム ポータル。**既定の状態のプロセス テンプレートには、デフォルトのチーム ポータルがあります。このポータルは、チームのメンバと組織内の他のユーザーの間で連携をとるための中心のハブになります。チーム ポータルを修正して、プロセス要件に合わせてポータルの外観、動作、およびコンテンツを変更することができます。
- **プロセス ガイダンス。**既定の状態のプロセス テンプレートには、関連するプロセス ガイダンスが用意されており、チーム プロジェクトで使用されるロール、フォーム、レポートおよびワークフローについて説明しています。要件に合うようにプロセス テンプレートをカスタマイズする場合には、プロセス ガイダンスを編集して、さまざまなコンポーネントに対する変更を反映させる必要があります。
- **レポート。**既定の状態のプロセス テンプレートには、デフォルトのレポートがいくつか用意されています。デフォルトのレポートが不十分または不適切な場合は、自身の要件に基づいて独自のカスタム レポートを作成できます。
- **作業項目の種類とクエリ。**既定の状態のプロセス テンプレートには、作業項目の種類、作業項目のデフォルトのインスタンスおよびクエリがあります。作業項目のデフォルトの種類、作業項目のインスタンスおよびクエリがプロセス要件に対して不十分、または不適切な場合は、次のようにすることができます。ワークフロー、または追跡したい作業項目の種類に合うように、作業項目の種類をカスタマイズします。たとえば、次のことができます。

  - 新しい作業項目の種類を追加する。
  - 既存の作業項目の種類を削除する。
  - デフォルトの作業項目の種類のインスタンスを追加する。
  - デフォルトの作業項目の種類のインスタンスを削除する。

- 独自のパブリックまたはプライベートなクエリを作成する。

既存の作業項目の種類を修正することもできます。たとえば、以下です。

- フィールドを追加する。
- フィールドの名前を変更する。
- フィールドに対して使用できる値のリストを限定する。
- 状態およびサポートされる状態の遷移を変更する。
- フィールドを必須または読み取り専用にする。
- あるフィールドを他のフィールドに依存させる。
- フィールドの値を自動的に取り込む。
- フォーム上の情報の表示を再配置する。
- Microsoft Office Project のカラムを変更して、特定のフィールドがマップされるようにする。

カスタマイズのプロセスが機能するしくみ

プロセス テンプレートのカスタマイズには次の手順が含まれます。

1. ユーザーが**新しいチーム プロジェクト** ウィザードを起動します。
2. ウィザードで次の情報が要求されます。
   - チーム プロジェクトの名前
   - チーム プロジェクトの作成時に使用するプロセス テンプレート

   ウィザードに表示される画面は、使用されるプラグインによって決まります。たとえば、プロセス テンプレートに Windows SharePoint Services プラグインが含まれていない場合は、プロジェクト ポータルに関する情報を要求するための画面はありません。
3. ユーザーが、ウィザードで要求される情報を入力して **[完了]** をクリックすると、ウィザードは、チーム プロジェクトの作成を実行するためのプラグインを呼び出します。プラグインが呼び出される順序は、XML プロセス定義ファイルによって定義されます。
4. ウィザードが、プロセス テンプレートに含まれている指示を読み込み、指定されている項目を作成

および構成します。

プロセス テンプレートによって指示が提供されるため、ユーザーは、作成する作業項目のさまざまな種類に関する詳細を指定する必要はありません。

注意:新しいチーム プロジェクトを作成するときにウィザードで問題が発生すると、問題の内容を説明し、正しい処置を指示するエラー メッセージが表示されます。

まとめ

ソフトウェアを作成するのにアジャイル方法論を採用している場合は、MSF Agile プロセス テンプレートを使用します。Software Engineering Institute (SEI) Capability Maturity Model® Integration の方法論を採用している場合は、MSF CMMI を使用します。

プロセス テンプレートのアーキテクチャの主なコンポーネントは、プロセス テンプレートのプラグイン、XML プロセス定義ファイル、および**新しいチーム プロジェクト** ウィザードです。

プロセス要件に対してデフォルトのプロセス テンプレートが十分でない場合は、手動で XML プロセス定義ファイルをカスタマイズしてテンプレートをカスタマイズしたり、Process Editor Tool を使用してプロセス テンプレートをカスタマイズしたりできます。

カスタマイズの最も一般的な領域は、グループとアクセス許可、ソース管理のチェックイン メモとポリシー、区分とイテレーション、レポート、および作業項目の種類の定義です。

参考資料

- プロセス テンプレートの選択についての詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400752(vs.80).aspx の「プロセス テンプレートの選択」を参照してください。
- MSF CMMI プロセス テンプレートをダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyID=12A8D806-BB98-4EB4-BF

6B-FB5B266171EB&displaylang=en へアクセスしてください。

- MSF Agile プロセス テンプレートをダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyID=EA75784E-3A3F-48FB-824E-828BF593C34D&displaylang=en へアクセスしてください。
- プロセス テンプレートのカスタマイズの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194945(VS.80).aspx の「プロセス テンプレートのカスタマイズの概要」を参照してください。
- Process Template Editor を含めて Team Foundation Server Power Tools をダウンロードするには、http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx へアクセスしてください。
- Microsoft Partners で提供されるテンプレートをレビューするには、http://msdn2.microsoft.com/en-us/teamsystem/aa718801 を参照してください。

第 14 章 アジャイル ソフトウェア開発プロジェクトに対する MSF

目的

- Microsoft® Solution Framework (MSF) for Agile Software Development (MSF Agile) のプロセス テンプレートを使用する場合について学習します。
- 一般的にチームがどのように MSF Agile プロセス テンプレートを使用するかを確認します。
- チーム特有のニーズに合わせて MSF Agile プロセス テンプレートをカスタマイズします。

概要

MSF Agile プロセス テンプレートで定義されるプロセスには、Agile ソフトウェアの動向、および MSF の原理と実践が盛り込まれています。プロセスは、アジャイル ソフトウェア エンジニアリング戦略をサポートしていますが、これはアプリケーションを作成するために複数のイテレーションとシナリオベースのアプローチを使用します。このテンプレートには、構成管理、プロジェクト管理、作業項

目の追跡、連携のためのプロジェクト ポータルなど、チームの開発をサポートするのに必要な自動化およびガイダンスが含まれています。

この章では、一般的な MSF Agile ソフトウェア開発プロジェクトのワークフローについて説明し、MSF Agile プロセスを使用したチームの例を示します。また、テンプレートのデフォルトの設定について、および提供されるテンプレートでどのオプションをカスタマイズしなければならないかを説明します。

この章の参照の仕方

MSF Agile プロセス テンプレートおよびプロセス ガイダンスがどのように機能するか、またこのテンプレートがさまざまなチームによって、どのようにうまく使用されているかについて、理解を深めたい場合はこの章を読んでください。この章を最大限に利用するには、次のようにします。

- デフォルトのレポート、作業項目、アクセス許可など、MSF Agile プロセス テンプレートの詳細を理解するには、**「MSF for Agile Software Development のデフォルト」セクションを読んでください。**
- 実際のチームが MSF Agile をどのように使用してアプリケーションを開発およびリリースしたかを参照するには、**「MSF for Agile Software Development の実践例」セクションを読んでください。**
- プロセス テンプレートの概要について詳しく知りたい場合は、**「第 13 章 プロセス テンプレートの説明」を読んでください。**

MSF for Agile Software Development のワークフロー

MSF Agile プロセス テンプレートは、ソフトウェア開発のライフサイクルに関するさまざまな役割 (ビジネス アナリスト、設計者、プロジェクト マネージャ、開発者、テスト担当者など) によって、イテレーション中に実行される一連のタスクを定義します。図 14.1 は、定義されたそれぞれのタスクに関連付けられている主なアクティビティを示しています。

タスク	主なアクティビティ
プロジェクト バックログを作成する	プロジェクト ビジョンを理解する
ソリューション アーキテクチャを作成する	アーキテクチャのプロトタイプを作成する インターフェイスを決定する インフラストラクチャ アーキテクチャを作成する
お客様の使用シナリオを作成する	シナリオに分解する シナリオを TFS に取り込む シナリオの一覧で優先順を付ける
サービス品質 (QoS) の要件を作成する	セキュリティの目標を決める パフォーマンスの目標を決める 要件を TFS に取り込む 要件の優先順を決める
イテレーションの計画	シナリオおよび要件を開発およびテストのタスクに分割する 開発およびテストのタスクに対する見積を作成する 開発およびテストのタスクをスケジュールし、割り当てる
イテレーションの開発	開発タスクのコードを記述する 開発タスクの単体テストを作成/更新する 単体テストを実行し、コード分析を実行する
イテレーションのテスト	テスト タスクの検証テストを記述する テスト ケースを実行し、予備テストを実施する 検出された問題のバグを公開する
イテレーションの進捗をレビューする	イテレーションに対するタスクの進捗を評価する イテレーションで特定されたバグの優先順を付ける テストのメトリックしきい値を評価する

図 *14.1* *MSF Agile のタスクと主なアクティビティ*

MSF for Agile Software Development のデフォルト

MSF Agile プロセス テンプレートを使用して新しいチーム プロジェクトを作成すると、Microsoft

Visual Studio® のメイン ウィンドウに、プロセス ガイダンスの概要を説明するコンセプト ページが表示されます。これは、MSF Agile プロセスへの最初のビューポイントです。プロジェクト ポータルのホーム ページからこの情報にアクセスすることもできます。

ツールの構成はプロセスの説明だけでなく、（シナリオ、サービス品質要求、タスク、バグ、およびリスクのような） 作業項目、プロジェクト レポート、ロール （グループとアクセス許可）、およびプロジェクト ポータルを含みます。MSF Agile のテンプレートで提供される主な項目は次のとおりです。

- 作業項目
- グループおよびアクセス許可
- ソース管理
- 区分とイテレーション
- レポート
- ポータル

以降のセクションでは、MSF Agile のプロセス テンプレートを使用するときに利用可能な重要な初期設定を詳しく説明します。

作業項目

MSF Agile プロセス テンプレートには、次の作業項目の種類が含まれています。

- **バグ。**アプリケーションにおける問題、または潜在的な問題を表します。
- **リスク。**プロジェクトにマイナスの影響を与える可能性のあるイベントまたは状況を表します。
- **シナリオ。**作成中のシステムを介したユーザーとのやりとりにおける単一のパスを表します。
- **タスク。**チーム メンバが実施する作業の詳細な項目を特定します。
- **サービス品質要求。**セキュリティ、パフォーマンス、管理性の要件など、機能以外の要件を表します。

MSF Agile プロセス テンプレートに基づいて新しいチーム プロジェクトを作成する場合には、次の作

業項目が作成されます。これは、プロジェクトの開始時に実行しなければならない一般的なタスク セットを提供することにより、ユーザーの作業を支援します。

- **セットアップ: アクセス許可の設定。**このタスクの目的は、ビルド サービス (Build Service)、プロジェクト管理者 (Project Administrator)、貢献者 (Contributor)、または閲覧者 (Reader) の 4 つのセキュリティ グループのいずれかに対してチーム メンバを追加することです。
- **セットアップ: ソース コードの移行。**このタスクの目的は、既存のプロジェクトを Microsoft Visual Studio Team Foundation Server (TFS) へ移行する場合に、既存のソース コードを Microsoft Visual SourceSafe® から移行することです。チーム プロジェクトに対するチーム メンバのアクセス許可を付与する前に、ソース コードの移行を完了しなければなりません。
- **セットアップ: 作業項目の移行。**既存のプロジェクトを TFS に取り込む場合には、バグやタスクなどの作業項目を、Clearquest から、またはカンマ区切り形式の CSV ファイルから移行することができます。チーム プロジェクトに対するチーム メンバのアクセス許可を付与する前に、この作業項目の移行を完了しなければなりません。
- **セットアップ: チェックイン ポリシーの設定。**このタスクの目的は、ソース コードのチェックインに関するビジネス ルールまたはポリシーをセットアップすることです。
- **セットアップ: ビルドの構成。**このタスクの目的は、最初のソース ツリーを作成し、ビルドを定期的に (通常は毎日) 実行するようセットアップすることです。
- **セットアップ:インストールおよび作業の開始のためにユーザーにメールを送信。**このタスクの目的は、チームのメンバが接続しなければならない TFS について、およびチーム プロジェクトで作業を開始するために、どのチーム プロジェクトを使用するかについて、情報を電子メールでメンバに送信することです。
- **ビジョン ステートメントの作成。**このタスクの目的は、プロジェクトのビジョン ステートメント、つまりプロジェクトのすべての関係者が共有する、プロジェクトの望ましい結果の概要を作成することです。
- **セットアップ: チーム プロジェクト ポータルにプロジェクトの説明を作成。**このタスクの目的は、新しいチーム プロジェクトについてより適切に説明するために、プロジェクトの目的、目標、ビジョンなどのデフォルトの説明を変更することです。
- **ペルソナの作成。**このタスクの目的は、システムとやりとりするペルソナを作成することです。ペルソナはシステムのターゲット ユーザーであるため、アプリケーションの設計について考察する

ときに、ペルソナを使用することができます。

- **イテレーション期間の定義。**このタスクの目的は、プロジェクトで使用するイテレーションのサイクルを定義することです。この長さは、プロジェクトの規模および複雑さによって異なります。
- **テストのしきい値を明記したテスト アプローチ ワークシートの作成。**このタスクの目的は、プロジェクトのイテレーションのテスト戦略を最初から理解することです。テストのアプローチを理解すると、テスト タスクをより効率よくスケジュールし、開発者がテストに関する考慮事項に留意して実装するうえで役に立ちます。
- **シナリオ リストのブレーンストーミングおよび優先度の決定。**このタスクの目的は、主な使用シナリオを特定し、優先順位を付けることです。
- **サービス品質要求リストのブレーンストーミングおよび優先度の決定。**このタスクの目的は、セキュリティ、パフォーマンス、管理性のシナリオなど、機能以外の QoS 要求を特定することです。
- **セットアップ: プロジェクトの構造の作成。**このタスクの目的は、開発チームが作業する区分を取得するプロジェクト構造を作成することです。
- **イテレーション計画の作成。**このタスクの目的は、開発作業をどのようにイテレーションに分割するかを決定することです。

レポート

MSF Agile プロセス テンプレートにおいてデフォルトで使用できるレポートは次のとおりです。

- **バグ (優先度順)**。正しいバグが見つかっているかどうかを表します。このレポートは、優先度の高いバグと低いバグが検出されている割合を示します。
- **バグ率。**どのくらい効率よくバグが検出、修正、および解決されているかを表します。このチャートは、新しいバグ、バグのバックログ、バグの解決に対する時系列の傾向を示しています。
- **ビルド。**ビルドのクオリティを表します。このレポートによって、ビルドのクオリティ、およびその他の詳細情報も含めて、使用できるビルドの一覧が提供されます。
- **プロジェクト速度。**チームが作業をどのくらい迅速に完了しているかを表します。このレポートは、計画された作業をチームがどのくらい迅速に完了しているか、および完了率の変化を日ごとに表します。
- **品質指標。**ソフトウェアの品質を表します。このレポートは、プロジェクトの状態を追跡するため

に、テスト結果、バグ、コード カバレッジ、およびコード チャーンを 1 つのレポートで収集します。

- **ロード テストの概要。**このレポートは、アプリケーションにおけるロード テストの結果を示しています。
- **回帰。**このレポートは、以前は合格したけれども、現在は不合格になっているすべてのテストの一覧を示します。
- **再アクティブ化。**再度アクティブになっている作業項目の数を表します。このレポートは、解決およびクローズが時期尚早であった作業項目を示します。
- **関連作業項目。**作業項目間の依存関係を表します。このレポートは、依存関係を追跡できるように、他の作業項目にリンクされている作業項目の一覧を示します。
- **残存作業。**作業がどのくらい残っているか、またそれがいつ完了する予定かを表します。このレポートは、時間とともに残存、解決、およびクローズした作業の量を示します。残りの作業の今後の傾向を予測すると、コードが完了するのがどの時点かを推定することができます。
- **•計画していなかった作業**。未計画の作業がどのくらいあるかを表します。すべての作業量に対する残りの作業量を示し、また、計画していたアクティビティと計画していなかったアクティビティを区別します。
- **トリアージ。**優先順位を付ける必要がある作業項目を表します。このレポートは、まだ提案の状態にある作業項目を示します。
- **作業項目。**どの作業項目がアクティブかを表します。このレポートは、すべての有効な作業項目を示します。
- **作業項目 (所有者)。**チームの各メンバにどのくらいの作業が割り当てられているかを表します。このレポートは、チーム メンバごとの作業項目を示します。
- **作業項目 (状態)。**アクティブ、解決済み、および終了になっている作業項目の数を表します。このレポートは、アクティブ、解決済み、および終了になっているすべての作業項目の一覧を示します。

グループとアクセス許可

MSF Agile プロセス テンプレートにおいてデフォルトで使用できるグループは次のとおりです。

- **閲覧者（Reader）。**このグループのメンバは、チーム プロジェクトに対して読み取り専用のアクセス権を持っています。
- **貢献者（Contributor）。**このグループのメンバは、チーム プロジェクトの項目を追加、修正および削除できます。
- **ビルド サービス（Build Services）。**このグループのメンバは、チーム プロジェクトに対してビルド サービス アクセス許可を持ちます。このグループは、サービス アカウントに対してのみ使用されます。
- **プロジェクト管理者（Project Administrator）。**このグループのメンバは、チーム プロジェクトのすべての操作を実行することができます。

ソース管理

MSF Agile は、ソース管理において次のデフォルト設定を使用します。

- **複数チェックアウト。**既定では、MSF Agile は、複数のチーム メンバが同じファイルを同時に作業できるよう、複数チェックアウトを許可しています。何らかの競合が発生した場合は、チェックイン時に解決する必要があります。
- **アクセス許可。**ソース管理におけるデフォルトのアクセス許可は次のとおりです。
    - **プロジェクト管理者。**すべての権限を持っています。
    - **ビルド サービス。**読み取り、変更の保留、チェックイン、ラベル付け、ビルドの開始およびビルドの編集を行う権限があります。
    - **貢献者。**読み取り、変更の保留、チェックイン、チェックアウト、ラベル付け、およびビルドの開始を行う権限があります。
    - **閲覧者。**ソース管理への読み取り専用アクセス権のみがあります。

区分とイテレーション

MSF Agile の既定の状態のプロセス テンプレートには、区分またはイテレーションに対する分類の構造がありません。このような場合は、プロジェクトのコンポーネントまたは機能に基づいて区分を作成する方法を推奨します。イテレーションには、（計画、開発、およびテストなどの） 主要なアクティ

ビティの特定の組を繰り返す、時間に基づいたサイクルを使用できます。

MSF for Agile Software Development の実践例

次の例は、MSF for Agile Software Development プロセスが Microsoft の *patterns & practices* チーム、および Microsoft 以外の開発チームでどのように採用され、使用されているかを示しています。

例 *1: patterns & practices* チーム

次の例は、MSF Agile プロセスを使用して、一般的な *patterns & practices* プロジェクトがどのように実行されるかを示しています。

イテレーション 0 の新しいプロジェクト

- プロダクト マネージャ
  1. プロジェクトの要件を収集するために、お客様および関係者とやりとりします。これらの要件は、プロジェクト バック ログという名前の Microsoft Office Word の文書に取得されます。
  2. Microsoft Office PowerPoint® を使用して、プロジェクトに対するビジョン ステートメントを作成します。
  3. お客様およびさまざまな関係者とブレーンストーミングし、プロジェクトの要件およびビジョンを扱うシナリオを定義します。
  4. プロジェクト マネージャおよび他の関係者と連携し、シナリオの優先順位を決定します。
- プロジェクト マネージャ:
  1. シナリオを作業項目として TFS 内に取り込みます。
  2. プロジェクトの規模、および実現可能性によって、イテレーション サイクル期間を決定します。

イテレーション前の計画

- プロジェクト マネージャは、イテレーションの期間中にどのシナリオを作業するべきかを、優先順に基づいて決定します。

- プロダクト マネージャとプロジェクト マネージャは、シナリオに対するサービス品質 (QoS) 要件を作成します。QoS はシナリオにリンクされます。

イテレーションの計画

- プロジェクト マネージャ:
  1. 開発者および他のチーム メンバと連携して、シナリオを開発タスクに分割します。
  2. TFS 内に開発タスクを取り込み、シナリオにリンクさせます。
  3. それぞれの開発タスクに対して受け入れ基準を定義します。
  4. QoS 要求をテスト タスクに分割します。
  5. TFS 内にテスト タスクを取り込み、QoS にリンクさせます。
  6. それぞれのテスト タスクに対して受け入れ基準を定義します。
  7. タスクをスケジュールし、割り当てます。
- 開発者は、それぞれの開発タスクを見積もります。

  重要 – タスク (開発者のシナリオ) を実施するのに 1 日または 2 日よりも長くかかりそうな場合は、これらのタスクをサブシナリオに分割する必要があります。
- テスト担当者は、それぞれのテスト タスクに対して見積もります。

イテレーションの期間中

- プロジェクト マネージャはイテレーションをガイドします。
- 開発者は開発タスクのコードを記述し、受け入れ基準を満たしたら、タスクを終了します。
- テスト担当者は、自身に割り当てられているテスト タスクを実行し、特定された問題に対して新しいバグ (作業項目) を作成します。

イテレーションの後

- プロジェクト マネージャ:
  1. プロジェクトの進捗を評価し、現在のイテレーションで完了していないシナリオについて、再度優先順位を付けます。

2. 関係者にステータス レポートを提供します。
3. 優先順位に基づいて、次のイテレーション中にどのシナリオを実行すべきかを決定します。

- プロダクト マネージャ
  1. 新しく検出されたシナリオを追加します。
  2. (必要に応じて) シナリオに再度優先順位を付けます。
  3. プロジェクト マネージャと共に、プロジェクトに対する QoS 要求を作成します。QoS はシナリオにリンクされます。

例 *2:* フィールド カスタマ エンゲイジメント

次の例は、フィールド カスタマ エンゲイジメントで MSF Agile プロセスがどのように使用されているかを示します。

イテレーション 0 の新しいプロジェクト

- ビジネス アナリスト:
  1. 短い (1 ページの) ビジョン ステートメントを生成します。
  2. 入力を提供できる現場のお客様を特定し、システムに対するペルソナを作成します。
  3. お客様と一緒にシナリオについてブレーンストーミングします (名前のみ)。
  4. お客様と一緒にシナリオの優先順位を決めます。
  5. 次のイテレーションに対するシナリオを記述します。
- プロジェクト マネージャ:
  1. 開発者を集めて見積もりを取得します。見積もりは、大まかな概算見積もりです。
  2. コストを見積もった結果、優先順位が変わるかどうか確認します。
  3. 次のイテレーションに対してシナリオをスケジュールします。
- アーキテクトは、シナリオをアーキテクチャ タスクに分割します。
- 開発者:
  1. シナリオを開発タスクに分割します。
  2. 適切なビルド戦略 (可能であれば、継続的な統合) を定義します。
- テスト開発者は、シナリオをテスト タスクに分割します。

イテレーションの期間中

- プロジェクト マネージャ:
  1. イテレーションをガイドします。
  2. プロジェクトをガイドします。
- 設計者は、ソリューション アーキテクチャを定義します。
- 開発者は、開発タスクを実行します。
- テスト担当者は、シナリオをテストします。

イテレーション 0 の後

この時点で、タスクはわずかに変わっています。

- ビジネス アナリスト:
  1. (必要に応じて) ペルソナを更新します。
  2. 新しく検出されたシナリオを追加します。
  3. (必要に応じて) シナリオに再度優先順位を付けます。
  4. 次のイテレーションに対するシナリオを記述します。
- プロジェクト マネージャ:
  1. 新しいシナリオを見積もります。
  2. 次のイテレーションに対してシナリオをスケジュールします。
- アーキテクトは、シナリオをアーキテクチャ タスクに分割します。
- 開発者:
  1. シナリオを開発タスクに分割します。
  2. ビルド プロセス (可能であれば、継続的な統合) を更新します。
- テスト開発者は、シナリオをテスト タスクに分割します。

MSF Agile プロセス テンプレートのカスタマイズ

MSF Agile プロセス テンプレートをカスタマイズして特定の組織の要件に合わせるには、次の 2 つ

の方法を使用できます。

- **XML ファイルを手動でカスタマイズする。**手動によるカスタマイズではエラーが発生しやすくなりますが、プロセス テンプレートのカスタマイズをきめ細かくコントロールできます。詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243782(VS.80).aspx の「プロセス テンプレートのカスタマイズ」を参照してください。
- **Process Template Editor**。TFS のエクスペリエンスを改善する一連の拡張機能、ツール、コマンドライン ユーティリティである Visual Studio 2005 Team Foundation Server Power Tool の最新バージョンには、ユーザー インターフェイス ベースのツールが用意されており、これを使用してプロセス テンプレートを参照およびカスタマイズすることができます。TFS に接続すると、このツールを使用して、作業項目の種類の定義、およびアクティブなプロジェクトのグローバル リストをカスタマイズできます。詳細については、「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

まとめ

MSF Agile プロセス テンプレートは、ソフトウェア開発のライフサイクルに関するさまざまな役割によって実行される、一連のタスクを定義します。MSF Agile プロセス テンプレートは、アジャイル プロセスをサポートするための作業項目、グループとアクセス許可、ソース管理、区分とイテレーション、レポート、およびチーム ポータルを定義します。

プロセスの要件に対してデフォルトのプロセス テンプレートが十分でない場合は、手動で XML プロセス定義ファイルをカスタマイズしてテンプレートをカスタマイズしたり、TFS Power Tool に付随している Process Editor Tool を使用してカスタマイズしたりできます。

参考資料

- MSF Agile プロセス テンプレートをダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyID=EA75784E-3A3F-48FB-824E-828BF593C34D&displaylang=en へアクセスしてください。
- Process Template Editor を含めて Team Foundation Server Power Tools をダウンロードす

るには、http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx へアクセスしてください。

- プロセス テンプレートのカスタマイズの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194945(VS.80).aspx の「プロセス テンプレートのカスタマイズの概要」を参照してください。
- 手動によるプロセス テンプレートのカスタマイズの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243782(VS.80).aspx の「プロセス テンプレートのカスタマイズ」を参照してください。

第 7 部

レポーティング

第 7 部の内容

- **レポーティングの説明**

第 15 章 レポーティングの説明

目的

- Microsoft® Visual Studio® Team Foundation Server (TFS) のレポーティングのアーキテクチャについて説明します。
- TFS のレポーティングを構成するコンポーネントについて認識します。
- 使用できる各レポートの目的について説明します。
- どのプロセス テンプレートにどのレポートが含まれているかを理解します。
- レポートのカスタマイズおよび新しいレポートの作成を行います。

概要

この章では、TFS のレポーティング アーキテクチャについて、および新しいチーム プロジェクトで使用できる一般的なレポートについて説明します。また、一般的なレポーティング シナリオを、TFS で使用できるレポートに結び付け、既存のレポートをカスタマイズしたり、新しいレポートを作成したりするための一般的な理由について説明します。TFS のレポーティングを使用すると、チーム プロジェ

クトのさまざまな側面について集約されたデータを参照することができます。この情報を使用してプロジェクトの進捗、プロジェクトの状態、および開発チームとテスト チームの有効性を分析することができます。

TFS のレポーティングでは、Microsoft SQL Server™ 2005 Reporting Services を使用してレポートを作成、管理および実行します。各プロセス テンプレートには、あらかじめ定義されている一連のレポートが含まれています。これらのレポートは、プロジェクトの作成時にプロジェクトのレポート フォルダに配置されます。Reporting Services を使用して、これらのレポートを修正することも、プロジェクトに対するカスタム レポートを作成することもできます。既存のプロセス テンプレートに対して新しいレポートを追加して、レポートを他のチーム プロジェクトで使用できるようにすることができます。

この章の参照の仕方

この章を使用して、TFS のレポーティングがどのように機能するか、およびプロジェクトの状態およびステータスを評価するうえでどのように役に立つかを理解します。この章を最大限に利用するには、次のようにします。

- TFS のレポーティングを使用する一般的な理由について理解し、それぞれの標準レポートの目的について理解するには、**「シナリオとソリューション」セクションを読んでください。**
- レポーティング システムはどのようなコンポーネントで構成されているか、およびそれらがどのような相互関係にあるかを理解するには**「物理的なアーキテクチャ」セクションを読んでください。**
- レポートのカスタマイズおよび作成で使用できるメカニズムについて学習するには、**「レポートのカスタマイズ」セクションを読んでください。**
- このガイドの「方法」を読みます。この章で説明しているさまざまな手順について順を追って理解するには、次の「方法」を読んでください。
    - Visual Studio 2005 Team Foundation Server でレポートをカスタマイズする方法
    - Visual Studio 2005 Team Foundation Server のカスタム レポートを作成する方法
    - Visual Studio 2005 Team Foundation Server でリスク オーバー タイム レポートを作成する方法

シナリオとソリューション

プロジェクト マネージャおよびチーム リーダーが、進行中のプロジェクトの最新の内容を把握するために、最もよく使用される方法がレポートです。新しいチーム プロジェクトを作成すると、選択したプロセス テンプレートに基づいて、一連のレポートが生成されます。これらのレポートは、プロジェクトの Microsoft Office SharePoint® のポータル サイトから、または Visual Studio 内のチーム エクスプローラのレポート ノードから使用できます。

TFS レポートを使用すると次のような一般的な疑問に対する回答が得られます。

- アプリケーションはいつ頃出荷できるようになるか?
- 計画に従って作業が進行しているか?
- ビルドのクオリティはどうか?
- 定義されているシナリオに対する開発のステータスはどうか?
- 開発作業はどのくらい迅速に進んでいるか?
- バグは修正されているか?
- バグは再発しているか?

Team Foundation Server のレポート

Microsoft Solution Framework (MSF) for Agile Software Development (MSF Agile) および MSF for CMMI® Process Improvement (MSF CMMI) の両方のプロセス テンプレートのそれぞれには、デフォルトで一連のレポートが含まれています。

バグ

プロセス テンプレート内のバグ関係のレポートを使用して、どのようなタイプのバグが生成および修正されているかを確認し、傾向を特定することができます。バグ関係のレポートとして、次のものを使用できます。

- **バグ (優先度順)**。正しいバグが見つかっているかどうかを表します。このレポートは、優先度の高いバグと低いバグが検出されている割合を示します。このレポートは、提供されている両方のプロセス テンプレートで使用できます。
- **バグ率。**どのくらい効率よくバグが検出、修正、および解決されているかを表します。このレポートは、新しいバグ、バグのバックログ、バグの解決に対する時系列の傾向を示しています。このレポートは、提供されている両方のプロセス テンプレートで使用できます。

リリース管理

リリース管理レポートは、ソフトウェアがリリースに対してどのくらい適合しつつあるかを判断します。リリース管理レポートとして、次のものを使用できます。

- **実際の品質と計画された速度。**品質を損なわずにどのくらいのシナリオを完了できるかを示します。このレポートは、各イテレーションに対して、全体の品質に対する見積もりの規模の関係を表します。このレポートは、両方のプロセス テンプレートで使用できます。
- **ビルド。**ビルドの品質を表します。このレポートは、ビルドの品質、およびその他の詳細情報も含めて、使用できるビルドの一覧を提供します。このレポートは MSF for CMMI Process Improvement で使用できます。
- **品質指標。**ソフトウェア品質について表します。このレポートは、プロジェクトの状態を追跡するために、テスト結果、バグ、コード カバレッジ、およびコード チャーンを 1 つのレポートに収集します。このレポートは MSF Agile および MSF CMMI で使用できます。
- **速度。**チームが作業をどのくらい迅速に完了しているかを表します。このレポートは、計画された作業をチームがどのくらい迅速に完了しているか、および完了率の変化を日ごとに表します。このレポートは MSF Agile および MSF CMMI で使用できます。
- **シナリオの詳細。**どのシナリオに対してアプリケーションを作成しているかを示します。このレポートは、完了のステータス、リスク、テストの進捗など、各シナリオに関する情報を提供します。このレポートは MSF CMMI で使用できます。

テスティング

テスティング レポートを使用して、テスティングの有効性および進捗を監視することができます。テスト レポートとして、次のものを使用できます。

- **回帰。**ある時点でテストに合格したけれども、現在は合格していないテストを表します。このレポートは、以前は合格したけれども、現在は不合格になっているすべてのテストの一覧を示します。このレポートは MSF CMMI で使用できます。
- **必要条件テスト履歴および概要。**シナリオおよび要件がどのくらい適切にテストされたかを表します。このレポートは、定義されているシナリオおよび要件に対するテストの進捗を表します。このレポートは MSF CMMI で使用できます。
- **アクティブなバグを伴わないで失敗しているテスト。**既知のすべての欠陥に対してそれを追跡するためのバグが存在するか。 このレポートは、不合格で、かつ未解決のバグに関連付けられていないテストを表します。このレポートは MSF CMMI で使用できます。
- **アクティブなバグを伴って成功しているテスト。**バグ一覧は最新で、アプリケーションの品質と整合性がとれているか。 このレポートは、テストが合格している古いバグを表します。このレポートは MSF CMMI で使用できます。
- **ロード テストの概要。**アプリケーションのパフォーマンスに関するロード テストが、どのような結果かを表します。このレポートは、アプリケーションにおけるロード テストの結果を示しています。このレポートは MSF Agile で使用できます。

作業項目

作業項目レポートを使用して、プロジェクトの現在の状態、および現在の進捗を評価できます。作業項目レポートとして、次のものを使用できます。

- **懸案事項およびブロックされた作業項目。**未解決の問題がどのくらい残っているかを示します。このレポートは、未解決の問題、およびその解決に関する傾向を表します。このレポートは MSF CMMI で使用できます。
- **再アクティブ化。**再度アクティブになっている作業項目の数を表します。このレポートは、解決およびクローズが時期尚早であった作業項目を示します。このレポートは MSF Agile および MSF

CMMI で使用できます。

- **関連作業項目。**どの作業項目が他の作業項目に依存しているかを表します。このレポートは、依存関係を追跡できるように、他の作業項目にリンクされている作業項目の一覧を示します。このレポートは MSF CMMI で使用できます。
- **残存作業。**作業がどのくらい残っているか、またそれがいつ完了する予定かを表します。このレポートは時間とともに残存、解決、およびクローズした作業の量を示します。残りの作業の今後の傾向を予測すると、コードが完了するのがどの時点かを推定することができます。このレポートは MSF Agile および MSF CMMI で使用できます。
- **トリアージ。**優先順位を付ける必要がある作業項目を表します。このレポートは、まだ提案の状態にある作業項目を示します。このレポートは MSF CMMI で使用できます。
- **計画していなかった作業**。計画していなかった作業がどのくらいあるかを表します。このレポートは、すべての作業量に対する残りの作業量を示し、また、計画していたタスクと計画していなかったタスクを区別します。このレポートは MSF Agile および MSF CMMI で使用できます。
- **作業項目。**どの作業項目がアクティブかを表します。このレポートは、すべてのアクティブな作業項目を示します。このレポートは MSF CMMI で使用できます。
- **作業項目 (所有者)。**チームの各メンバにどのくらいの作業が割り当てられているかを表します。このレポートは、チーム メンバごとの作業項目を示します。このレポートは MSF CMMI で使用できます。
- **作業項目 (状態)。**アクティブ、解決済み、および終了になっている作業項目の数を表します。このレポートは、状態ごとの作業項目の一覧を示します。このレポートは MSF CMMI で使用できます。

レポートのカスタマイズ

MSF のどちらのプロセス テンプレートにも存在しないレポートが必要な場合があります。このような場合は、次の 3 つの方法のいずれかでレポートをカスタマイズすることができます。

- **既存のレポートでフィルタを使用する。**多くのレポートには、レポートのフィルタに使用できるパラメータが用意されています。たとえば、日付、区分、イテレーション、優先度などのフィルタを使用できます。レポートで提供されているデータの一部分を参照するには、これらのフィルタを使

用します。これらのフィルタは一時的なものであり、そのレポート以外をブラウズした場合には解除されます。

- **既存のレポートをカスタマイズする。**必要なレポートが既存のレポートに類似している場合は、既存のレポートのコピーを作成し、それを修正するのが最も簡単な方法です。たとえば、時間の経過に伴うリスクをプロットし、チームがプロジェクト リスクに対してどのようにうまく処理しているかを分析することができます。
- **新しいレポートを作成する。**新しいレポートを最初から作成できます。

既存のレポートを修正するか、または最初から新しいレポートを作成した場合には、チームの他のメンバが使用できるように、それらのレポートをレポート サーバーへ公開することができます。既存のレポートを修正したい、または新しいレポートを作成したい場合には、次のいずれかの方法を使用できます。

- Microsoft Office Excel® を使用して、レポーティング データベースのデータからピボット テーブルを作成します。
- Visual Studio に新しいレポート サーバー プロジェクトを作成し、新しいレポートを作成するか、または既存のレポートをインポートします。

Visual Studio に新しいレポート サーバー プロジェクトを作成する方法は、レポートの作成および修正において最も強力で柔軟な方法です。

注意:チームのレポーティング サイトからレポート ビルダを使用することができます。ただしこのツールは、Visual Studio のレポーティング シナリオに対しては完全にはサポートされていないため、推奨されません。

詳細情報

- 既存のレポートのカスタマイズに関する詳細については、「Visual Studio Team Foundation Server でレポートをカスタマイズする方法」を参照してください。
- カスタム レポートの作成に関する詳細については、「Visual Studio Team Foundation Server の

カスタム レポートを作成する方法」を参照してください。

- リスク オーバー タイム レポートを作成するための手順ごとのガイドは、「Visual Studio Team Foundation Server のリスク オーバー タイム レポートを作成する方法」を参照してください。

物理的なアーキテクチャ

Team Foundation Server は SQL Server 2005 上に構築されており、SQL Server Analysis Services を使用してデータを集約し、レポートを実行します。Microsoft Excel または Visual Studio 2005 レポート デザイナ を使用して新しいレポートを作成することができます。レポートは SQL Server 2005 Reporting Services でホストされており、レポート サーバーの Web サイト、チームの SharePoint プロジェクト ポータル、またはチーム エクスプローラのレポート ノードから参照できます。図 15.1 は、レポーティングの物理的なアーキテクチャを表しています。

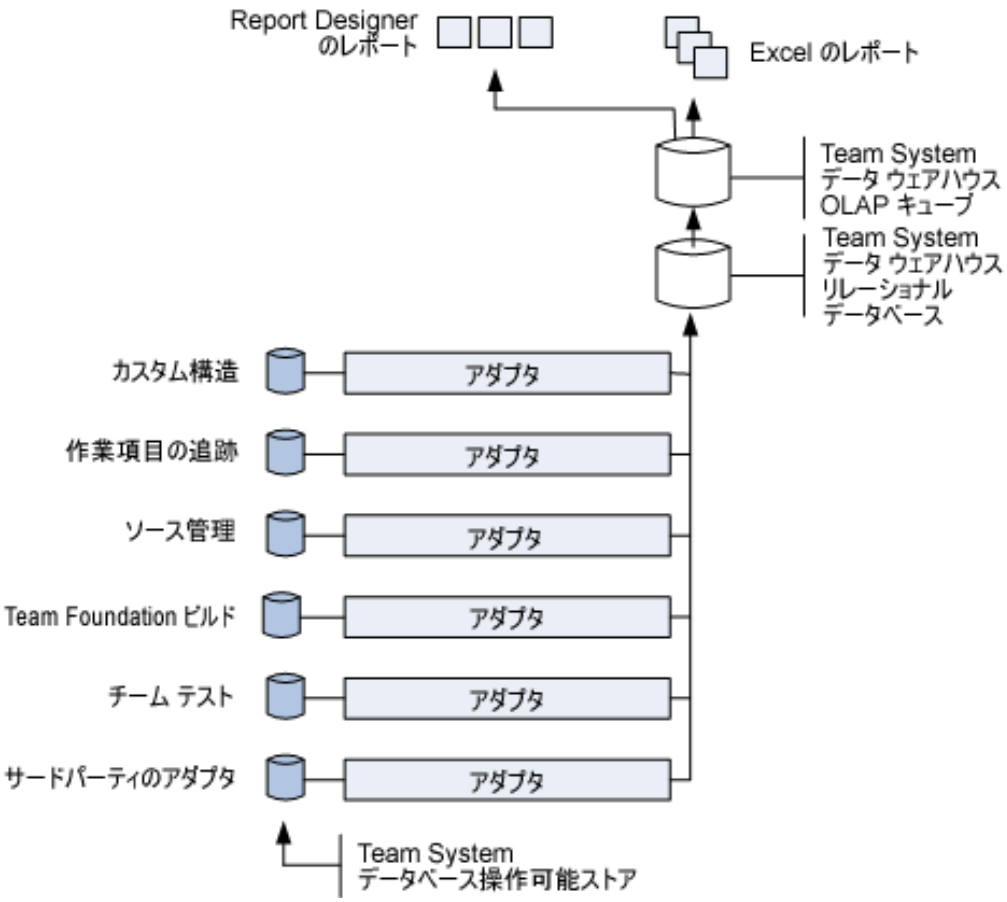


図 *15.1* レポーティングの物理的なアーキテクチャ

TFS の各コンポーネントは、それぞれ一連のトランザクション データベースを保持しています。これには、作業項目、ソース管理、テスト、バグ、およびチーム ビルドが含まれています。このデータはリレーショナル データベースに集約されます。次にこのデータは、傾向に基づいたレポーティングおよびより高度なデータ分析をサポートするために、オンライン分析処理 (OLAP) キューブに配置されます。

TfsWarehouse のリレーショナル データベースは、トランザクションよりも、データ クエリで使用することを意図して設計されたデータ ウェアハウスです。データは、トランザクション処理用に最適化されている、TFS のさまざまなデータベースから、レポーティングのためにこのウェアハウスへ送信されます。ウェアハウスは プライマリ レポーティング ストアではありませんが、これを使用してレポートを作成することができます。TfsReportDS データ ソースは、リレーショナル データベースを指しています。Team System のデータ ウェアハウス OLAP キューブは、SQL Server Analysis Services を介してアクセスされる OLAP のデータベースです。このキューブは、「今月と先月でクローズされたバグ件数の比較は?」といったデータの傾向分析を行うレポートに有用です。TfsOlapReportDS データ ソースは、分析サービス データベースにおける Team System のデータ ウェアハウス OLAP キューブを指しています。

レポーティング システムのコンポーネント

レポーティング システムは、次のサーバー側コンポーネントとクライアント側コンポーネントで構成されています。

サーバー側のコンポーネント

サーバー側のコンポーネントには次のものがあります。

- **レポート サーバー データベース。**これらのデータベースには、レポート定義、履歴レポート、および構成データが含まれています。
- **レポート サーバー Web サービス。**この Web サービスは、レポート サーバーに対するプログラムのアクセスを提供します。

- **レポート マネージャ Web サイト。**このサイトは、Web ブラウザを介してレポート サーバーに対するユーザー アクセスを提供します。
- **Windows サービス。**このサービスは、レポートのスナップショットのスケジューリングおよびデリバリを提供します。

クライアント側のコンポーネント

クライアント側のコンポーネントには次のものがあります。

- **ブラウザ。**このコンポーネントは、レポート マネージャの Web サイトに対するアクセスを提供します。
- **チーム エクスプローラ。**このコンポーネントは、Visual Studio を介してレポートに対するアクセスを提供します。

レポートの開発ツール

開発ツールには次のものが含まれます。

- **Business Intelligence Designer Studio (BIDS)**。開発者はこのコンポーネントを使用して、Visual Studio 2005 からレポートを設計および配置することができます。
- **Excel**。Excel を使用して、レポーティング ストアからピボット テーブルを生成することができます。
- **レポート ビルダ。**エンド ユーザーはこのコンポーネントを使用して、一時的なレポートを設計できます。これは Team Foundation のレポーティング シナリオに対しては完全にサポートされていないため、推奨されません。

まとめ

MSF Agile プロセス テンプレートおよび MSF CMMI プロセス テンプレートにはそれぞれ、バグ、リリース管理、テスティングおよび作業項目の追跡に対するデフォルトのレポート セットが用意されています。

- プロセス テンプレートのバグ関係のレポートを使用して、どのようなタイプのバグが生成されて

いるかを確認し、バグの傾向を特定するうえで役に立てることができます。

- リリース管理レポートは、アプリケーションがリリースに適合しているかを判断するうえで有用です。
- テスティング レポートを使用して、テスティング作業の有効性および進捗を監視することができます。
- 作業項目レポートを使用して、プロジェクトの現在の状態、および現在の進捗を評価できます。

既存のレポートを修正するか、または新しいレポートを作成する場合は、チームのレポーティング サイトから使用できるレポート ビルダを使用する、Excel を使用してレポーティング データベースのデータからピボット テーブルを作成する、Visual Studio に新しいレポート サーバー プロジェクトを作成するなどの方法があります。

参考資料

- Team Foundation Server のレポーティングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194922(VS.80).aspx を参照してください。

第 8 部

チーム環境のセットアップと保守

第 8 部の内容

□Team Foundation Server の配置

□Team Foundation Server に対するインターネット アクセスの提供

第 16 章 Team Foundation Server の配置

目的

- シングルサーバーとマルチサーバーの配置のメリットおよびデメリットについて学習します。
- 組織の要件に合わせた配置トポロジを選択します。

概要

この章では、Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) を配置するための主なアプローチについて概説し、組織で TFS を配置するときに直面する重要な決定ポイントについて説明します。ここでは、2 つの配置オプションについて、およびこれらのオプションをどのように選択したらよいかについて説明します。

TFS の配置オプションには、シングルサーバーとデュアルサーバーのインストレーションの 2 つがあります。シングルサーバーのインストレーションは、データ層とアプリケーション層を 1 つのサーバ

ー上に配置します。デュアルサーバーのインストレーションでは、アプリケーション層とデータ層を別々のサーバーに分けます。また、別のサーバー上にビルド サーバーおよびソース管理プロキシをインストールすることができます。各クライアントはサーバーへのアクセスが必要で、適切なクライアント側のツールをインストールする必要があります。

この章の参照の仕方

この章を使用して、TFS の配置方針を決定します。この章を最大限に利用するには、次のようにします。

- **TFS のアーキテクチャを理解します。**TFS のアーキテクチャは、必ず完全に理解してください。Team Foundation Server のアーキテクチャに慣れていない場合は、「TFS のアーキテクチャ」のセクションを読むか、または詳細については「第 2 章 Team Foundation Server のアーキテクチャ」を読んでください。
- **配置方針を選択します。**組織のニーズに最も合う配置の方針を選択します。まだ方針を選択していない場合は、「配置のシナリオ」のセクションを読んで、どの配置方針がチームに最も適しているかを判断します。

TFS のアーキテクチャ

図 16.1 は Team Foundation のアーキテクチャを示しています。

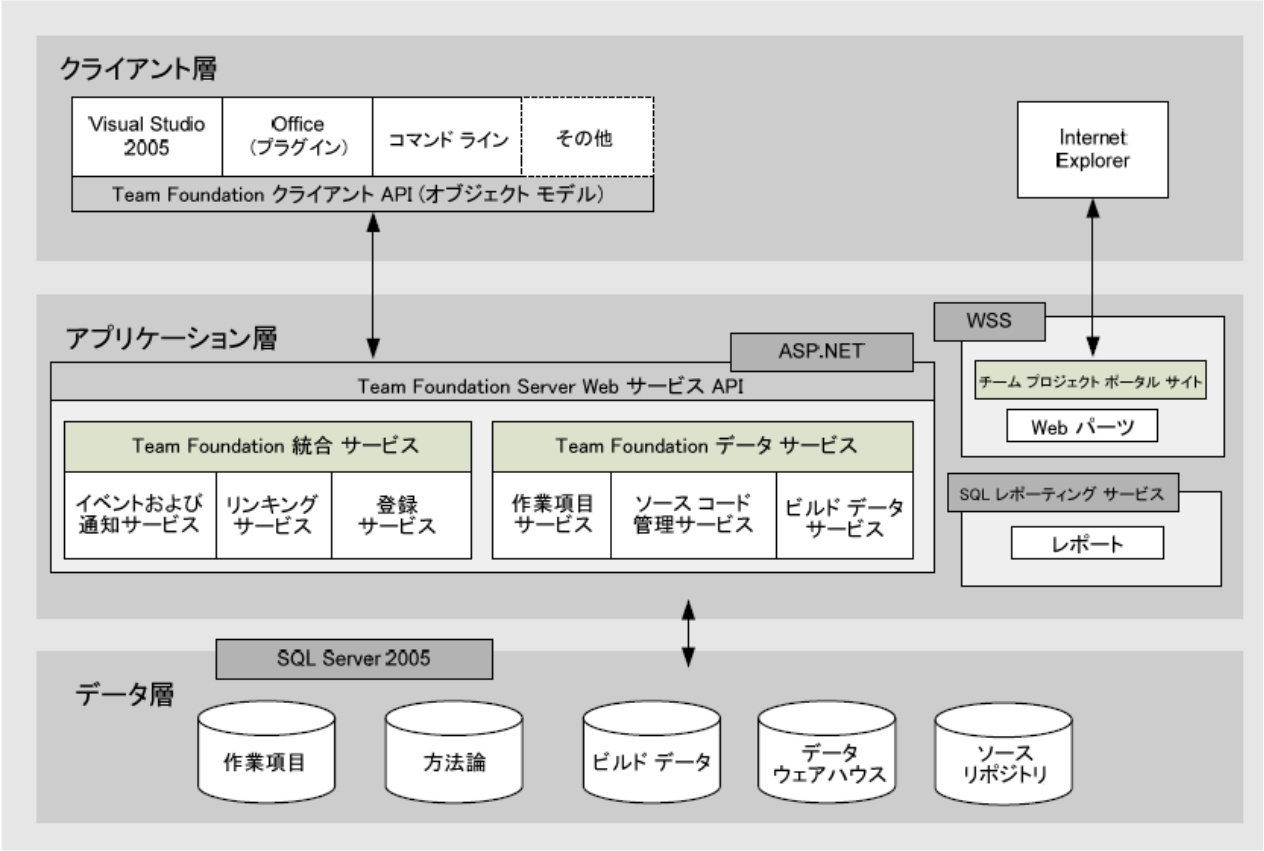


図 *16.1* *TFS* のアーキテクチャ

TFS のアーキテクチャは、データ層、アプリケーション層、およびクライアント層の 3 つの層で構成されています。これらの層は論理的なもので、3 つのすべての層を同じコンピュータにインストールすることもできます。

Team Foundation のデータ層は Microsoft SQL Server™ 2005 から構成されており、作業項目、バージョン管理システム、テスト結果、およびレポートを格納する多数のデータベースがインストールされています。

アプリケーション層には、インターネット インフォメーション サービス (IIS) に統合されている Web ベースのフロント エンド、Team Foundation の Web サービス、および Microsoft Office SharePoint ® サービスが含まれています。アプリケーション層には、ビルド サービスおよびソース管理のプロキシ サーバーも含まれています。

クライアント層には、TFS にアクセスするアプリケーションが含まれています。開発者は、スタンドアロン ツールとして、または Visual Studio 2005 の一部としてインストールされているチーム エクスプローラを使用して、Team Server へ接続します。プロジェクト マネージャは Microsoft Office Excel® または Microsoft Office Project を使用します。サードパーティのツールを使用してサーバーへ接続することもできます。

詳細については、「第 2 章 Team Foundation Server のアーキテクチャ」を参照してください。

配置のシナリオ

TFS は次のように配置することができます。

- シングルサーバー配置
  - ワークグループ内
  - Microsoft Active Directory ® のディレクトリ サービスを使用する
- デュアルサーバー配置

ワークグループを使用するシングルサーバー配置

この配置のアプローチでは、Active Directory のドメイン コントローラを持たないワークグループを作成します。チームの規模が小さい場合は、このインストレーション モードを使用します。このインストレーション モードを使用する場合は、サーバーにログオンするために、各ユーザーはサーバー上にローカル アカウントが必要になります。このタイプの配置では、シングルサーバーにのみインストールすることが可能で、デュアルサーバーのインストレーションはサポートされていません。

Active Directory を使用するシングルサーバー配置

Active Directory がある場合は、配置で 2 つの選択があります。つまり、データ層とアプリケーション層を同じサーバーへインストールする方法と、データ層とアプリケーション層を別のサーバーへインストールする方法です。

組織にとってどの配置のタイプが適しているか

組織にとって、シングルサーバーまたはデュアルサーバーのどちらのインストールが正しい選択かを決めるには、次の内容について検討します。

- **サポートが必要なユーザー数。**400 人を超えるユーザーを計画している場合は、デュアルサーバー配置がニーズに合うかどうか検討してください。
- **TFS でサポートするプロジェクト数。**多数のプロジェクトをサポートしている場合は、デュアルサーバーの TFS 配置がビジネス ニーズに合うかどうか検討してください。TFS の各インスタンスは、最大 5000 個のプロジェクトをサポートできます。5000 個を超えるプロジェクトをサポートしなければならない場合は、複数の Team Foundation Server インスタンスをセットアップすることを検討してください。
- **TFS 専用にできるサーバーがあるか。** シングルサーバーの Team Foundation Server 配置のサーバーは、TFS の機能専用にする必要があります。この TFS は、メール サーバー、ファイル サーバー、他のアプリケーションのデータベース サーバーなど、他の目的に使用することはできません。

シングルサーバー配置のメリット

シングルサーバーの配置を実装するかどうか決定する場合には、次のメリットについて検討します。

- **簡潔性**

  □TFS の配置のすべての側面を単一のサーバー上で管理できます。

  □ユーザーおよびグループに対するすべてのアクセス権および許可を 1 つのサーバー上で構成できます。

  □バックアップおよび保守について 1 つのサーバーのみスケジュールすれば済みます。
- **可用性。**アプリケーション層とデータ層の両方が 1 つのサーバー上にあるため、配置を計画するときに、アプリケーション層とデータ層の間のネットワークの制約や、ネットワークの遅延を考慮する必要はありません。

デュアルサーバー配置のメリット

デュアルサーバーの配置を実装するかどうか決定する場合には、次のメリットについて検討します。

- **拡張性。**シングルサーバー配置は最大 400 人のユーザーを対象として設計されていますが、デュアルサーバー配置では、この制約が最大 2000 人まで拡張されます。
- **フェールオーバー。**保守または修理の場合には、アプリケーション層のサーバーを他のデータ層のサーバーへリダイレクトできます。スタンバイまたはフェールオーバーのアプリケーション層のサーバーとして動作できるような追加のサーバーを構成および配置することはできません。

シングルサーバー配置

図 16.2 は、一般的なシングル サーバー配置を表しています。サーバー上にインストールされているのは、TFS アプリケーション層とデータ層、および SharePoint Services と SQL Server 2005 です。

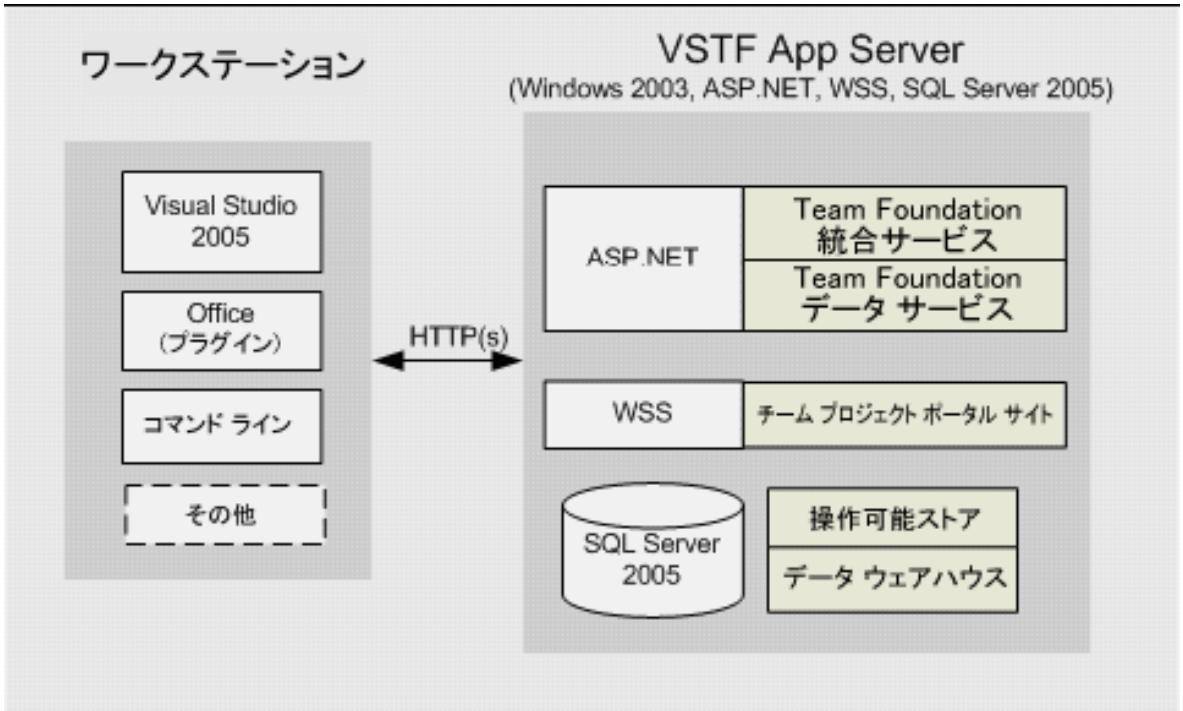


図 *16.2* 一般的なシングルサーバーのインストレーション

デュアルサーバー配置

図 16.3 は、一般的なデュアル サーバー配置を表しています。TFS アプリケーション層は、SharePoint

Services とともに 1 つの層にインストールされます。TFS のデータ層は、SQL Server 2005 とともに別のサーバー上にインストールされます。

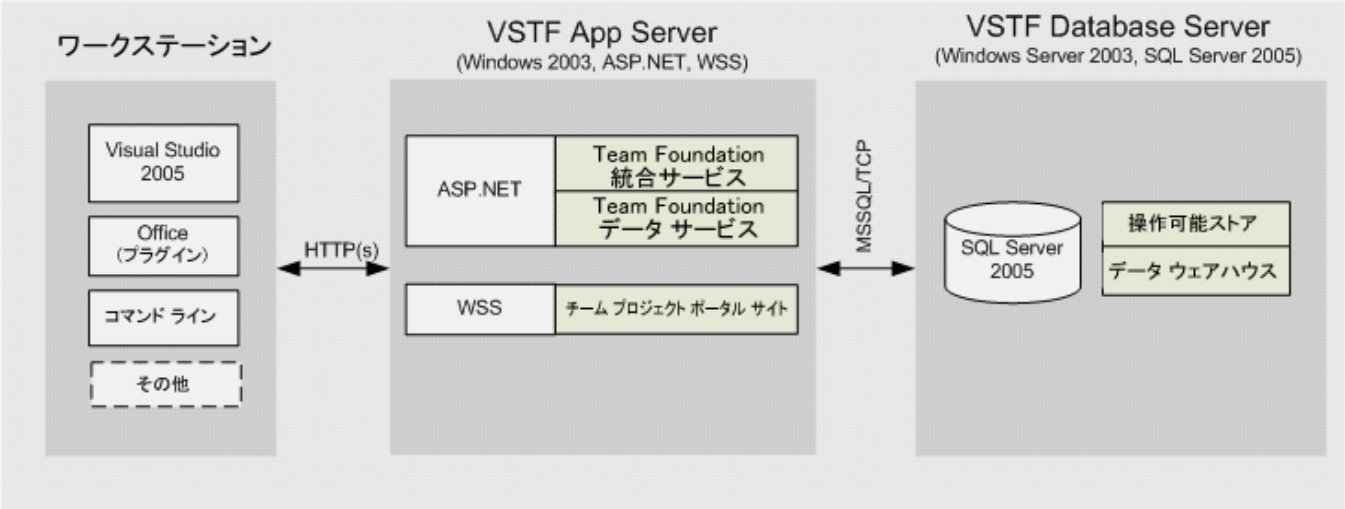


図 *16.3* 一般的なデュアルサーバーのインストレーション

その他のサーバー

シングルサーバーまたはデュアルサーバーのインストレーションで、別のビルド サーバーおよびプロキシ サーバーをインストールできます。これらのサーバーは、アプリケーション層と同じサーバーにインストールすることも、別のサーバーにインストールすることも可能です。

ビルド サーバーのインストレーション

別のサーバー上にビルド サービスを位置付けて、ビルドのパフォーマンスを改善し、アプリケーション層の負荷を軽減することができます。たとえば、アプリケーション層のサーバーのパフォーマンスが、頻繁なビルドによって影響を受けている場合は、ビルド サービスを別のサーバーへ移行することを検討してください。

Team Foundation プロキシ サーバー

Team Foundation のプロキシ サーバーは、ソース管理ファイルのコピーをキャッシュします。ネットワークを介してソース管理サーバーへアクセスしており、大幅な遅延がある場合には、プロキシ サ

ーバーを使用します。

Team Foundation Server のトポロジ

シングルサーバーまたはデュアルサーバーのいずれかのインストレーションに決定したら、いくつかのトポロジを使用することができます。トポロジは簡単なものから複雑なものまであり、各トポロジは、決まった規模のチームに対して設計されています。

簡単なトポロジ

図 16.4 は、TFS アプリケーション層とデータ層のコンポーネントが 1 つのサーバー上に配置されている、簡単なトポロジを表しています。TFS のプロキシ サーバーは、別のサーバーに対して配置されます。このサーバーには、同じドメインのクライアント ワークステーションからアクセスできます。

この構成は、最大 400 人のユーザーの開発チームやパイロット プロジェクトに適しています。

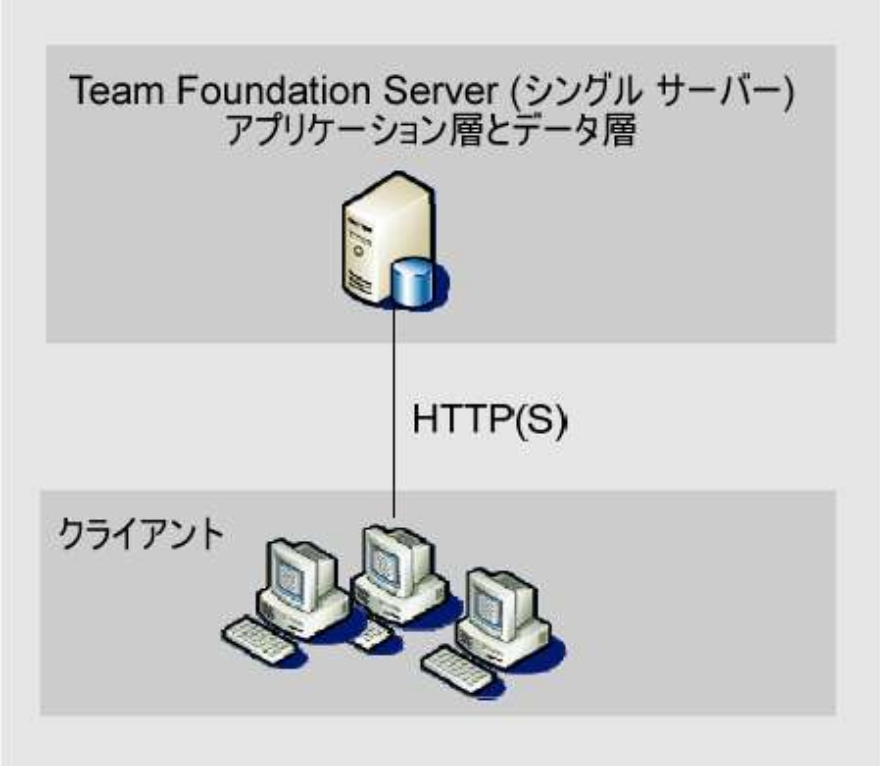


図 *16.4* 簡単な *Team Foundation Server* のトポロジ

中規模のトポロジ

図 16.5 は 異なる層にインストールされている TFS を表します。アプリケーション サービスは 1 つのアプリケーション層のノードに配置され、データベースは別のデータ層のノードに配置されます。

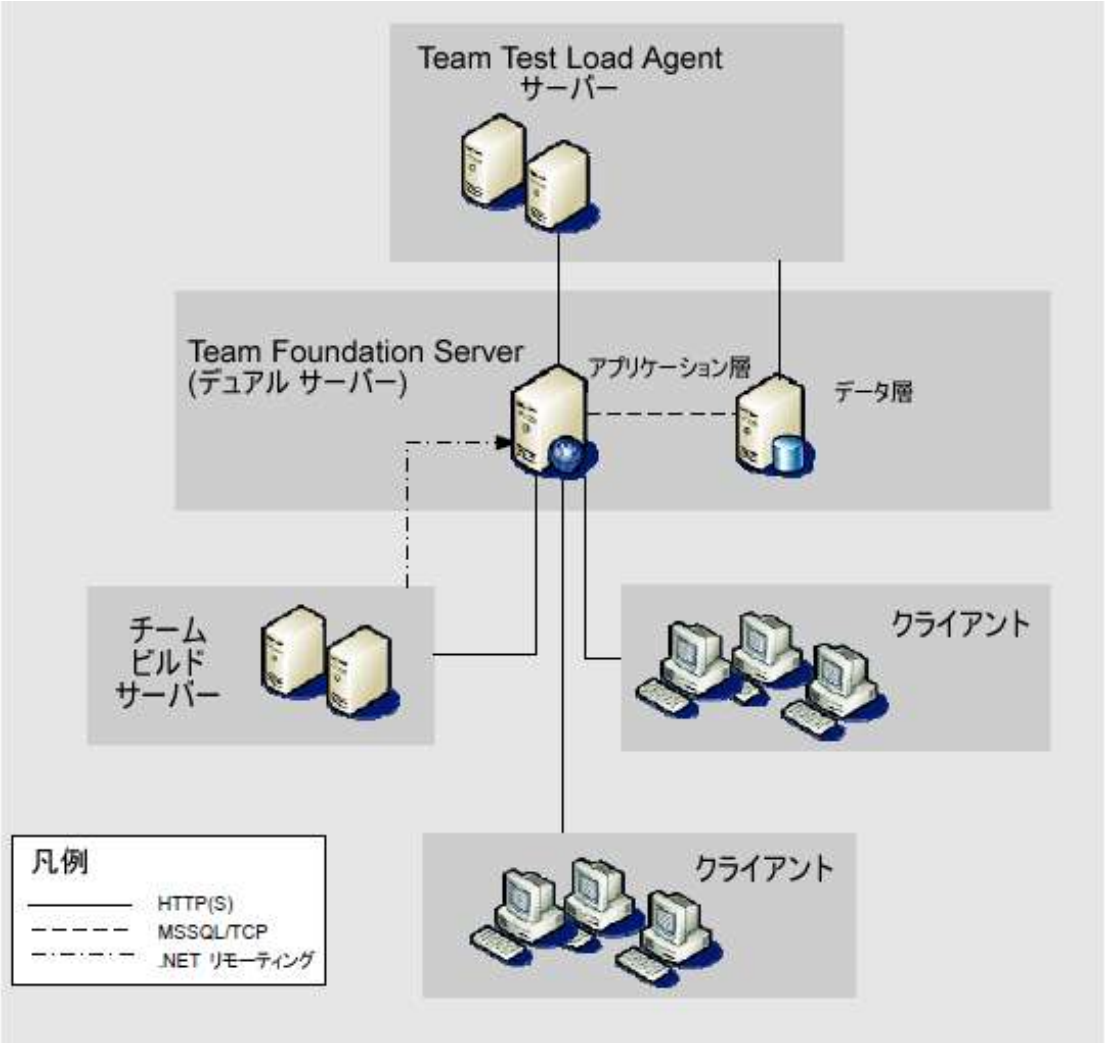


図 *16.5* 中規模の *Team Foundation Server* のトポロジ

図 16.5 は、1 つのテスト リグ、および別のノードに配置されたいくつかのビルド サーバーも表しています。クライアント ノードはサーバーと同じドメイン内にあるか、またはサーバーと信頼関係のあるドメイン内のいずれかにあります。この程度の複雑なトポロジは、ユーザーが 400 ～ 2000 人程度のより大規模な開発チームを対象としています。

複雑なトポロジ

図 16.6 に示されている複雑なトポロジは、中規模のトポロジに似ています。ただし、このトポロジでは、フェールオーバーのコンポーネントが、アプリケーション層のスタンバイ サーバー、および SQL クラスタリング テクノロジを備えたデータ層に追加されています。

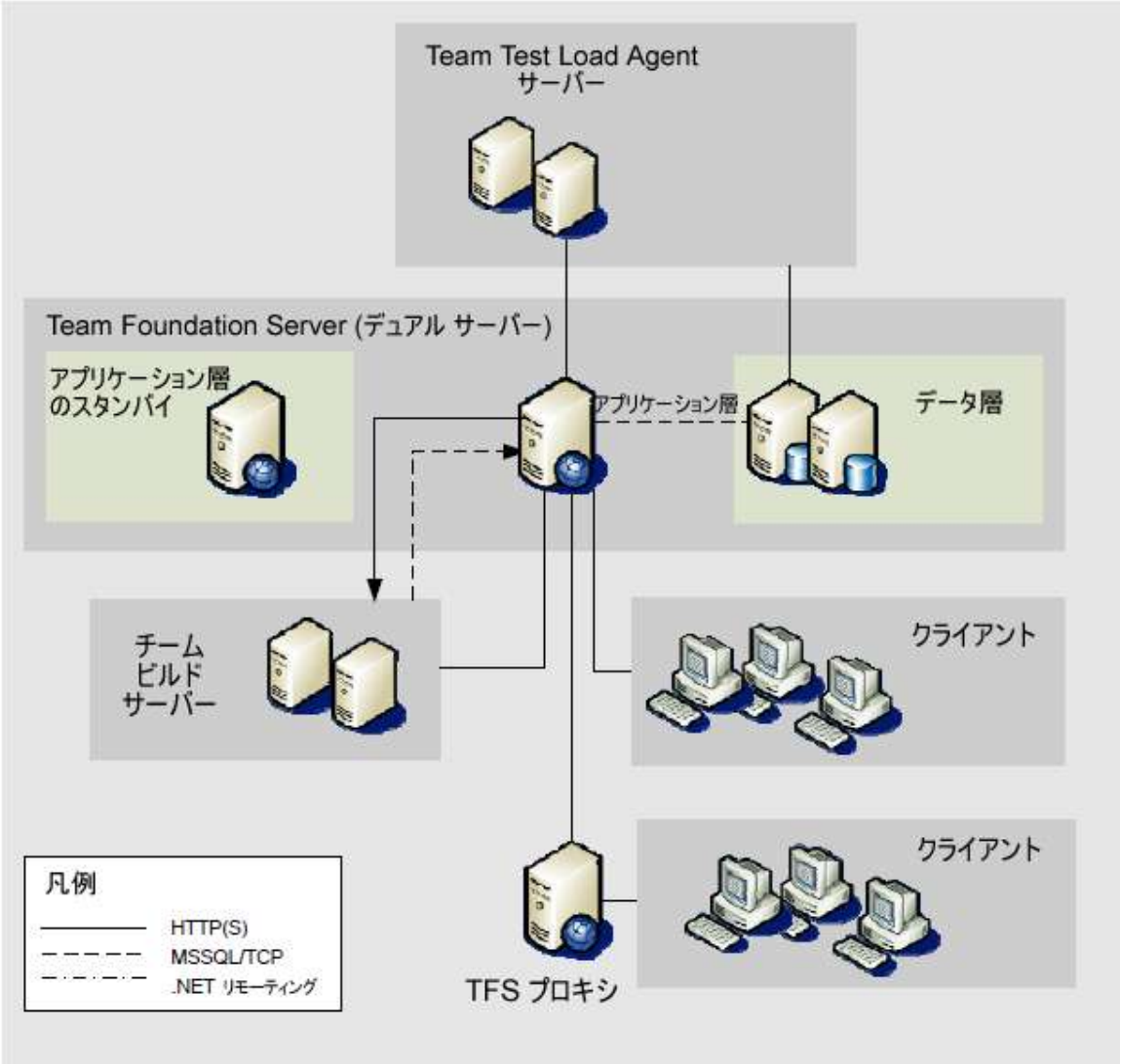


図 *16.6* 複雑な *Team Foundation Server* のトポロジ

図 16.6 は、制約のある帯域幅接続を使用している、地理的に離れた子ドメインも表しています。これらのクライアントは、TFS プロキシ サーバーを使用して、ソース管理に対するアクセス時間を改善しています。

その他の考慮事項

TFS を配置する場合には、次のことに注意してください。

- 既に SharePoint をセットアップしており、これを使用して Team Foundation Server の SharePoint サイトをホストしようとする場合は、TFS SharePoint サイトをもう 1 つのサーバーへ移行することができます。
  詳細については、
  http://blogs.msdn.com/bharry/archive/2006/10/30/moving-your-tfs-sharepoint-site.aspx
  を参照してください。
- 大規模なチームでは、OLAP エンジンおよびキューブを 3 つ目のマシンへ移行するとメリットがあることがわかっています。SQL クラスタリングをデータ層でセットアップし、1 つのノードで SQL を備えたアクティブ/アクティブ構成を持ち、他方のノードで OLAP を持ち、互いにこの 2 つのフェールオーバーとして動作するように構成できます。詳細については、
  http://msdn2.microsoft.com/ja-jp/library/aa721760(vs.80).aspx および
  http://msdn2.microsoft.com/ja-jp/library/ms252505(VS.80).aspx を参照してください。

Team Foundation Server の拡張性およびバックアップ方針

Team Foundation Server のインストレーションおよび配置の一環として、サーバーのバックアップおよびフェールオーバーをどのように管理するかを決める必要があります。選択するバックアップおよびフェールオーバーの方針は、インストレーションおよび機能の規模、および組織で使用できるリソースによって決まります。データ層は SQL Server 2005 の上に構築されているため、採用する方針は、SQL Server のバックアップで利用しているアプローチがベースとなります。

現在、SQL Server 2005 のインストレーションをミラーリングまたはクラスタリングしている場合は、TFS のデータ層に対して同じアプローチを使用できます。また、アプリケーション層のサーバーの障害をどのように管理するかも決める必要があります。アプリケーション層のフェールオーバーをサポートしたい場合は、バックアップのアプリケーション層サーバーを正しく配置し、サーバーに対して迅速にフェールオーバーできるようにする必要があります。

企業に対して適切なインストレーションおよびバックアップ/リカバリ方針を選択する

TFS をインストールする場合には、インストレーションおよびバックアップ/リカバリの方針について、いくつか選択する必要があります。インストレーションの方針を決定する場合には、次の内容を考慮します。

- チームの規模
- プロジェクトの数
- プロジェクトの規模
- チームの場所
- フェールオーバーの必要性
- バックアップの必要性

Team Foundation Server で推奨されるハードウェア

通常、プロジェクト数が少なく、比較的規模の小さいチームは、1 つの層のインストレーションで運用できますが、規模の大きいチームでは、2 つの層およびより高速なハードウェアが必要です。選択するインストレーションが 1 つの層か、または 2 つの層かによって、バックアップとフェールオーバーのメカニズムが影響を受けます。

1 つの層または 2 つの層のどちらのインストレーションにするかを決定し、チームのサポートに必要なハードウェアを特定する場合には、表 16.1 が役に立ちます。

構成	層	CPU	ハード ディスク ドライブ	メモリ
20 人未満のユーザーで 1 つのサーバー	アプリケーション層およびデータ層のサーバー	シングル プロセッサ、2.2 GHz	8 GB	1 GB
20 ～ 100 人のユーザーで 1 つのサーバー	アプリケーション層およびデータ層のサーバー	デュアル プロセッサ、2.2 GHz	30 GB	2 GB

100 ～ 250 人のユーザーで 2 つのサーバー	アプリケーション層のサーバー	シングル プロセッサ、2.2 GHz	20 GB	1 GB
	データ層のサーバー	デュアル プロセッサ、2.2 GHz	80 GB	2 GB
250 ～ 2000 人のユーザーで 2 つのサーバー	アプリケーション層のサーバー	デュアル プロセッサ、2.8 GHz	40 GB	4 GB
	データ層のサーバー	クワッド (4) プロセッサ、2.7 GHz	ダイレクト アタッチ ストレージ、14,000 ～ 15,000 RPM RAIDO 0 スピンドル	16 GB

表 16.1 *TFS* の配置に必要なハードウェア

バックアップおよびフェールオーバーの方針

バックアップとフェールオーバーの方針を検討する場合には、サーバーが使用できなくなったときのチームの生産性に与える影響を考慮する必要があります。

バックアップ

TFS インストレーションの配置の一環として、バックアップの方針を計画する必要があります。次の内容を考慮します。

- バックアップの頻度
- 完全なバックアップと差分バックアップの頻度
- バックアップに必要なストレージ (オンサイト、オフサイトのバックアップなど)

SQL Server 2005 のデータベースで使用するものと同じ標準のバックアップ プラクティスを使用できます。

バックアップを使用して、次の 3 つのシナリオで TFS をリストアできます。

- データのみのリカバリ
- シングルサーバー配置の完全リカバリ
- デュアルサーバー配置の完全リカバリ

データのみのリカバリは、データ層が破損した場合に使用します。バックアップ データとログを使用して、データベースを完全に回復することができます。

サーバーで障害が発生した場合は、サーバー リカバリを使用します。この場合には、データベース全体を 2 台目のコンピュータへリストアできます。

アプリケーション層のスタンバイ サーバー

アプリケーション層のサーバーにはバックアップするデータはありませんが、サーバーで障害が発生する可能性はあります。この障害の損失を緩和するためには、ウォーム スタンバイ サーバーを使用してアプリケーション層でフェールオーバーできるよう考慮する必要があります。

フェールオーバー

TFS にフェールオーバーのソリューションを提供するかどうか検討する場合は、TFS が使用できない場合に企業の生産性が失われることによる損失に対して、フェールオーバー サーバーを提供するのに必要なハードウェアの費用を考慮する必要があります。

フェールオーバーにより、インストレーションはさらに複雑になり、メンテナンスの費用も増えます。方針を決定する場合には、費用の検討事項としてメンテナンス費用も計算に入れる必要があります。

クラスタリングは、リソースおよびメンテナンスの点で費用が高くなります。組織で、クラスタ サーバーを管理するためのリソースを既に用意している場合は、クラスタリングが推奨されます。

ミラーリングは費用がかかりますが、クラスタリングよりは安価です。ミラーリングでは、メンテナンスの場合にプリンシパル サーバーをオフラインにできるというメリットがあります。組織で、2 つ目のデータ層サーバーをセットアップおよび保持できる場合は、ミラーリングを検討するとよいでしょう。

データ層

データ層サーバーのクラスタリング

必要なリソースが組織にある場合は、クラスタ内に専用のサーバーをセットアップすることを検討する必要があります。クラスタは、データ層に対して継続したアクセスを提供します。ただし、クラスタに対するハードウェアの要件は厳密です。クラスタをセットアップおよび保持するためのリソースに関するコストは高くなります。

クラスタリングの場合、TFS は、パッシブ ノード、アクティブ ノード、およびシングル クォーラム デバイス サーバーの構成をサポートします。データ層がパッシブ ノードへフェールオーバーすると、このノードは、クォーラムおよびデータ層のオーナーシップをとります。

クラスタ内で TFS をインストールする前に、インストレーションに対するクラスタを準備する必要があります。SQL Server 2005 のクラスタリングに関する詳細については、http://www.microsoft.com/downloads/details.aspx?familyid=818234dc-a17b-4f09-b282-c6830fead499&displaylang=en の「SQL Server 2005 Failover Clustering White Paper」を参照してください。

データ層サーバーのミラーリング

サーバーのミラーリングによって、あるサーバーのデータと、別のサーバーのデータのコピーが同期されます。データ層のサーバーはプライマリ サーバーで、ミラーされたデータを持つサーバーはバック

アップまたはミラーリング サーバーとなります。データ層のサーバーで障害が発生した場合は、ミラー サーバーへ手動で切り替えることができます。

ミラー サーバーを使用すると、メンテナンスおよび修理のときにメイン サーバーをオフラインにできる、メインのデータ層サーバーで障害が発生したときにすばやいリカバリのメカニズムを提供できる、ということがあります。

ミラーリングは同期的または非同期的にすることができます。ロール スイッチングと呼ばれる動きの中で、メイン サーバーからミラー サーバーへ、サーバーを切り替えることができます。ロール スイッチが発生すると、ミラーは、プリンシパル サーバーのロールを引き継ぎます。以前のプリンシパルがまだ有効な場合は、ミラーのロールを引き継ぐことができます。原則的に、ロールは切り替えたり、元に戻したりできます。

TFS では、自動のスイッチングはサポートされていません。代わりに、手動のスイッチングを使用する必要があります。

データ層に対して SQL のミラーリングを構成する方法

1. データベースおよびトランザクション ログの完全なバックアップを作成します。
2. Reporting Services の暗号化キーをバックアップします。
3. ミラー サーバーに SQL Server 2005 をインストールします。
4. データ層のデータを、ミラー サーバーへリストアします。
5. データ層のプリンシパル サーバー上の各データベースに対して、**データベース ミラーリング セキュリティ構成ウィザード**を実行して、ミラー サーバーを構成します。
6. ミラーリングを開始します。

ミラー サーバーのフェールオーバー

次の手順を実行して、ミラー サーバーを手動でフェールオーバーする必要があります。

1. TFS のアプリケーション層
    a. 新しいサーバーを使用できるように Report Service を再構成します。

   b. デフォルトの Web サイトを停止します。
   c. SharePoint の Web サイトを停止します。
   d. SharePoint タイマー サービスを停止します。
   e. TfsServerScheduler サービスを停止します。
   f. ReportServer アプリケーション プールを停止します。
   g. TFS App Pool アプリケーション プールを停止します。
2. ミラー データ層のサーバーで、正しいサービスアカウントが追加されていることを確認します。
3. プリンシパル サーバーからミラーリング サーバーへ、各データベースをフェールオーバーします。
4. 新しいサーバー上にデータ ウェアハウスを構築します。
5. ミラー層のサーバーを使用できるよう、次のようにしてアプリケーション層のサーバーを構成します。
   a. コマンド プロンプトから **TFSAdminUtil RenameDT MirrorDataTierServer** を実行します。
   b. IIS を再起動します。
   c. ミラー データ層のサーバーを参照するよう、Reporting Services の接続文字列を変更します。
   d. ミラー データ層のサーバーを使用するよう、SharePoint のサーバーを変更します。
   e. SharePoint タイマー サービスを開始します。
   f. TfsServerScheduler サービスを開始します。
   g. ReportServer アプリケーション プールを開始します。
   h. TFS App Pool アプリケーション プールを開始します。
   i. Reporting Services を開始します。
   j. StampWorkItemCache Web サービスを呼び出します。

アプリケーション層

アプリケーション層のフェールオーバー

プライマリ アプリケーション層のサーバーをセットアップした後で、アプリケーション層のウォームフェールオーバーができるように、ウォーム スタンバイ コンピュータを追加することができます。

スタンバイ ハードウェアおよびソフトウェア

スタンバイ サーバーは、プライマリ サーバーとまったく同じである必要はありませんが、アプリケーション層のハードウェア要件を満たしている必要があります。TFS のアプリケーション層のソフトウェアを、ウォーム スタンバイ サーバーへインストールします。

ユーザー アカウント、権限の変更、ソフトウェアの更新が同じであることも含めて、両方のサーバーが同じ構成であることを確認する必要があります。プライマリ コンピュータのすべての更新は、ウォーム スタンバイ サーバーにも必ず適用されます。

フェールオーバーでの問題を最小限にするには、プライマリ コンピュータとスタンバイ コンピュータの両方で同じホスト名を使用するよう、ネットワーク アダプタを構成する必要があります。このようにするには、多くの方法があります。

アプリケーション層サーバーのフェールオーバー

アプリケーション層のサーバーを手動でフェールオーバーします。プライマリ サーバーで障害が発生した場合は、ウォーム スタンバイ サーバーを手動で有効化するために、次の手順を実行する必要があります。スタンバイ サーバーで **ActivateAT** コマンドを渡して **TFSAdminUtil** ユーティリティを実行し、プライマリ サーバーのフェールオーバーをサポートします。

サーバーをウォーム フェールオーバーする方法:

1. アプリケーション層のスタンバイ サーバーが起動したら、オリジナル サーバーをオフラインにします。
2. スタンバイ サーバーで次のようにします。

a.管理者としてログします。

b.**ActivateAT** を渡して **TFSAdminUtil** を実行します。

c.スタンバイ サーバーで Web サービスを開始します。

このコマンドにより、次の処理が行われます。

- ウォーム スタンバイ サーバーの名前が TFS の統合データベースに登録されます。
- ウォーム スタンバイ アプリケーション層のサーバーを、有効なデータ層のサーバーへ接続します。

- 正しいアプリケーション層のサーバーが、正しいデータ層のサーバーへ接続されていることを確認します。

アプリケーション層のフェールオーバー サーバーを有効にする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252501(VS.80).aspx の「方法 ： フェールオーバーアプリケーション層サーバーをアクティブにする」を参照してください。

まとめ

TFS のアーキテクチャには、アプリケーション層、データ層、およびクライアント層の 3 つの層があります。サーバーをインストールする場合には、アプリケーション層とデータ層を同じ物理サーバーへインストールするか、または別のサーバーへインストールするか選択できます。TFS の配置の選択は、主に、サポートしたいユーザーの数によって決まります。チームのニーズをサポートするトポロジを選択したら、どのレベルのバックアップおよびフェールオーバーのサポートが必要か決定します。

データ層では、他の SQL Server 2005 のバックアップに対して組織で使用しているものと、同じバックアップ メカニズムを使用できます。フェールオーバーのサポートに対して、データ層サーバーをミラーリングするか、またはクラスタ化するかを選択できます。

アプリケーション層では、自動のフェールオーバーをサポートしていません。迅速なフェールオーバーをサポートする必要がある場合は、手動でフェールオーバーできる、ウォーム フェールオーバー サーバーを提供することができます。

参考資料

- TFS のインストールの詳細については、http://go.microsoft.com/fwlink/?linkid=40042 の「*Visual Studio 2005 Team Foundation Installation Guide*」を参照してください。
- TFS の拡張性の制限の詳細については、http://blogs.msdn.com/bharry/archive/2006/01/04/509314.aspx の「Team Foundation Server Capacity Planning」を参照してください。

- OLAP キューブおよび分析エンジンを別のサーバーへ移行する詳細については、http://msdn2.microsoft.com/ja-jp/library/aa721760(vs.80).aspx の「方法 : データ ウェアハウス SQL Server Analysis Services データベースを別のサーバーに移動する」を参照してください。
- SQL Server 2005 のクラスタリングに関する詳細については、http://www.microsoft.com/downloads/details.aspx?familyid=818234dc-a17b-4f09-b282-c6830fead499&displaylang=en の「SQL Server 2005 Failover Clustering White Paper」を参照してください。
- SQL Server のフェールオーバー クラスタ作成の詳細については、http://uat.technet.microsoft.com/ja-jp/library/ms179530(SQL.90).aspx の「新しい SQL Server 2005 フェールオーバー クラスタを作成する方法 (セットアップ)」を参照してください。
- データ層に対する SQL Server クラスタのセットアップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252505(VS.80).aspx の「データ層サーバーのクラスタ化」を参照してください。
- TFS SharePoint サイトを別のサーバーへ移行する方法の詳細については、http://blogs.msdn.com/bharry/archive/2006/10/30/moving-your-tfs-sharepoint-site.aspx の「Moving your TFS SharePoint site」を参照してください。
- Team Foundation Server の拡張性の詳細については、http://blogs.msdn.com/bharry/archive/2005/12/09/502190.aspx の Brian Harry のブログを参照してください。
- 障害リカバリの計画の詳細については、http://www.microsoft.com/technet/itshowcase/content/vs05teamsystemnote.mspx の「Visual Studio Team System User Education」を参照してください。
- Team Foundation Server のバックアップの障害およびリカバリの詳細については、http://msdn2.microsoft.com/en-gb/library/ms253159(VS.80).aspx の「http://msdn2.microsoft.com/ja-jp/library/ms253159(VS.80).aspx」を参照してください。
- データ層サーバーのクラスタリングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252505(VS.80).aspx の「データ層サーバーのクラスタ化」を参照してください。
- Team Foundation Server のデータ層のミラーリングに関する詳細については、

http://msdn2.microsoft.com/ja-jp/library/aa980644(VS.80).aspx の「Team Foundation データ層サーバーのミラーリング」を参照してください。

- データ層に対する SQL Server ミラーリングの構成の詳細については、http://msdn2.microsoft.com/ja-jp/library/aa980629(VS.80).aspx の「方法 ：Team Foundation データ層サーバーに対して SQL Server ミラーリングを構成する」を参照してください。
- データ層のフェールオーバーの詳細については、http://msdn2.microsoft.com/ja-jp/library/aa980627(VS.80).aspx の「方法 ： ミラー化されたデータ層サーバーにフェールオーバーする」を参照してください。
- プリンシパル サーバーが使用できなくなった場合のデータ層のフェールオーバーの詳細については、http://msdn2.microsoft.com/ja-jp/library/aa980528(VS.80).aspx の「方法 ： プリンシパル データ層サーバーを使用できない場合にミラー化されたデータ層サーバーにフェールオーバーする」を参照してください。
- アプリケーション層のフェールオーバー サーバーを有効にする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252501(VS.80).aspx の「方法 ： フェールオーバー アプリケーション層サーバーをアクティブにする」を参照してください。
- アプリケーション層のフェールオーバー サーバーの有効化の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252486(VS.80).aspx の「フェールオーバー アプリケーション層サーバーのアクティブ化」を参照してください。

第 17 章 Team Foundation Server に対するインターネット アクセスの提供

目的

- リモート アクセスの主なシナリオ、およびそれらのシナリオをいつ適用するか、について学習します。
- インターネットを介して Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) へ

リモート アクセスを提供します。

- Team Foundation Server のプロキシを使用してリモート アクセスのパフォーマンスを改善します。

概要

この章では、インターネットを介して TFS へリモート アクセスを提供する方法について説明します。リモート アクセスを提供するには、次の 3 つの方法のいずれかを選択できます。

- バーチャル プライベート ネットワーク (VPN) を介して TFS にアクセスを提供できます。
- Microsoft Internet Security and Acceleration (ISA) Server などのリバース プロキシを通じて TFS へアクセスを提供できます。
- エクストラネット上で TFS サーバーをホストできます。

TFS の Service Pack 1 (SP1) 以前の バージョンでは、VPN アクセスのみサポートしています。TFS SP1 は、基本認証に対するサポートを追加しています。これにより、VPN の他に、エクストラネットおよびリバース プロキシのソリューションを使用できるようになります。

この章の参照の仕方

この章を使用して、TFS サーバーを、インターネットを介したリモート アクセスに対してセットアップします。この章を最大限に利用するには、次のようにします。

- **シナリオの一覧を使用します。**サーバーに対して外部のインターネット アクセスを提供するために、どのアプローチを採用すべきかを迅速に判断するには、シナリオ一覧を調べます。
- **参照先のウォークスルーを使用します。**証明書および Secure Sockets Layer (SSL) アクセスをインストールおよび構成して、基本認証およびダイジェスト認証とともに使用するための、手順を追ったガイダンスを参照するには、この章のウォークスルーを使用します。
- **「リモート アクセスのパフォーマンスの改善」のセクションを使用します。**インターネットを介して送信する必要があるトラフィック量を減少させる方法を特定するには、「リモート アクセスのパフォーマンスの改善」のセクションを読んでください。

主な方針

TFS サーバーにリモート アクセスを提供するには、次のソリューション方針があります。

- **VPN 接続を使用する。**TFS を内部ネットワークの中に配置し、外部のユーザーは VPN を介して TFS へアクセスします。内部ユーザーは、TFS に直接アクセスします。
- **リバース プロキシを介して TFS をパブリッシュする。**TFS を内部ネットワークの中に配置し、ISA Server などの 1 つ以上のリバース プロキシ マシンが、クライアントの要求をインターネットから TFS へ提供します。
- **TFS をエクストラネットへ配置する (「ホステッド シナリオ」)。**外部のクライアントのみ TFS にアクセスします。TFS はエクストラネット上のファイアウォールの外部に配置します。

一般的なシナリオ

- **リモート オフィス。**VPN アクセスを伴うリモート ユーザーをサポートしている場合は、VPN ソリューションを使用します。これは、汎用的なセキュリティを提供する、TFS のすべての機能に対してリモート アクセスできるようにする、TFS プロキシを使用してパフォーマンスを改善する、といったことを実現するための最も簡単なソリューションです。
- **海外のチーム。**VPN アクセスを伴わない、またはドメインに対してアクセスしないリモート ユーザーをサポートしている場合は、リバース プロキシのシナリオを使用します。このソリューションは、セットアップするのが少し難しくなりますが、リモート ユーザーは、VPN を必要とせずに、内部に配置されている TFS にアクセスできるようになります。
- **ホステッド コミュニティ。**コミュニティ開発サイトなど、専用に使用する TFS インストレーションを使用する一部のリモート ユーザーをサポートしている場合は、エクストラネットのシナリオを使用します。このソリューションは、リモート ユーザーと内部ネットワーク リソースの間を最大に分離します。

VPN 接続の使用

図 17.1 は、VPN を介して TFS を公開するためのアーキテクチャを示しています。

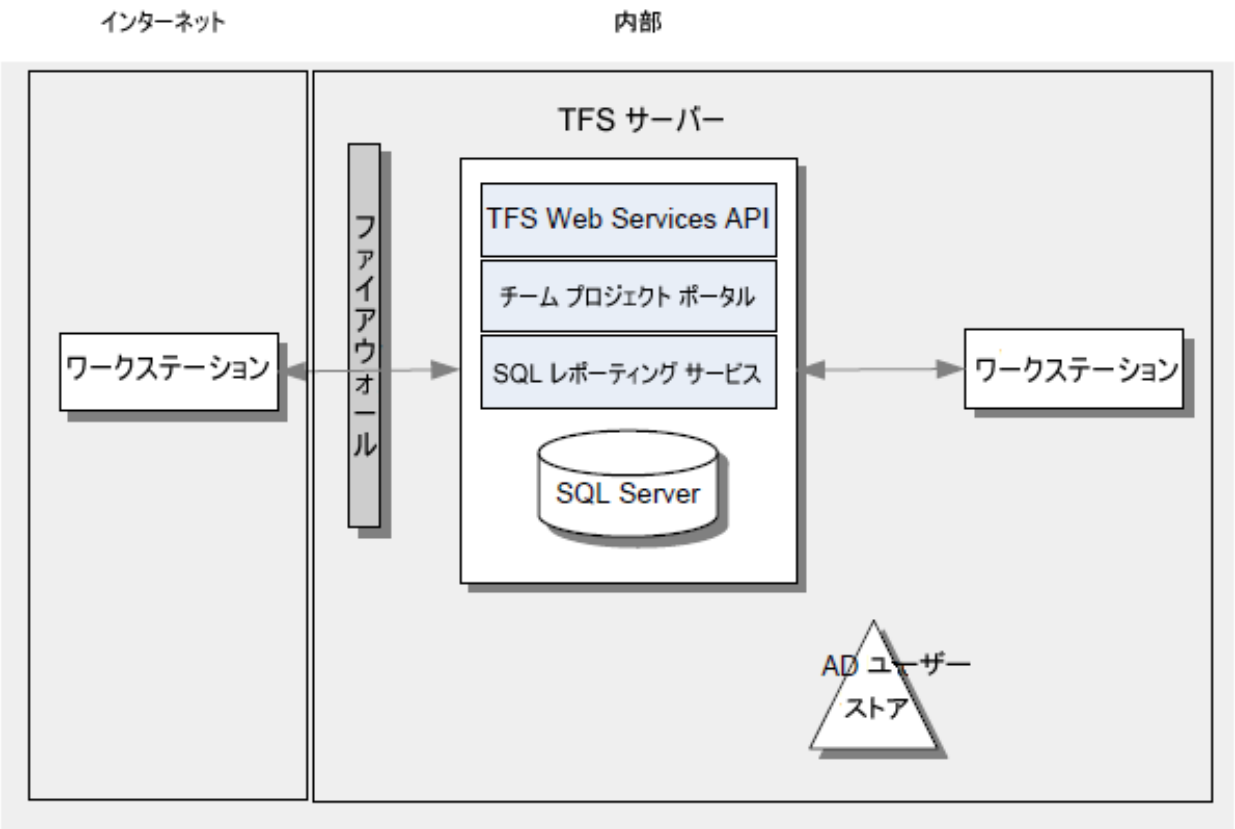


図 *17.1* *VPN* を介した *TFS* のアーキテクチャ

このアプローチでは、リモート開発チームは、内部ネットワークの TFS に対して直接 VPN 接続を使用します。TFS に SP1 が搭載されていない場合、または統合 Windows 認証が必要な場合は、VPN 接続を使用する方法しかありません。TFS は、VPN アクセスなど、帯域幅が低い状態で機能するように設計されており、このような状態でも、許容できる程度のパフォーマンスを提供します。

メリット

- TFS プロキシも含めて、TFS のすべての機能が動作します。
- このアプローチは、統合 Windows 認証の使用をサポートしており、エンタープライズの既存のインフラストラクチャを利用できます。
- VPN が既にセットアップされている場合は、これを採用するのが最も簡単なソリューションです。

デメリット

- インフラストラクチャに対して VPN ソリューションをセットアップできない、またはリモートユーザーが使用できない可能性があります。

VPN の作成に関する詳細については、http://support.microsoft.com/kb/324747 を参照してください。

リバース プロキシを介した TFS のパブリッシュ

このアプローチでは、内部のネットワークに TFS サーバーをインストールし、ISA Server Web パブリッシング機能を使用して外部のネットワークに TFS を公開します。リモート ユーザーは、SSL を介して TFS にアクセスし、基本認証を使用します。TFS SP1 がインストールされていない場合は、このオプションは機能しません。

境界上にドメイン コントローラがないネットワーク

図 17.2 は、内部ネットワーク上にドメイン コントローラを持つ ISA を介して TFS を公開するためのアーキテクチャを示しています。

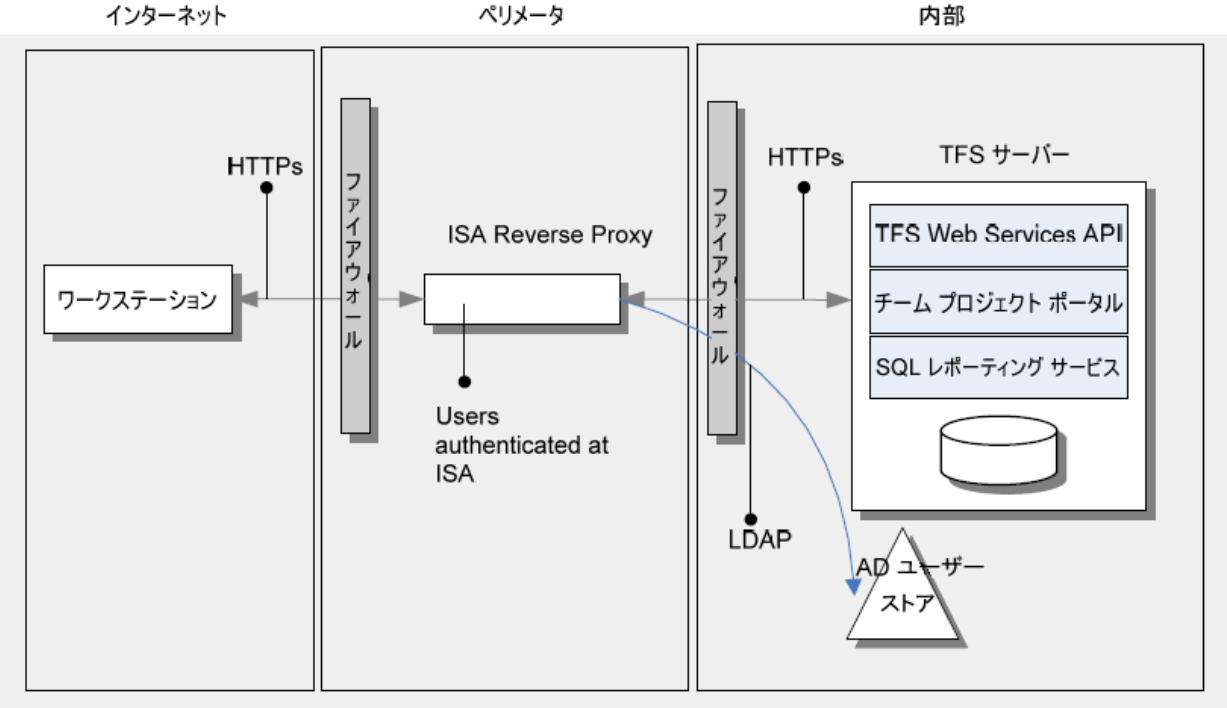


図 *17.2*

ISA、内部ネットワーク上のドメイン コントローラを介した *TFS* のアーキテクチャ

境界ネットワークにドメイン コントローラがない場合は、ファイアウォール上でポートをオープンし、ISA Server から内部のドメイン コントローラへ、Lightweight Directory Access Protocol (LDAP) で接続できるようにします。具体的に言うと、外部の TFS ユーザーを認証できるようにします。

境界上にドメイン コントローラがあるネットワーク

図 17.3 は、境界ネットワーク上にドメイン コントローラを持つ ISA を介して TFS を公開するためのアーキテクチャを示しています。

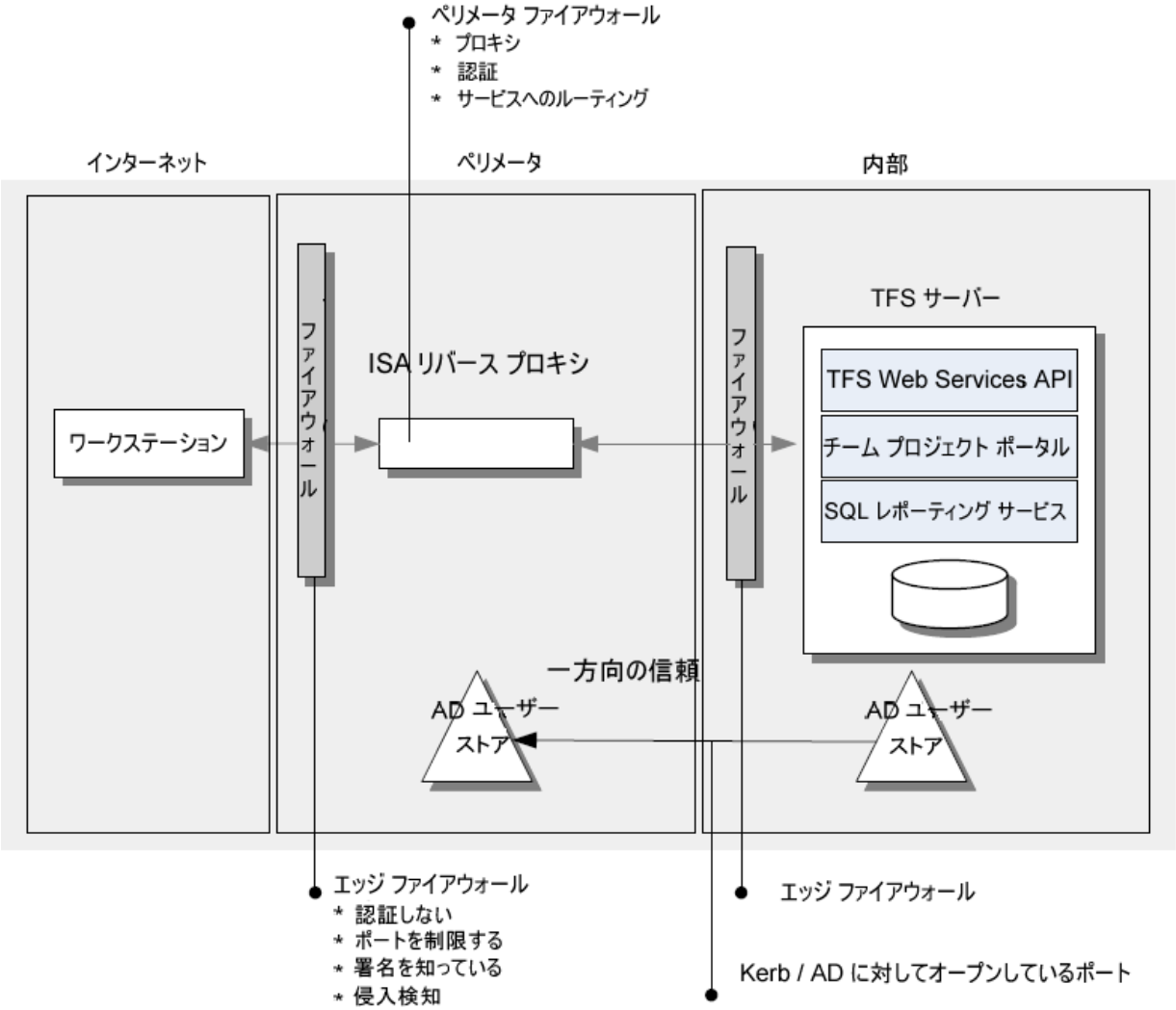


図 *17.3* *ISA*、境界ネットワーク上のドメイン コントローラを介した *TFS* のアーキテクチャ

境界ネットワーク上にドメイン コントローラがある場合は、リモート ユーザーは、境界のドメイン コントローラに対して直接認証することができます。内部および境界のドメイン コントローラ間の一方向の信頼により、外部ユーザーは TFS サーバーにアクセスできます。内部ユーザーは、統合 Windows 認証を使用して、TFS サーバーへ直接アクセスします。

メリット

- ISA Server などのリバース プロキシは、ユーザーを認証し、トラフィックを検査します。
- リモート ユーザーは、ドメインにアクセスする必要はありません。
- リモート ユーザーは、VPN アクセスする必要はありません。

デメリット

- リモートの場所で TFS プロキシを使用できません。
- リモート ユーザーは、TFS グループへユーザーを追加できません。
- リモート ユーザーは Microsoft Active Directory® グループを、ソース管理のフォルダに追加できません。
- リモート ユーザーは、ビルドをリモートで開始または管理できません。
- リモート ユーザーは新しいチーム プロジェクトを作成できません。
- リモート ユーザーはテスト結果を TFS へ発行できません。

注意:基本認証を使用する場合は、常に SSL を使用してください。基本認証は、クリア テキストで資格情報を送信します。この情報をプロテクトするには SSL を使用します。

TFS をエクストラネットへ配置する (「ホステッド シナリオ」)

図 17.4 は、TFS をエクストラネットへホスティングするためのアーキテクチャを示しています。

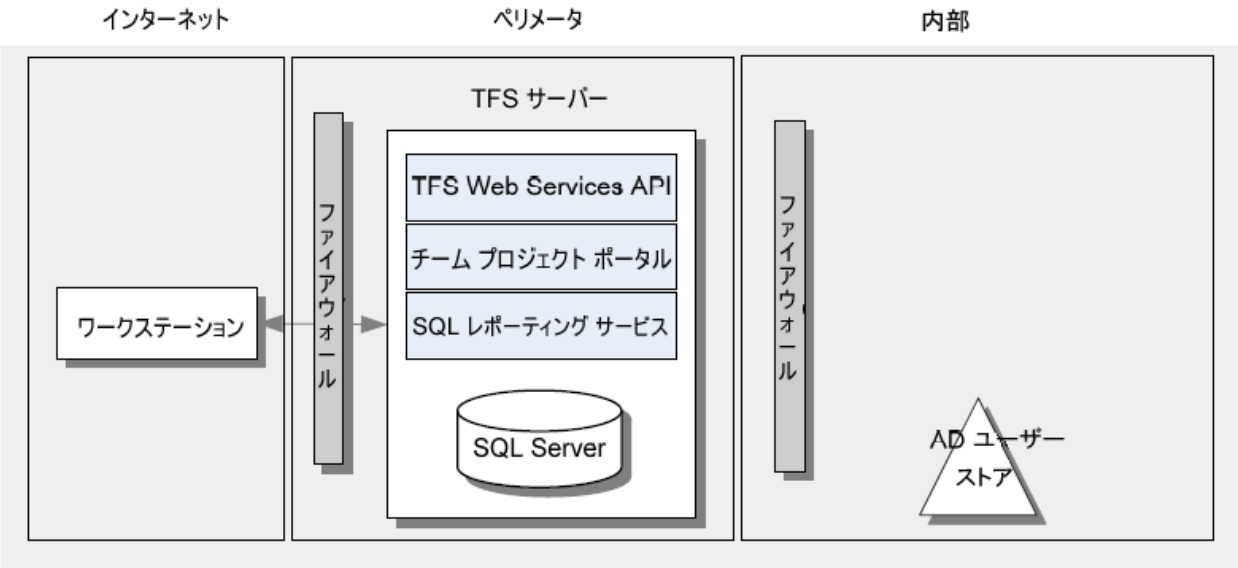


図 *17.4* エクストラネットでホストされている *TFS* のアーキテクチャ

このアプローチでは、完全な TFS インフラストラクチャ (アプリケーション層とデータ層の両方) を、内部のイントラネットとは離して、境界ネットワークの内部にインストールします。TFS に対するすべての接続は、(外部ユーザーの接続でも、内部ユーザーの接続でも) インターネットを介します。TFS は、ドメイン コントローラ (DC) あり、またはなし、のいずれの場合も動作します。境界ネットワークに、DC に対するアクセス機能がない場合は、TFS の Active Directory サービス機能は動作しません。たとえば、ユーザーを TFS グループに追加する、Active Directory グループをソース管理のフォルダに追加する、などの処理は DC なしではできません。TFS SP1 がインストールされていない場合は、このオプションは機能しません。

メリット

- TFS ユーザーは、内部ネットワークから明確に分離されます。
- リモート ユーザーは、ドメインにアクセスする必要はありません。

デメリット

- リモートの場所で TFS プロキシを使用できません。
- リモート ユーザーはビルドを開始または管理できません。
- リモート ユーザーは新しいチーム プロジェクトを作成できません。

- リモート ユーザーはテスト結果を TFS へ発行できません。
- 内部ユーザーは、外部ユーザーと同じように、SSL を介して、エクストラネットの TFS へ接続する必要があります。

注意:基本認証を使用する場合は、常に SSL を使用してください。基本認証は、クリア テキストで資格情報を送信します。この情報をプロテクトするには SSL を使用します。

基本認証 / SSL

TFS SP1 を使用しており、エクストラネットまたはリバース プロキシのシナリオをサポートしたい場合は、TFS のアプリケーション層で IIS を構成し、SSL を介して基本認証を使用できるようにします。基本認証では、プロテクトされていない Base64 のエンコード形式を使用し、インターネットを介してログオン資格情報が渡されます。クライアントの資格情報を守るには、SSL を使用している Secure HTTP (HTTPS) 接続を介す場合のみ基本認証を使用します。

Internet Server API (ISAPI) フィルタを使用すると、リモート クライアントは、SSL 上の基本認証を使用して接続しますが、一方でローカル クライアントは、統合 Windows 認証を使用して接続します。ISAPI フィルタは、"外部/インターネット" として構成されたクライアントを監視し、401 応答で NTLM 認証を外すことで、これらクライアントが、基本認証などの他の認証方法を使用するように強制します。

詳細情報

- リモート接続に対して基本認証および HTTPS を必要とするように TFS サーバーを構成する方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/aa833873(VS.80).aspx の「チュートリアル：HTTPS と SSL (Secure Socket Layer) の使用を必須とする Team Foundation Server のセットアップ」を参照してください。
- ISAPI フィルタのセットアップの詳細については、http://msdn2.microsoft.com/ja-jp/library/aa833872(VS.80).aspx の「チュートリアル：Secure Socket Layer (SSL) および ISAPI フィルタを使用する Team Foundation Server のセットアップ」を参照してください。

- TFS の ISAPI フィルタの詳細については、James Manning のブログ、http://blogs.msdn.com/jmanning/archive/2006/10/27/the-tfs-quot-extranet-quot-isapi-filter-mechanics.aspx の「The TFS extranet ISAPI filter mechanics」を参照してください。

Team Foundation Server プロキシ

図 17.5 は、Team Foundation Server プロキシを使用するアーキテクチャを示しています。

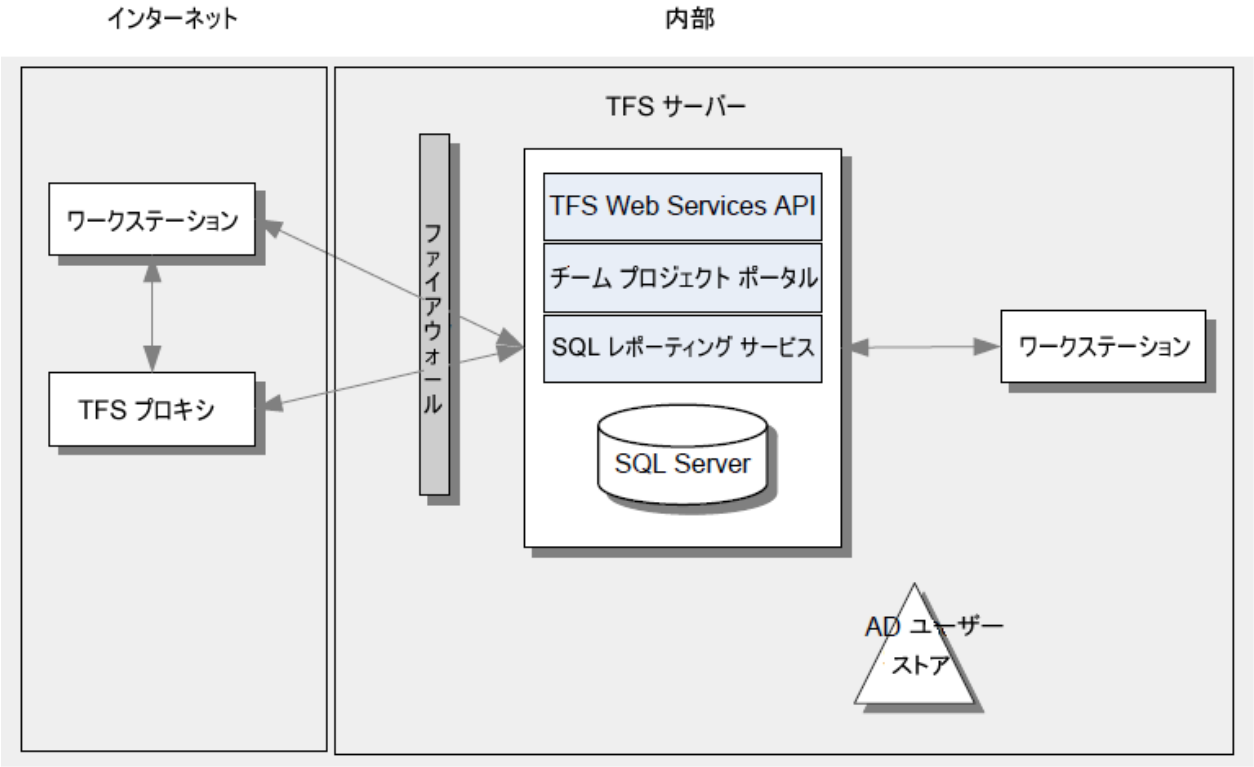


図 *17.5* *TFS* プロキシのアーキテクチャ

TFS プロキシはリモート アクセスには必須ではありませんが、ソース管理ファイルのオプショナルなキャッシュになります。VPN を通じて TFS に接続しているリモート オフィスに TFS プロキシをインストールすると、リモート チームのパフォーマンスを改善するうえで役に立ちます。これにより、リモート オフィスのプロキシ サーバー上でソース管理ファイルをキャッシュするため、パフォーマンスが改善されます。リモート クライアントが、ソース管理リポジトリ内のソース コードにアクセスしなければならない場合は常に、TFS プロキシにソースを要求します。プロキシは、(キャッシュを使用できる場合は) キャッシュからローカル バージョンを返します。ソースがキャッシュの中にない場合

は、プロキシは、リモート TFS サーバーにソースを要求します。これによりネットワークのトラフィックが軽減し、リモートの場所におけるソース管理の応答性が改善されます。

プロキシのパフォーマンスを改善するためのヒント

プロキシのパフォーマンスを改善するためには、次の推奨事項を検討します。

- 必ずキャッシングを使用できるようにし、キャッシュのパフォーマンスを監視します。プロキシがどのように実行されているかを確認するために、プロキシ サーバー上で定期的に、(デフォルトでインストールされている) パフォーマンス カウンタ、および (エラーと警告のための) イベント ログを監視します。TFS プロキシは、ProxyStatistics.xml という名前の Extensible Markup Language (XML) ファイルにキャッシュ パフォーマンスの統計を保存しています。これらの統計を保存する間隔は変更することができます。ProxyStatistics.xml ファイルは、プロキシのインストレーション ディレクトリの App_Data フォルダにあります。
- プロキシ サーバーへ最新のファイルを取り出すために、スケジュール タスクを定期的に実行します。このようにすると、ファイルの最新バージョンをプロキシ キャッシュ内で確実に使用できるようになり、これらのファイルに対する以降のクライアント要求がキャッシュでヒットします。
- 低い帯域幅 (3 Mbps 以下) で大きなファイルをダウンロードすることがわかっている場合は、Web.config で **executionTimeout** の構成を適切な値に設定します。デフォルトの値は 1 時間、つまり <httpRuntime executionTimeout="3600"/> です。
- 長時間プロキシが停止する場合には、無益な再接続をしないために、クライアント上のプロキシを無効にします。既定では、再接続は 5 分ごとに試行されます。
- 特定のワークスペースを参照されないようにして、不要なファイルの転送を避けるには、ワークスペースのクローキングを使用することを検討します。クローキングにより、特定のワークスペース フォルダがビューから見えなくなり、パフォーマンス帯域幅が改善されます。また、現在は必要ないフォルダとファイルが、ローカル ワークスペースにコピーされないようにすることで、ローカル ディスクのスペースが節約されます。ワークスペース内で既存のフォルダのマッピングをクロークすることができますが、クロークすることを意図して特別に、新しいフォルダ マッピングを作成する方が適切な方法です。

パフォーマンスを最適化するためのこれらのガイダンスの詳細については、このガイドの「ソース管理

のガイドライン」の「Distributed / Remote Development」を参照してください。

ミラー アカウント

TFS プロキシは、リモート オフィスにおいて VPN 接続でのみサポートされています。ただし、エクストラネットまたはリバース プロキシのシナリオを使用して、TFS プロキシが必要な小規模のリモート チームに TFS を配置している場合は、ミラー アカウントを使用してプロキシを使用することができます。

プロキシを有効にするために、対応するユーザー名とパスワードを持つワークグループ アカウントを、TFS、TFS プロキシ、およびリモート クライアントの各コンピュータ上で使用できます。この方法では、3 つの異なる場所で、すべてのユーザーに対して正確なユーザー名/パスワードの一致を保持する必要があり、管理の時間が増えるため、この方法を使用するのは小規模なリモート チームに限られます。

詳細情報

- TFS プロキシについて詳しく学習するには、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx の「Team Foundation Server のプロキシとソース管理」を参照してください。
- TFS プロキシの管理の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253156(VS.80).aspx の「Team Foundation Server Proxy に対するリモート接続の管理」を参照してください。
- TFS プロキシのトラブルシューティングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400681(VS.80).aspx の「Team Foundation Server Proxy のトラブルシューティング」を参照してください。

リモート ビルド サーバー

図 17.6 は、リモート ビルド サーバーを使用するアーキテクチャを示しています。

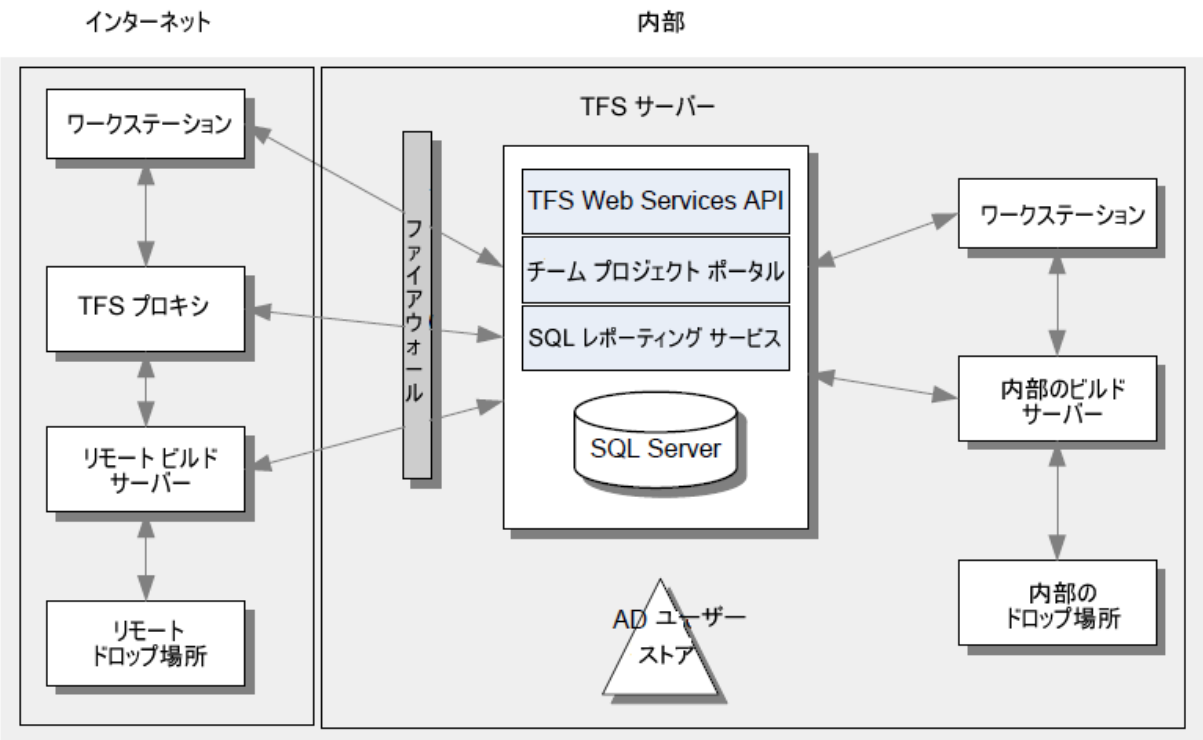


図 *17.6* リモート ビルド サーバーのアーキテクチャ

リモート チームのパフォーマンスをさらに改善するために、リモート オフィス内にビルド サーバーをセットアップすることができます。TFS プロキシがリモート オフィスにインストールされている場合は、TFS プロキシは他のすべてのソース管理クライアントと同様に機能し、各ビルドの前にプロキシからコードを取得します。リモート ビルド サーバーには、次のメリットがあります。

- リモート チームのチーム ビルドは、内部のビルド サーバーで負荷が増えるのではなく、チームのビルド サーバーに影響を与えます。
- リモート ビルドは、リモート チームへバイナリを提供しますが、ネットワークを介してバイナリを送信する必要はありません。

リモート ビルド サーバーは、内部のビルド サーバーで生成されたビルドに対する完全な置き換えとしては使用できません。内部のビルドと同じソース管理のバージョンから、リモート ビルドが生成されている場合でも、ビルドまたはソースの構成がサーバー間で異なっているため、異なる動作に見えることがあります。特に、リリースが近い場合は、ガイドラインとして、内部ビルドの重要なテストをベースにしてください。

注意:アプリケーション層は、ポート 9191 上のビルド サーバーと通信します。リモートのビルド サーバーがある場合は、アプリケーション層がこのポート上で接続できるようファイアウォールをセットアップします。

まとめ

SP1 を搭載していない TFS を使用している場合は、VPN を使用してリモート アクセスを提供します。TFS SP1 を使用している場合は、SSL を介して基本認証を使用し、TFS をエクストラネットに配置することも、リバース プロキシを通じてアクセスを提供することもできます。

特に VPN のシナリオでリモート アクセスのパフォーマンスを改善したい場合は、TFS プロキシをインストールおよび構成して、リモート ロケーション内でソース管理ファイルをキャッシュできます。

参考資料

- TFS のリモート開発のセットアップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms242919(VS.80).aspx の「チュートリアル : Team Foundation Server を使用したリモート オフィスの設定」を参照してください。
- SSL を使用した TFS のセットアップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms242875(VS.80).aspx の「チュートリアル : SSL (Secure Sockets Layer) を使用する Team Foundation Server のセットアップ」を参照してください。
- TFS の Basic および Digest 認証の詳細については、http://msdn2.microsoft.com/ja-jp/library/aa833874(VS.80).aspx の「Team Foundation Server の基本認証およびダイジェスト認証」を参照してください。
- TFS プロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx の「Team Foundation Server のプロキシとソース管理」を参照してください。
- TFS プロキシの管理の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253156(VS.80).aspx の「Team Foundation

Server Proxy に対するリモート接続の管理」を参照してください。

- TFS プロキシのトラブルシューティングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400681(VS.80).aspx の「Team Foundation Server Proxy のトラブルシューティング」を参照してください。
- TFS プロキシのパフォーマンス調査の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252455(VS.80).aspx を参照してください。

第 9 部

Visual Studio 2008 Team Foundation Server

第 9 部の内容

第 18 章 Visual Studio 2008 Team Foundation Server の新機能

Microsoft® Visual Studio® 2008 Team Foundation Server では、多くの新しい機能を導入しています。主な変更は次のとおりです。

- **管理、運用およびセットアップ。**インストレーションが簡潔になり、セットアップの時間が短縮されて、より多くの配置シナリオをサポートできるように改善されました。
- **ビルド。**ビルドには、継続的な統合、スケジュール ビルド、およびすぐに使用できるビルドのキューイング ソリューションが含まれています。UI からより多くの機能を使用できるようになったため、ビルド管理および拡張性が簡潔になりました。
- **バージョン管理。**バージョン管理では、オフラインの作業に対してより優れたサポートが提供され、パフォーマンスが改善されました。
- **作業項目の追跡。**作業項目の追跡では、改善されたクエリ ビルダが導入され、作業項目のアタッチメントのサポートが改善されました。

製品に関するこれらの変更点は、以降に一覧で簡潔にまとめており、その後に、変更点がこのガイドのガイダンスにどのように影響を与えるかについて、表で説明しています。この章を使用して、Microsoft Visual Studio Team Foundation Server のアップグレード計画に役立ててください。

管理、運用およびセットアップ

- **簡潔なインストレーション** – Visual Studio 2005 TFS に比べてインストレーションが簡単かつ迅速になりました。この改善は、別のデータ層のインストレーションを排除したこと、またドメイン アカウントの要件を排除したことによるものです。Team Foundation Server 2008 は、可能な限り、ビルトインのマシン アカウント (Network Service など) をサポートしています。
- **SharePoint 2007 のサポート** – SharePoint 2007 および SharePoint Services 3.0 のサポートを追加しています。Team Foundation Server 2008 は、Team Foundation のアプリケーション層のサーバーから、別のサーバー上の SharePoint をサポートします。
- **Windows Server 2008 のサポート** – Microsoft Windows Server™ の次期バージョン (Microsoft Windows Server 2008 など) や インターネット インフォメーション サービス (IIS) 7.0 をサポートします。
- **X.509 クライアント証明書 –** 認証セキュリティを改善するために、X.509 クライアント証明書の使用をサポートします。
- **大規模なグループの同期** – パフォーマンスおよび堅牢性を改善し、多数のユーザー (TFS のシングル インスタンスで約 30,000、またはそれ以上のユーザー) をサポートできるようにします。
- **SQL の名前付きインスタンスのサポート** – 複数の TFS インスタンス間、または他のアプリケーションとの間で SQL Server の共有を可能にします。これにより、TFS の複数のインスタンスで、SQL Server 2005 の同じインストレーションを使用することができます。
- **デフォルト以外のポートのサポート** – 構成能力が改善されており、別の Web サイトおよびポートをサポートします。

ビルド

- **継続的な統合ビルド** – 継続的な統合ビルドを開始させる正確なタイミングを設定することが可

能なビルド トリガの作成をサポートします。たとえば、チェックインのたびにビルドを開始するようトリガを設定したり、X 分ごとよりも多くビルドを開始しないようローリング ビルドをセットアップしたりできます。

- **ビルド キューイングのサポート** – ビルド キューイングおよびキュー管理をサポートします。複数のチェックインが複数のビルドに並ぶことがあるため、これは継続した統合にとって特に有用です。
- **スケジュール ビルド** – スケジュール ビルドをサポートします。組織の要件に基づいて、指定した時間にビルドを開始するよう構成できます。
- **ドロップ管理** – ドロップ管理をサポートします。これにより、どのタイミングでビルドを自動的に削除するかについてポリシーを設定できます。
- **ビルド プロパティの指定** – どのソースおよびソースのバージョンをビルドするか、ビルドの種類の新しいビルド プロパティに沿って指定することができます。ビルドをカスタマイズするために、さらに多くのプロパティが公開されています。また、ビルドをキューイングするときに、MSBuild のコマンドライン パラメータを渡すことができます。
- **ビルド ターゲットの拡張性** – ビルド ターゲットの拡張性を改善します。たとえば、Visual Studio の各ソリューション、またはプロジェクトがビルドされる前後で、ターゲットを簡単に実行できるようになりました。
- **ビルド管理** – Visual Studio からビルドを停止および削除することができます。
- **ビルドの構成** – ビルドの一環として、実行するテストを指定する機能が簡単になりました。
- **ビルド プロジェクト ファイルの場所の柔軟性** – MSBuild プロジェクト ファイル (およびそれに関連付けられている rsp ファイル) は、必ず **TeamBuildTypes** フォルダを使用するのではなく、バージョン管理の階層の任意の場所に格納することができます。
- **GUI テストのサポート** – ビルドの一環として、グラフィカル ユーザー インターフェイス (GUI) テストを実行できます。
- **チェックイン ポリシー** – 新しいチェックイン ポリシーをサポートします。これによりユーザーは、継続的な統合ビルドに失敗したときに、コードをチェックインできないようになります。
- **ビルド サーバーの管理** – 複数のビルド マシンを管理する機能を改善します。
- **ワークスペースのマッピング** – ビルドの定義を「実際の」ワークスペースに関連付けることができます。これは、複数のチーム プロジェクトのコードを取得できること、クライアントのマッピングを指定できることなどを意味します。作業フォルダのマッピングは、workspacemapping.xml

ではなく、GUI で管理されます。

バージョン管理

- **アノテーション** – アノテーション機能をサポートして、開発者がソース コード ファイルを調べて、コードの各セクションを最後に変更した人の詳細を行レベルで確認したりできます。
- **フォルダ Diff** – フォルダの比較をサポートします。異なるファイルを特定するために、フォルダの内容を再帰的に比較します。フォルダ Diff は、ローカル フォルダとローカル フォルダ、ローカル フォルダとサーバー フォルダ、およびサーバー フォルダとサーバー フォルダを比較することができます。
- **Destroy** – バージョン管理システムからファイルおよびフォルダを削除する機能を使用して、Destroy 機能をサポートします。いったん破壊されたファイルおよびフォルダは、リカバリできません。
- **チェックアウト時に最新バージョンを取得** – チェックアウトのときにファイルの最新バージョンをダウンロードするオプションが含まれています。
- **ワークスペース ワイルド カード マッピング** – クローク フォルダでフォルダまたはファイルをマッピングすること、およびワイルドカードのマッピングをすることが可能になり、サブフォルダをマッピングせずに、1 つのフォルダ内のすべてのファイルをマップできるようになります。
- **パフォーマンスの改善** – バージョン管理のさまざまなパフォーマンス強化がなされ、バージョン管理のパフォーマンスがすべての側面で改善されています。(10,000 ファイル未満の) 比較的小規模なサーバー/プロジェクトに対するメリットは中程度ですが、(特にファイルの数が数十万に達する場合の) 大規模なプロジェクトでは、メリットがかなりあります。
- **Team Foundation Server 2008 コマンド ライン ヘルプ** – *tf* ツールのコマンドライン ヘルプを表示する機能をサポートします。*tf help* を実行してヘルプを取得し、*tf command /help* を実行して個々のコマンドに対するヘルプを取得します。
- **オフラインの改善** – オフライン作業のエクスペリエンスを改善し、また、オンラインに戻すための*tfpt* オンライン機能を Visual Studio の 統合開発環境 (IDE) に統合しました。
- **チェックイン オーバーライド情報の取得** – ウェアハウスを上書きするチェックイン ポリシーの追加をサポートします。

作業項目の追跡

- **添付ファイルの改善 –** 添付ファイルの追加をサポートするためのドラッグアンドドロップをサポートし、添付ファイルの複数選択ができるようになります。
- **クエリ ビルダ –** クエリ ビルダの使い勝手が、次の点で改善されました。
    - ドロップダウンのフィルタリングが現在のプロジェクトに基づいている
    - MRU リストが改善された
    - カラムのドラッグアンドドロップ
    - Shift キー + マウスによるクリックに基づいたマルチカラムのソーティング

Visual Studio 2005 Team System との互換性の問題

Visual Studio 2008 Team Foundation Server のクライアントは Visual Studio 2005 Team Foundation Server で作業することが可能で、Visual Studio 2005 のクライアントは 次の互換性の問題を除いては、Visual Studio 2008 Team Foundation Server で作業することができます。

- **Visual Studio のアドイン** – クライアント側の Visual Studio のアドインは、再コンパイル (またはポリシーを変更) する必要があります。これは、Team Foundation Server オブジェクトモデル (TFSOM) アセンブリのバージョンが変わり、アドインを新しいアセンブリにバインドしなければならないためです。
- **チーム ビルド** – ビルド定義をリストする、ビルドを開始および停止する、ビルド レポートを調べる、などのほとんどのビルドの操作は、Visual Studio 2005 TFS および Visual Studio 2008 のクライアントとサーバーの組み合わせで機能します。この場合には、次のような既知の問題があります。

1. Visual Studio 2008 Team Foundation Server のインスタンスは、Visual Studio 2008 Team

Foundation Server のビルド サーバーとしか機能しません。

2. Visual Studio 2005 のクライアントが Visual Studio 2008 Team Foundation Server インスタンス上でビルドを開始するには、ビルドの定義を $/<TeamProject>/TeamBuildTypes/<name> に格納しておく必要があります。
3. Team Foundation Server 2008 のデータベース内にある tfsbuild.proj ファイルのプロパティに加えられた変更は、データベース内では更新されず、同期されません。
4. Team Foundation Server 2008 の継続的な統合機能で作業している場合は、Visual Studio 2005 のクライアントはビルドを開始できますが、ビルドをキューイングすることはできません。キュー内のビルド リストを参照するには、ビルド エージェントのリストなどを参照してください。
5. Visual Studio 2008 Team Foundation Server のクライアントを使用して、Visual Studio 2005 TFS サーバー上に新しいビルドの種類を作成することはできません。
6. Visual Studio 2008 Team Foundation Server のクライアントを使用している場合は、Visual Studio 2005 Team Foundation Server 上でビルドを開始するためのダイアログのパラメータは変更できません。

ガイダンスの影響

Visual Studio 2005 Team Foundation Server のガイダンス	Visual Studio 2008 Team Foundation Server のガイダンス
デュアルサーバー配置は、最大 2000 人のユーザーをサポートできる。	デュアルサーバー配置を使用して、最大 30,000 人のユーザーをサポートできる。
配置の一環として正しいドメイン アカウントが必要である。	ドメイン アカウントは不要になり、代わりに (Network Service のアカウントなど) ビルトインのマシン アカウントを使用できる。
カスタム ソリューションを使用して継続的な統合ビルドを作成する。	Visual Studio のビルド トリガを使用して、継続的な統合ビルドまたはローリング ビルドを作成および構成できる。
自動化されたテストをビルドの一環として使用して、ビルドの品質を測定でき	テスト一覧を生成して、ビルド ステップの一環として実行するテストの指定が簡単にできる。自動化されたビル

る。	ド テストの一環として GUI テストを実行することができる。
チーム ビルドで認識できるように、ビルドの種類は特定のフォルダに配置する必要がある。	ビルド定義のプロジェクト ファイル (tfsbuild.proj) は、バージョン管理の階層の任意の場所に保存できる。
カスタム ソリューションを使用して、スケジュール ビルドを作成する。	カスタム ソリューションを使用せずに Visual Studio のスケジュール ビルドを作成できる。
すぐに使用できる一連のチェックインポリシーがある。	CI ビルドの失敗についての新しいチェックインポリシーを使用できる。これにより、CI ビルドが失敗するときにコードをチェックインできないようにする。
ツール converter.exe を使用して、VSS から Team Foundation Server へ移行する。	Visual Studio ツールキットを使用して、Team Foundation Server と、VSS などの他のソース管理システムの間の変換を構築し、ソリューションをミラーリングする。
ワークスペース マッピングを使用して、ローカル マシンと同期したいファイル セットを定義する。	Team Foundation Server 2008 では、クローク フォルダのフォルダまたはファイルのマッピングが可能になり、また、ワイルドカード マッピングによってサブフォルダをマッピングせずに 1 つのフォルダ内のすべてのファイルをマップすることができる。
workspacemapping.xml ファイルを使用してワークスペース マッピングを修正する。	Team Foundation Server 2008 の GUI を使用して、ワークスペース マッピングを管理する。workspacemapping.xml は使用しない。
TFS Power Tool を使用してオフラインで作業する。	Visual Studio 統合開発環境を使用してオフラインで作業する。
ファイルの最新バージョンを取得することと、編集のためにそれをチェックアウトすることは、ソース管理において 2 つの別の操作である。	オプションを使用して、編集のためにファイルをチェックアウトするときに、ファイルの最新バージョンを自動的に取得することができる。

あらかじめ作成されているステップをカスタマイズして、他のチーム プロジェクトからプロジェクト アセンブリを参照するときに依存関係を取得する。	ビルド定義のワークスペース テンプレートは VS GUI で管理されており、複数のチーム プロジェクトからのマッピング パスなど、標準のワークスペースの柔軟性を完全に備えている。
TFSBuild コマンドライン ツールを使用してビルドを削除する。	Visual Studio 統合開発環境を使用してビルドを停止および削除する。

参考資料

- Visual Studio 2008 Team Foundation Server の詳細については、http://go.microsoft.com/?linkid=6625887 の「An Overview of Microsoft Visual Studio Code Name "Orcas" White Paper」を参照してください。

ガイドライン : チーム ビルド

索引

戦略

- スケジュールされたビルドを使用して定期ビルドを生成する
- 継続的な統合 (CI) ビルドを使用して、チェックインに関するフィードバックを迅速に得る
- CI ビルドがビルド サーバーのパフォーマンスに悪影響を及ぼす場合にローリング ビルドを使用する
- 分岐を使用してビルドの不具合を減らす
- チェックイン ポリシーを使用してチェックインの品質を向上させる
- ビルドの通知アラートを使用して、いつビルドが完了したのかを知る

分岐

- 部分分岐の作成時に、新しいチーム ビルドの種類を使用する
- 完全分岐の作成時に、TFSBuild.proj ファイル内のソリューションへのパスを変更する

チェックイン ポリシー

- チェックイン ポリシーを使用してチェックインの品質を向上させる
- チェックイン ポリシーを使用して作業項目をビルドに関連付ける

継続的な統合ビルド

- CI ビルドを使用して、チェックインに関するフィードバックを迅速に得る
- CI ビルドがビルド サーバーのパフォーマンスに悪影響を及ぼす場合にローリング ビルドを使用する
- ローリング ビルドの周期がビルド時間よりも確実に短くなるようにする

カスタマイズ

- カスタム ポストビルド ステップを使用してインストーラ プロジェクトをビルドする
- MS Build Toolkit Extras を使用して Microsoft .NET 1.1 アプリケーションをビルドする
- TFSBuild.proj を使用してビルドを修正する
- カスタム プリビルド ステップを使用して、別のチーム プロジェクトと依存関係にあるプロジェクトをビルドする

配置

- 大規模なチームの場合に、ビルド サービスを別のサーバーにインストールする

パフォーマンス

- インクリメンタル ビルドを使用してパフォーマンスを向上させる
- ビルド時に不要なフォルダを同期しないようにする
- ワークスペースを使用して、不要なファイルとプロジェクトがチーム ビルドの最中にチェックアウトされないようにする
- パフォーマンスを向上させるため、ビルド マシンを複数台使用してみる

プロジェクト

- チーム プロジェクト間に依存関係が生じないようにする
- ファイル参照の代わりにプロジェクト参照を使用する
- Web アプリケーションに Web 配置プロジェクトを使用する
- 小規模なチーム プロジェクトで作業している場合は、単一ソリューション戦略を使用する
- 独立した複数のサブプロジェクトを持つ大規模なチーム プロジェクトで作業している場合は、分割ソリューション戦略を使用する
- 独立した多数のサブプロジェクトを必要とする非常に大規模なチーム プロジェクトで作業している場合は、複数ソリューション戦略を使用する

スケジュールされたビルド

- スケジュールされたビルドを使用して定期ビルドを生成する

テストドリブン開発

- ビルドごとにコード分析を実行する
- ビルドごとに自動テストを実行する
- 自動テストに不合格になったときビルドを停止する設定にする

作業項目

- 作業項目を使用してビルドの中断を追跡する

戦略

- **スケジュールされたビルドを使用して定期ビルドを生成する**
- **CI ビルドを使用して、チェックインに関するフィードバックを迅速に得る**
- **CI ビルドがビルド サーバーのパフォーマンスに悪影響を及ぼす場合にローリング ビルドを使用する**
- **分岐を使用してビルドの中断を減らす**
- **チェックイン ポリシーを使用してチェックインの品質を向上させる**
- **ビルドの通知アラートを使用して、いつビルドが完了したのかを知る**

スケジュールされたビルドを使用して定期ビルドを生成する

予測可能な一定の間隔ごとにビルドを生成するときは、スケジュールされたビルドを使用してください。

一般的に、テスト チームおよびその他に提供されるビルドは、高い信頼性が必要であり、決まった時間間隔ごとに利用できるものでなければなりません。この条件が満たされれば、ビルドに関するフィードバックがタイミングよく収集できます。

Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) のチーム ビルド機能では、ユーザー インターフェイスからのスケジュールされたビルドをサポートしていません。その代わりに、Microsoft Windows® のタスク スケジューラを使用することにより、TFSBuild コマンド ライン ユーティリティを実行して所定の時刻にビルドを開始することができます。

スケジュールされたビルドを作成するには、以下の操作を行います。

1. 以下のように TFSBuild コマンド ラインを作成します。

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

2. 作成したコマンドラインをバッチ ファイルに挿入します。
3. 所定の間隔でこのバッチ ファイルを実行する Windows スケジュール タスクを作成します。

参考資料

- チーム ビルドでスケジュールされたビルドを設定する詳細については、このガイドの「第 9 章 チーム ビルドにおけるスケジュール済みのビルドのセットアップ」を参照してください。
- TFS でスケジュールされたビルドを設定する詳細については、このガイドの「Visual Studio Team Foundation Server でスケジュール済みのビルドをセットアップする方法」を参照してください。

継続的な統合ビルドを使用して、チェックインに関するフィードバックを迅速に得る

大きな変更点やビルドの品質に関するフィードバックを、チェックインするたびに迅速に開発チームに提供したい場合は、CI ビルドを使用することが望ましいです。 開発チームのビルドの問題点が短時間で解消され、コードの品質を向上させるツールとして使用することも可能です。

Team Foundation Server 2005 には、CI ソリューションは用意されていませんが、独自の CI ビルド ソリューションを実装するためのフレームワークは提供されています。TFS で CI ビルドを設定する詳細については、「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。この方法の項目では、Microsoft Visual Studio Team System (VSTS)

開発チームから提供されるソリューションを使用します。このソリューションでは、TFS サーバーにアクセスできるアカウントの下で稼働する Web サービスをインストールします。Team Foundation Server は、特定のイベントが発生したときに電子メール メッセージを送信することも、Web サービスを呼び出すことも可能です。CI ソリューションでは、このイベント メカニズムによって **CheckinEvent** イベントに Web サービスが登録されます。そのため、この Web サービスによって、チェックインが生じるたびに、Web サービスはチーム ビルドを開始します。

参考資料

- 詳細については、このガイドの「第 8 章 チーム ビルドでの継続的な統合のセットアップ」を参照してください。
- CI ビルドの設定の詳細については、このガイドの「Visual Studio Team Foundation Server でスケジュール済みのビルドをセットアップする方法」を参照してください。
- VSTS CI ソリューションの使用方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181710(vs.80).aspx の「Team Foundation ビルドの概要」を参照してください。
- VSTS CI ソリューション MSI をダウンロードするときは、http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi にアクセスしてください。
- TFS でのアジャイル開発と CI の詳細については、http://www.microsoft.com/japan/msdn/msdnmag/issues/06/03/TeamSystem/default.aspx の「常時結合を可能にするための Team Foundation Server の拡張」を参照してください。

CI ビルドがビルド サーバーのパフォーマンスに悪影響を及ぼす場合にローリング ビルドを使用する

チェックインするたびにすぐにビルドするのは、最も簡単な CI 戦略であり、一般的に最も速くフィードバックが得られる方法です。しかし、ビルド サーバーの負担になるほど速くチェックインが行われてしまう場合は、所定の回数だけチェックインが行われた後か、所定の期間が経過した後のいずれかにビルドが行われるローリング ビルドという方法を使用することが望ましいです。ローリング ビルドを使用する必要があるかどうかを判断するときは、以下の項目を調べてください。

- チーム ビルドの長さ (分単位)
- チェックインの平均周期 (分単位)
- チェックインが頻繁に行われる時間帯

ビルドの長さがチェックインの平均周期より長い場合は、最初のビルドが完成しきらないうちに次のチェックインが行われるため、各ビルドが連続して実行されます。各ビルドが完成する前にチェックインが連続して行われる場合は、ビルド サーバーのパフォーマンスに影響が及び、(スケジュールされたビルドなど) 他のビルドが開始されなくなります。チェックインが頻繁に行われる時間帯を見直して、CI ビルドがスケジュールされたビルドや他の重要なチーム ビルドの実行に影響するかどうかを判断してください。

参考資料

- 詳細については、このガイドの「第 8 章 チーム ビルドでの継続的な統合のセットアップ」を参照してください。

分岐を使用してビルドの不具合を減らす

ビルドの不具合をできるだけ避けるため、アクティブな開発には Development 分岐を使用し、統合ビルドには Main 分岐を使用するようにしてください。

下の例は、Development 分岐を作成した後に分岐がどのような形になるかを示した例です。

- **Development** – 開発分岐
    - **Source**
- **Main** – 統合分岐
    - **Source**
    - **その他のアセット フォルダ**

リリース分岐を操作するときは、以下の注意点を忘れないでください。

- **分岐するとき ：** デイリー ビルドを作成していて、ビルドの安定化とビルド統合とに問題が生じる場合は、メイン分岐と開発分岐の両方を作成して、デイリー ビルドの予測可能性をもっと高くすることが望ましいです。また、チェックイン ポリシーをさらに厳しくして、チェックインの品質向上を検討しても良いでしょう。
- **分岐しないとき ：**CI ビルドしか作成しない場合、または既にデイリー ビルドが予測可能なほど安定している場合は、統合分岐の余分なオーバーヘッドは不要なことがあります。
- **分岐に対する権限 ：**
    - **Main** 分岐に対する権限については、マージと統合を担当する開発者には読み取り/書き込み権限を付与し、他全員には読み取り専用権限を付与することが望ましいです。
    - **Development** 分岐に対する権限については、全員に読み取り/書き込み権限を付与することが望ましいです。
- **分岐内でのビルドの周期 ：**
    - **Main** 分岐でのデイリー ビルド
    - **Development** 分岐での CI ビルド
- **分岐に対するテスト項目 ：**
    - **Main** 分岐に対して、統合テスト、パフォーマンス テスト、セキュリティ テストを実行してください。
    - **Development** 分岐に対して、機能テストとクイック フィードバック テストを実行してください。

Main 分岐は、既に開発分岐へのチェックインが済んだ変更を統合するためのステージング エリアとして使用してください。アクティブな開発はすべて **Development** 分岐で実行し、不具合を生じない変更を **Main** 分岐に統合してください。

参考資料

- 分岐戦略およびマージ戦略の定義の詳細については、このガイドの「第 5 章 分岐とマージの方針の定義」を参照してください。
- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and

Merging Primer」を参照してください。

- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

チェックイン ポリシーを使用してチェックインの品質を向上させる

チェックインの品質を向上させるためには、コード分析とテスト ポリシーとを組み合わせて使用することが望ましいです。たとえば、特定のテストを実行し、そのテストに合格しないと、ソースが TFS ソース管理にチェックインされないようにするには、付属のテスト ポリシーを使用します。また、セキュリティ、パフォーマンス、移植性、メンテナンス性、信頼性の各ルールを確実にすることにより、コードが一定の品質基準を満たすようなコード分析ポリシーを設定することも可能です。

この種のチェックイン ポリシーを適用するだけでなく、コーディングの基準とガイドラインとを義務付けるポリシーも適用することにより、特定のコード品質を持つコードに対してテストをすることができます。

チーム プロジェクトのコード分析チェックイン ポリシーを適用するときは、[チーム エクスプローラ] で、チーム プロジェクトを右クリックし、[チーム プロジェクトの設定] を指して、[ソース管理] をクリックします。[チェックイン ポリシー] タブをクリックして、[追加] をクリックし、適切なポリシーを選択して設定します。

参考資料

- カスタム チェックイン ポリシーの作成および使用の詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してくださ

い。

- チェックイン ポリシーのカスタマイズ方法を知りたいときは、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインの際に特定のパターンだけを禁止するサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインの際にコメントを義務付けるサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーの登録方法を知りたいときは、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。

ビルド通知アラートを使用して、いつビルドが完了したのかを知る

ビルドが完了したときに自身または他者に電子メール メッセージを送信するアラートを作成しておくと、ビルドの経過を追跡することができます。

この機能は重要です。チーム間での作業の受け渡しがスムーズになるからです。たとえば、テスト チームは、完了したビルドについて電子メールで通知を受ければ、わざわざ指示を待たなくてもテスト段階に入ることができます。

参考資料

- ビルド通知については、http://msdn2.microsoft.com/ja-jp/library/ms181725(VS.80).aspx の「方法 : ビルドの通知の電子メールを受け取る」を参照してください。
- ビルド通知の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181335(VS.80).aspx の「方法 : 警告を追加または編集する」を参照してください。

分岐

- **部分分岐の作成時に、新しいチーム ビルドの種類を使用する**
- **完全分岐の作成時に、TFSBuild.proj ファイル内のソリューションへのパスを変更する**

部分分岐の作成時に、新しいチーム ビルドの種類を使用する

ソリューションの一部をチーム プロジェクトに含む分岐を作成するときは、ビルドの処理を正常に行う手段として、新しいビルドの種類を作成しなければならないことがあります。

部分分岐には次の 2 つの種類があります。

1. **どのビルドの種類の分岐も含まない部分分岐 ：** この部分分岐は、チーム プロジェクトの内部に生じますが、ソリューション ファイルとソース ファイルを分岐するだけであり、TeamBuildTypes フォルダを新しいフォルダに分岐する働きはしません。
2. **ビルドの種類の分岐を含む部分分岐 ：** この部分分岐は、チーム プロジェクトの内部に生じますが、TeamBuildTypes サブフォルダ (ビルドの種類のこと) の一部を分岐する以外に、関係のあるソリューション ファイルとソース ファイルを含む他のフォルダを分岐する働きもします。

チーム ビルドの種類を含まない部分分岐を作成した場合でも、既存のチーム ビルドはすべて機能し続けますが、その分岐をビルドする場合は、新しいチーム ビルドの種類を作成する必要が生じます。新しい分岐の内部でコードをビルドする場合は、チーム ビルド ウィザードを使用して新しいビルドの種類を作成してください。この新しいビルドの種類は、必ず、新しい分岐場所をポイントすることになりますが、場合によっては、そのビルドに含まれなければならないのに分岐されなかったソリューションの、親の場所をポイントすることもあります。

チーム ビルドの種類を含む部分分岐を作成する場合、その分岐と一緒にコピーされるビルドの種類は必ず、最初の親分岐の場所をポイントするため、新しい分岐をビルドすることができなくなります。分岐されたビルドの種類は、分岐された新しいコード場所をポイントするように修正してください。

参考資料

- ビルドの種類の更新方法については、

http://msdn2.microsoft.com/ja-jp/library/ms252500(VS.80).aspx の「方法 : 分岐したチーム プロジェクトのビルドの種類を更新する」を参照してください。

完全分岐の作成時に、TFSBuild.proj ファイル内のソリューションへのパスを変更する

新しい分岐 (チーム ビルドの種類など) を作成する際、ビルドの種類に含まれているパスは依然として前の場所をポイントしています。新しい分岐でビルドが機能するためには、ビルドの種類のプロジェクト ファイルに含まれているパスを更新して、分岐処理後に作成される新しいパス ロケーションを参照するようにしなければなりません。

フル分岐の作成時にはビルドの種類の分岐も行います。ビルドの種類には、最初のソース管理ツリーを起点とする、各フォルダへの参照が含まれています。このビルドの種類が分岐フォルダを参照するようにするためには、各プロジェクト ファイルに含まれているフォルダ参照を編集しなければなりません。

更新を実行するときは、変更する分岐からビルドの種類をチェックアウトし、アップデートを適用してから、変更内容を分岐に適用してください。

参考資料

- ビルドの種類の更新方法については、http://msdn2.microsoft.com/ja-jp/library/ms252500(VS.80).aspx の「方法 : 分岐したチーム プロジェクトのビルドの種類を更新する」を参照してください。

チェックイン ポリシー

- **チェックイン ポリシーを使用してチェックインの品質を向上させる**
- **チェックイン ポリシーを使用して作業項目をビルドに関連付ける**

チェックイン ポリシーを使用してチェックインの品質を向上させる

チェックインの品質を上げるときは、コード分析とテスト ポリシーとを組み合わせて使用してくださ

い。たとえば、特定のテストを実行し、そのテストに合格しないと、ソースが TFS ソース管理にチェックインされないようにするときは、付属のテスト ポリシーを使用します。また、セキュリティ、パフォーマンス、移植性、メンテナンス性、信頼性の各ルールを確実にすることにより、コードが一定の品質基準を満たせるようなコード分析ポリシーを設定することも可能です。

この種のチェックイン ポリシーを適用するだけでなく、コーディングの基準とガイドラインとを義務付けるポリシーも適用することにより、特定のコード品質を持つコードに対してテストをすることができます。

参考資料

- カスタム チェックイン ポリシーの作成および使用の詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してください。
- チェックイン ポリシーのカスタマイズ方法を知りたいときは、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインの際に特定のパターンだけを禁止するサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Check-in Policy to Disallow Certain Patterns」を参照してください。
- チェックインの際にコメントを義務付けるサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Check-in Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーの登録方法を知りたいときは、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。

チェックイン ポリシーを使用して作業項目をビルドに関連付ける

チェックインを作業項目に関連付けることを開発者に義務付けるときは、作業項目チェックイン ポリシーを設定してください。

ビルドに問題が生じた場合は、そのビルドにどの変更セットが関連付けられているのかを知り、その変更セットが関連付けられているのはどの作業項目なのかを知ることが大切です。これがわかれば、変更されたコードのチェックインをどの開発者が担当しているかがわかり、その開発者がプロジェクトのどの区分で作業しているのかもわかります。

完了した一連の作業項目にビルドを関連付けるためには、各チェックインを作業項目に関連付けなければなりません。このチェックインは、ビルドに関連付けられた変更セットとして示されます。ビルドから、変更セット、作業項目に至るまでたどることが可能です。

参考資料

- カスタム チェックイン ポリシーの作成および使用の詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してください。
- チェックイン ポリシーのカスタマイズ方法を知りたいときは、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。

継続的な統合ビルド

• CI ビルドを使用して、チェックインに関するフィードバックを迅速に得る

• CI ビルドがビルド サーバーのパフォーマンスに悪影響を及ぼす場合にローリング ビルドを使用する

• ローリング ビルドの周期がビルド時間よりも確実に短くなるようにする

CI ビルドを使用して、チェックインに関するフィードバックを迅速に得る

チェックインした後すぐに、ビルドの大きな変更点や品質に関するフィードバックを開発チームに提供したい場合は、継続的な統合ビルドを使用することが望ましいです。この方法だと、ビルドの問題点がタイミングよく修正でき、コードの品質を向上させるツールとしても使用できます。

Visual Studio 2005 Team Foundation Server には、CI ソリューションは用意されていませんが、独自の CI ビルド ソリューションを実装するためのフレームワークは提供されています。

TFS で CI ビルドを設定する詳細については、「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。この方法の項目では、VSTS 開発チームから提供されるソリューションを使用します。このソリューションでは、TFS サーバーにアクセスできるアカウントの下で稼働する Web サービスをインストールします。Team Foundation Server は、特定のイベントが発生したときに電子メール メッセージを送信することも Web サービスを呼び出すことも可能です。CI ソリューションでは、このイベント メカニズムによって **CheckinEvent** イベントに Web サービスが登録されます。そのため、チェックインが生じるたびに Web サービスはチーム ビルドを開始します。

参考資料

- CI ビルドの詳細については、このガイドの「第 8 章 チーム ビルドでの継続的な統合のセットアップ」を参照してください。
- CI ビルドの設定の詳細については、このガイドの「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。
- VSTS CI ソリューションの使用方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181710(vs.80).aspx の「Team Foundation ビルドの概要」を参照してください。
- Visual Studio Team System CI ソリューション MSI をダウンロードするときは、http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi にアクセスしてください。
- TFS でのアジャイル開発と CI の詳細については、http://www.microsoft.com/japan/msdn/msdnmag/issues/06/03/TeamSystem/default.aspx の「常時結合を可能にするための Team Foundation Server の拡張」を参照してください。

CI ビルドがビルド サーバーのパフォーマンスに悪影響を及ぼす場合にローリング ビルドを

使用する

チェックインするたびにすぐにビルドするのは、最も簡単な CI 戦略であり、一般的に最も速くフィードバックが得られる方法です。しかし、ビルド サーバーの負担になるほど速くチェックインが行われてしまう場合は、所定の回数だけチェックインが行われた後か、所定の期間が経過した後のいずれかにビルドが行われるローリング ビルドという方法を使用することが望ましいです。ローリング ビルドを使用する必要があるかどうかを判断するときは、以下の項目を調べてください。

- チーム ビルドの長さ (分単位)
- チェックインの平均周期 (分単位)
- チェックインが頻繁に行われる時間帯

ビルドの長さがチェックインの平均周期より長い場合は、最初のビルドが完成しないうちに次のチェックインが行われるため、各ビルドが連続して実行されます。各ビルドが完成する前にチェックインが連続して行われる場合は、ビルド サーバーのパフォーマンスに影響が及び、スケジュールされたビルドなど他のビルドに遅れが生じます。チェックインが頻繁に行われる時間帯を見直して、CI ビルドがスケジュールされたビルドや他の重要なチーム ビルドの実行に影響するかどうかを判断してください。

参考資料

- CI ビルドの設定の詳細については、このガイドの「第 8 章 チーム ビルドでの継続的な統合のセットアップ」を参照してください。

ローリング ビルドの周期がビルド時間よりも確実に短くなるようにする

ローリング ビルドの時間間隔は、ビルドが効率よく処理されるような時間間隔に設定することが重要です。ローリング ビルドの周期が、ビルドの完成に要する時間よりも短ければ、ローリング ビルドの休止期間の間にビルド マシンを他のビルドの種類に利用させることができます。

ローリング ビルドの理想的な間隔を決定するときは、チェックインの平均周期をビルドの長さで割ってください。たとえば、10 分かかるビルドがあり、1 回のチェックインが平均して 5 分だとすると、チェックイン間隔はチェックイン 2 回分に設定し、タイムアウト間隔は 10 分に設定できます。こう

すれば、ビルドが完成してから次のビルドが始まるようになります。ビルド サーバーに負荷がかかりすぎるときは、この値を増やしてください。

参考資料

- CI ビルドの詳細については、このガイドの「第 8 章 チーム ビルドでの継続的な統合のセットアップ」を参照してください。

カスタマイズ

- **カスタム ポストビルド ステップを使用してインストーラ プロジェクトをビルドする**
- **MS Build Toolkit Extras を使用して Microsoft .NET 1.1 アプリケーションをビルドする**
- **TFSBuild.proj を使用してビルドを修正する**
- **カスタム プリビルド ステップを使用して、別のチーム プロジェクトと依存関係にあるプロジェクトをビルドする**

カスタム ポストビルド ステップを使用してインストーラ プロジェクトをビルドする

チーム ビルドは既定ではセットアップ プロジェクトに対応していないため、セットアップ プロジェクトをコンパイルしてそのバイナリをビルド ドロップ ロケーションにコピーするときは、カスタム ポストビルド ステップを使用することが望ましいです。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms404859(VS.80).aspx の「チュートリアル：Visual Studio セットアップ プロジェクトをビルドするために Team Foundation ビルドを構成する」を参照してください。

MS Build Toolkit Extras を使用して Microsoft .NET 1.1 アプリケーションをビルドする

チーム ビルドは、既定では .NET 1.1 アプリケーションに対応していません。MSBuild Extras –

Toolkit for .NET 1.1 (MSBee) では、.NET 1.1 のビルドは可能ですが、プロジェクトとソリューションを Visual Studio 2005 にアップグレードする必要があります。Visual Studio 2005 のプロジェクトとソリューションにアップグレードできない場合でも、カスタム ポストビルド ステップを使用すれば .NET 1.1 アプリケーションはコンパイルできます。

参考資料

- MSBee をダウンロードするときは、http://www.codeplex.com/MSBee にアクセスしてください。
- カスタム ポスト ステップを作成して .NET 1.1 アプリケーションをコンパイルする詳細については、http://blogs.msdn.com/nagarajp/archive/2005/10/26/485368.aspx にある Nagaraju のブログ エントリを参照してください。

TFSBuild.proj を使用してビルドを修正する

ビルド サーバー、ドロップ ロケーション、ビルド ディレクトリなど、ビルドに関する情報を修正するときは、TFSBuild.proj ファイルを編集して修正します。

TFSBuild.proj ファイルには、チーム ビルドを実行するのに必要な情報が多く含まれています。この情報には、ビルド ロケーションが含まれているほか、当該ビルドで静的コード分析と単体テストを実行すべきかどうかに関する情報も含まれています。ビルドを修正するときは TFSBuild.proj ファイルを編集します。

TFSBuild.proj ファイルを編集するには、次の操作を行います。

1. ソース管理からファイルをチェックアウトします。
2. ファイルに含まれているビルド情報を更新します。
3. 再びファイルをチェックインして変更を適用します。

次回このビルドを実行したときには、修正されたビルド データが使用されます。

参考資料

- Team Foundation Build のカスタマイズの詳細については、http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx の「Customizing Team Foundation Build」を参照してください。

カスタム プリビルド ステップを使用して、別のチーム プロジェクトと依存関係にあるプロジェクトをビルドする

チーム ビルドは、複数のチーム プロジェクトを横断するようなビルディング ソリューションに対応していません。このようなビルディング ソリューションを可能にするためには、TFSBuild.proj ファイルをカスタマイズして、ビルドの依存先となるプロジェクトから、必要なコードをチェックアウトしなければなりません。

参考資料

- 詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の「Working with multiple team projects in Team Build」を参照してください。
- Team Foundation Build のカスタマイズの詳細については、http://msdn2.microsoft.com/en-us/library/ms400688(VS.80).aspx の「Customizing Team Foundation Build」を参照してください。

配置

- **大規模なチームの場合に、ビルド サービスを別のサーバーにインストールする**

大規模なチームの場合に、ビルド サービスを別のサーバーにインストールする

大規模なチーム ビルドは、処理に時間がかかり、大量のサーバー リソースを消費するおそれがあります。そのため、TFS サーバーでビルドを実行する場合に、サーバーの信頼性、パフォーマンス、拡張性に影響が出ます。

ビルドのパフォーマンスを上げ、アプリケーション層にかかる負荷を減らすため、専用のビルド サーバーでビルドを実行することを推奨します。

参考資料

- 『Team Foundation Installation Guide』をダウンロードするとき、あるいは、TFS およびチーム ビルドをインストールする詳細について知りたいときは、http://www.microsoft.com/downloads/details.aspx?FamilyId=E54BF6FF-026B-43A4-ADE4-A690388F310E&displaylang=en にアクセスしてください。

パフォーマンス

- **インクリメンタル ビルドを使用してパフォーマンスを上げる**
- **ビルド時に不要なフォルダを同期しないようにする**
- **ワークスペースを使用して、不要なファイルとプロジェクトがチーム ビルドの最中にチェックアウトされないようにする**
- **パフォーマンスを上げるため、ビルド マシンを複数台使用してみる**

インクリメンタル ビルドを使用してパフォーマンスを上げる

既定では、ビルドに必要なソース コード ツリーが全部チェックアウトされる前に、そのビルドの実行に使用されるディレクトリの内容がチーム ビルドによって消去されます。ビルドのソースをチェックアウトするのに使用されるワークスペースも、チーム ビルドによって削除され、初期化し直されます。前回のチーム ビルド以後に変化したソースだけが取得されるようチーム ビルドを設定すれば、パフォーマンスが上がります。

ビルドに必要なソースの量が多く、しかもビルド サーバーが TFS サーバーと異なる場所にある場合は、ソース コードのチェックアウトを実行するのに長時間を要するおそれがあります。そのような場合は、インクリメンタル ビルドの使用を検討してください。インクリメンタル ビルドを実行するためには、TFSBuild.proj ファイル内のいくつかの値が true になるよう設定する必要があります。以下の操作が必要です。

- チーム ビルドによってローカル ビルド フォルダとソース フォルダの中身が消去されるのを防ぐ。
- ビルドに使用されるワークスペースがチーム ビルドによって作成し直されるのを防ぐ。
- 変更されたソースだけがソース管理から取得されるようチーム ビルドを設定する。

インクリメンタル ビルドを実行するには、次の操作を行います。

1. インクリメンタル ビルドとなる新しいビルドの種類を作成します。
2. 今作成したインクリメンタル ビルドの種類に関連付けられている TFSBuild.proj ファイルをチェックアウトして、編集できるようにします。
3. TFSBuild.proj の末尾にある </project> という閉じ要素の直前に、以下の <PropertyGroup> セクションを追加します。

```
<PropertyGroup>
<SkipClean>true</SkipClean>
<SkipInitializeWorkspace>true</SkipInitializeWorkspace>
<ForceGet>false</ForceGet>
</ PropertyGroup>
```

以上の設定により、次のようになります。

• **SkipClean :** SkipClean を true に設定したことにより、ローカル ビルドとソース フォルダの中身が消去されなくなります。

• **SkipInitializeWorkspace :** SkipInitializeWorkspace を true に設定したことにより、既存のワークスペースが、ビルド マシンの適所に配置されたままとなります。

• **ForceGet:** ForceGet を false に設定したことにより、更新されたソースだけが取得されるようになり、ワークスペースのソースが全部取得されることはなくなります。

参考資料

- 詳細については、このガイドの「プラクティス一覧: チーム ビルド」を参照してください。
- インクリメンタル ビルドを設定する詳細については、

http://msdn2.microsoft.com/ja-jp/library/aa833876(VS.80).aspx の「方法 : インクリメンタル ビルド用に Team Foundation ビルドを構成する」を参照してください。

ビルド時に不要なフォルダを同期しないようにする

ワークスペース マッピングの規模を縮小するか、ビルドの一部として不要なフォルダをクロークするか、いずれかを行うことが望ましいです。

チーム ビルドの実行時に、サーバーは、必要なすべてのファイルをソース管理から取得します。取得されるファイルは、チーム ビルドの種類の作成に使用されるワークスペースによって定義されます。ワークスペースにマップされるファイルの中には、不要なものがある場合があります。含まれるフォルダの数が減るようワークスペースの定義を変更することができます。また、不要なファイルをクロークすることにより、ビルドの一部として取得されないようにすることも可能です。

たとえば、新規プロジェクトのデフォルト マッピングが $/TeamProject であるとします。すべてのソース ファイルが $/TeamProject/foo/bar/foobar/sources 以下にある場合は、そのディレクトリだけをマップすることが望ましいです。

ワークスペース マッピング以下にあるファイルがクロークされるように、WorkspaceMapping.xml ファイルを編集することができます。この WorkspaceMapping.xml ファイルは、チーム ビルドの種類の作成時に作成され、ビルドの実行時に取得されるフォルダの定義に使用されるファイルのことです。ビルドの一部として、不要なファイルとフォルダはクロークできます。ファイルを個別にクロークするとメンテナンス オーバーヘッドの生じるおそれがあるため、フォルダをクロークする方が好ましいです。

フォルダをクロークするには、次の操作を行います。

1. ソース管理から WorkspaceMapping.xml をチェックアウトします。
2. このファイルに、適切なクローク エントリを追加します。
3. WorkspaceMapping.xml をチェックインします。

下の例のようにすると、チーム ビルドの間に documentation フォルダは取得されなくなります。

```
<Mappings>
<InternalMapping ServerItem="$/MyTeamProject" LocalItem="c:¥projects¥teamproject”
Type="Map" />
<InternalMapping ServerItem="$/MyTeamProject/documentation" Type="Cloak" />
</Mappings>
```

参考資料

- ワークスペースでフォルダをクロークする詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181378(VS.80).aspx の「方法 : ワークスペースでフォルダをクロークまたはクローク解除する」を参照してください。
- チーム ビルドにフォルダを無視させる詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/10/13/480584.aspx の「How to make ‘Team Build’ skip getting certain folders?」を参照してください。
- WorkspaceMapping スキーマの詳細については、http://blogs.msdn.com/buckh/archive/2007/02/28/schema-for-the-workspacemapping-xml-file.aspx の「Schema for the WorkspaceMapping.xml file」を参照してください。

ワークスペースを使用して、不要なファイルとプロジェクトがチーム ビルドの最中にチェックアウトされないようにする

チーム ビルドは、ビルドを実行する手段としてソースをチェックアウトする働きをします。プロジェクトのソース ツリーが大きい場合は、ソースをチェックアウトするのに非常に時間がかかることがあります。チーム プロジェクトの一部だけをビルドする場合は、必要なソースだけがチェックアウトされるようにすることが望ましいです。

大規模なチーム プロジェクトには、Visual Studio ソリューションが複数含まれ、それぞれのソリューションが、チーム プロジェクトの個々の部分をビルドするのに使用されます。チーム ビルドの種類を

作成するときには、そのチーム ビルドに使用するソリューションを指定します。ワークスペースを指定せずにソリューション ファイルを指定した場合は、ビルドの実行前に、チーム プロジェクトに含まれているすべてのソースがチェックアウトされます。

必要なソースだけをチェックアウトするためには、最初にワークスペースを定義しなければなりません。チーム ビルドの種類を作成する前に、まずワークスペースを定義しておいて、そのワークスペースに、ビルドするソリューションだけをマップします。チーム ビルドの種類を定義するときは、定義の済んだワークスペースを選択してから、目的のソリューションを選択してください。この方法なら、そのワークスペースで定義したソースだけがチェックアウトされます。

参考資料

- チーム ビルドの種類を作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181286(VS.80).aspx の「チュートリアル : Team Foundation ビルドでのビルドの種類の作成」を参照してください。
- ワークスペースに含まれているすべてのソース コードがチーム ビルドによってチェックアウトされる理由については、http://blogs.msdn.com/anutthara/archive/2005/12/07/500923.aspx の「Why does Team Build sync all sources in spite of my selecting only a subset of solutions?」を参照してください。

パフォーマンスを上げるため、ビルド マシンを複数台使用してみる

ビルドの種類が複数あり、そのすべてを 1 台のビルド サーバーで実行した場合、処理しきれなくなることがあります。そのような場合は、各ビルドの種類を何台かのビルド サーバーに分けて実行することを検討することが望ましいです。

ビルドは実行に長い時間のかかることがあり、特に、大規模なプロジェクトの場合はなおさらです。CI または頻度の高いスケジュールされたビルドを使用している場合は、生成されるビルドの量にビルド サーバーが追いつけないおそれがあります。

ビルド サーバーを複数設置すれば、各サーバーに負荷が分散できます。各ビルドの種類をそれぞれのサーバーに割り当てて、ビルドの負荷を均等に散らしてください。

参考資料

- 詳細については、このガイドの「第 7 章 チーム ビルドの説明」を参照してください。

プロジェクト

- **チーム プロジェクトどうしの間に依存関係が生じないようにする**
- **ファイル参照の代わりにプロジェクト参照を使用する**
- **Web アプリケーションに Web デプロイメント プロジェクトを使用する**
- **小規模のチーム プロジェクトで作業している場合に、単一ソリューション戦略を使用する**
- **独立した複数のサブプロジェクトを持つ大規模なチーム プロジェクトで作業している場合に、分割ソリューション戦略を使用する**
- **独立した多数のサブプロジェクトを必要とする非常に大規模なチーム プロジェクトで作業している場合に、複数ソリューション戦略を使用する**

チーム プロジェクトの間に依存関係が生じないようにする

一般的に、複数のチーム プロジェクトを横断するような依存関係は避け、その代わりに、関連のあるソリューション/プロジェクトもしくは依存しているソリューション/プロジェクトはすべて、同じチーム プロジェクトの下に配置することが望ましいです。そうすれば、ビルド スクリプトをカスタマイズする必要性が減ります。依存関係がある場合は、その依存関係の定義にプロジェクト参照を使用するか、共有プロジェクトから自身のプロジェクトへその依存関係を分岐するか、いずれかを行ってください。ファイル参照は管理が難しいため、使用しない方が望ましいでしょう。

参考資料

- ワークスペースを作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 ： ワークスペー

スを作成する」を参照してください。

- ワークスペースを編集する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 ： ワークスペースを編集する」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。

ファイル参照の代わりにプロジェクト参照を使用する

同じ Visual Studio ソリューション内の別のアセンブリを参照する必要がある場合は、Visual Studio のプロジェクト参照を使用することが望ましいです。プロジェクト参照を使用すると、いくつかの処理を Visual Studio に自動的に実行させることができます。たとえば、ビルド構成の同期の維持 (デバッグ/リリース) や、バージョン管理のほかに、アセンブリのバージョンが変化したとき必要に応じてコンポーネントを再ビルドする処理が自動化できます。

参考資料

- プロジェクト参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。

Web アプリケーションに Web デプロイメント プロジェクトを使用する

Web デプロイメント プロジェクトは、Visual Studio Web Site プロジェクトか Web Application プロジェクトのいずれかに関連付けられます。Web デプロイメント プロジェクトを使用すると、ビルドの設定が管理できるほか、ASP.NET Web アプリケーションに共通するさまざまな設定も管理できます。たとえば、Web.config、接続文字列、仮想ディレクトリに簡単にアクセスできるようになるし、コンパイルの済んだ Web アプリケーションをホスティング サーバーに簡単に配置できるようにもなります。

参考資料

- 詳細については、http://msdn2.microsoft.com/en-us/asp.net/aa336619.aspx の「Visual

Studio 2005 Web Deployment Projects」を参照してください。

小規模なチーム プロジェクトで作業している場合に単一ソリューション戦略を使用する

小規模なチームで作業している場合は、すべてのプロジェクトを Visual Studio の単一ソリューションで管理することを検討してみてください。単一ソリューションだと、ソリューションを開いたときにコードがすべて利用できるため、開発が単純になります。また、どの参照も、ソリューション内の各プロジェクトを結ぶ参照になるため、参照の設定が簡単になります。ただし、ソリューションの外部にある、サードパーティ製のアセンブリ (購入したコンポーネントなど) を参照するときは、ファイル参照を使用する必要があります。

参考資料

- 詳細については、このガイドの「第 3 章 ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

独立した複数のサブプロジェクトを持つ大規模なチーム プロジェクトで作業している場合に、分割ソリューション戦略を使用する

大規模なチームで作業する場合は、1 個 1 個のソリューションがアプリケーション内の 1 個 1 個のサブシステムに相当するようなソリューションを、複数使用することを検討してみてください。こうしたソリューションを使用すると、すべてのプロジェクトに全部のコードをロードしなくても、システムの部分ごとに作業ができるようになります。ソリューションの構造は、依存関係のあるプロジェクトどうしがグループ化されるような構造にする方がよいです。そうすれば、ファイル参照ではなくプロジェクト参照が使用できるようになります。この機能を使用してアプリケーション全体をビルドする場合は、すべてのプロジェクトを含んだマスター ソリューション ファイルの作成を検討してみてください。

注意 : MSBuild を利用したチーム ビルドでビルドを行う場合は、参照されるプロジェクトの一部を含まないソリューション構造にすることができます。ソリューション全体が先にビルドされ、各ソリューションからバイナリ出力が生成されさえすれば、MSBuild は、ソリューションとビルドの境界の外にあるプロジェクト参照を正常にたどれるようになります。この方法で作成したソリューションは、

Visual Studio の build コマンドではビルドされず、チーム ビルドおよび MSBuild ビルドでのみ機能します。

参考資料

- 詳細については、このガイドの「第 3 章 ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

独立した多数のサブプロジェクトを必要とする非常に大規模なチーム プロジェクトで作業している場合に、複数ソリューション戦略を使用する

多数のプロジェクトを必要とする非常に大規模なソリューションで作業する場合は、ソリューションの拡張性の限界にぶつかるおそれがあります。この場合は、アプリケーションを複数のソリューションに分割してください。ただし、各ソリューションに含まれるすべての参照がプロジェクト参照であるため、アプリケーション全体のマスター ソリューションは作成しないでください。各ソリューションの外部にあるプロジェクト (サードパーティ製のライブラリや、別のサブソリューション内のプロジェクトなど) への参照はファイル参照です。つまり、マスター ソリューションはあり得ないわけです。その代わり、各ソリューションをビルドするときの順序を設定したスクリプトを使用しなければなりません。複数ソリューション構造に関連するメンテナンス タスクの 1 つは、各ソリューションの間を循環する参照を誤って作成しないようにすることです。この構造には、複雑なビルド スクリプトが必要であり、また、依存関係を明確にマッピングする処理も必要です。この構造では、Visual Studio でアプリケーション全体をビルドすることはできません。その代わり、TFS チーム ビルドも MSBuild も直接使用できます。

参考資料

- 詳細については、このガイドの「第 3 章 ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

スケジュールされたビルド

- **スケジュールされたビルドを使用して定期ビルドを生成する**

スケジュールされたビルドを使用して定期ビルドを生成する

予測可能な一定間隔ごとにビルドを生成するときは、スケジュールされたビルドを使用することが望ましいです。

一般的に、テスト チームその他に提供されるビルドは、高い信頼性が必要であり、決まった時間間隔ごとに利用できるものでなければなりません。この条件が満たされれば、ビルドに関するフィードバックがタイミングよく収集できます。

TFS のチーム ビルド機能では、ユーザー インターフェイスからスケジュールされたビルドを実行することはできません。しかし、Microsoft Windows のタスク スケジューラを使用して、TFSBuild コマンド ユーティリティを実行し、あらかじめ設定しておいた時刻にビルドを開始することは可能です。

スケジュールされたビルドを作成するには、次の操作を行います。

1. 以下のように TFSBuild コマンド ラインを作成します。

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

2. 作成したコマンドラインをバッチ ファイルに挿入します。
3. 所定の間隔でこのバッチ ファイルを実行する Windows スケジュール タスクを作成します。

参考資料

- 詳細については、このガイドの「第 9 章 チーム ビルドにおけるスケジュール済みのビルドのセットアップ」を参照してください。
- スケジュールされたビルドを設定する詳細については、このガイドの「Visual Studio Team Foundation Server でスケジュール済みのビルドをセットアップする方法」を参照してください。

テストドリブン開発

- **ビルドごとにコード分析を実行する**
- **ビルドごとに自動テストを実行する**
- **自動テストに不合格になったときビルドを停止する設定にしてみる**

ビルドごとにコード分析を実行する

ビルドの品質を上げるため、ビルドの一環としてコード分析を使用してください。ビルドにコード分析ステップを設けることにより、コードが一定の品質基準を満たせるようになり、その結果、セキュリティ、パフォーマンス、移植性、メンテナンス性、信頼性の各ルールが確実になります。

ビルドの種類に関するコード分析をオンにするときは、新しいチーム ビルドの種類を作成するときにチーム ビルドの種類ウィザードで [コード分析] チェック ボックスをオンにするか、ビルドの種類が作成された後で TFSBuild.proj ファイルを変更するか、いずれかを行ってください。

TFSBuild.proj ファイルでコード分析を有効にするには、次の操作を行います。

- プロジェクトの設定に関係なくすべてのプロジェクトでコード分析を実行する場合は、**<RunCodeAnalysis>** タグを **Always** に変更します。
- プロジェクト設定に基づいてプロジェクトごとにコード分析を実行する場合は、<RunCodeAnalysis> タグを Default に変更します。

参考資料

- ビルドの一環として行う自動コード分析の詳細については、このガイドの「Visual Studio Team Foundation Server でチーム ビルドを使用してコード分析を自動的に実行する方法」を参照してください。
- コード分析ツールの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms182023(VS.80).aspx の「コード分析ツールを

使用するためのガイドライン」を参照してください。

ビルドごとに自動テストを実行する

ビルドするたびにビルドの品質に関するフィードバックを自動的に得たいときは、自動テストを実行してください。ビルドに関連付けるテスト リストを作成するためには、Visual Studio Test Edition か Visual Studio Team Suite のいずれかをインストールしておかなければなりません。ビルド サーバーで自動テストを実行するためには、Visual Studio Developer Edition、Visual Studio Test Edition、Visual Studio Team Suite のいずれかをビルド サーバーにインストールしておかなければなりません。

チーム ビルド処理の一環として自動テストを実行するには、以下の操作を行います。

1. ビルドで実行する自動テストを 1 つ以上作成します。
2. テスト マネージャを使用してテスト リストを作成します。
3. そのテスト マネージャを使用して、テスト ビューからテスト リストへテストをドラッグ アンド ドロップすることにより、各テストを新規のテスト リストとしてグループ化します。
4. 新しいチーム ビルドの種類を作成します。
5. 自動テストを実行するチェック ボックスをオンにします。
6. テストとテスト リストとを作成したテスト プロジェクトを選択します。
7. 実行するテスト リストを選択します。

参考資料

- ビルド確認テストを自動的に実行する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms182465(VS.80).aspx の「方法 : ビルド検証テスト (BVT) を構成および実行する」を参照してください。
- Visual Studio Test Edition も VSMDI ファイルもない状態で自動ビルド テストを実行する方法については、http://blogs.msdn.com/buckh/archive/2006/11/04/how-to-run-tests-without-test-metadata-files-and-test-lists-vsmdi-files.aspx の「How to run tests in a build without test

metadata files and test lists (.vsmdi files)」を参照してください。

自動テストに不合格になったときビルドを停止する設定にしてみる

コンパイルにエラーが出た影響でビルドに失敗すると、その不具合を追跡するための作業項目が作成され、失敗したビルドには、失敗したことを示すマークが付きます。しかし、自動テストに不合格になってもビルドは停止しません。テストに不合格になった場合は警告が出ますが、ビルドは継続されます。

自動テストに不合格になった場合、ビルドを停止させることも可能です。自動テストの不具合を追跡する作業項目を自動的に生成することも可能です。

テストに不合格になったときにビルドを停止するには、次の操作を行います。

1. Program Files¥MSBuild¥Microsoft¥VisualStudio¥v8.0¥TeamBuild にある Microsoft.TeamFoundation.Build.targets ファイルを開きます。
2. テストに不合格になったとき停止するチーム ビルドの種類に対応した TFSBuild.proj ファイルを、編集用にチェックアウトしてから開きます。
3. RunTestWithConfiguration ターゲットを Microsoft.TeamFoundation.Build.targets から TFSBuild.proj ファイルの末尾 (</Project> 閉じタグの直前) へとコピーします。
4. ContinueOnError 属性を true から false へ変更します。

   注意 : テスト ツール タスクは 2 つあります。ビルド サーバーでのビルドの動作を変更するだけならば、エンドツーエンド タスクを変更してください。デスクトップ ビルド タスクは、開発者のデスクトップでビルドするときに使用するタスクです。
5. ビルドに失敗したときに作業項目を作成する場合は、</Target> という閉じタグの直前に OnError 要素を追加して RunTestWithConfiguration を変更してください。OnError 要素は以下のようになります。

```
<OnError ExecuteTargets="CreateWorkItem;"/>
```

また、Microsoft.TeamFoundation.Build.targets を直接変更すれば、テストに不合格になったときにチーム ビルドの種類をすべて停止することも可能です。この場合は、チーム ビルドの種類のすべての

動作が変化します。

上に推奨したソリューションは、実装するのは簡単ですが、Visual Studio の今後のバージョンで必ず機能するとは保証されません。Visual Studio のアップグレード後も、機能し続けることが保証されているソリューションを実装したい場合は、http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether-tests-passed-in-team-build.aspx にある Aaron Hallberg のブログ エントリ「Determining Whether Tests Passed in Team Build」を参照してください。

参考資料

- テストが不合格になったとき、作業項目が作成されるようにビルドを設定する詳細については、http://blogs.msdn.com/nagarajp/archive/2005/10/14/481290.aspx の「Create Workitems for Test Failures in TeamBuild」を参照してください。
- Visual Studio のアップグレード後も機能し続けることが保証されているソリューションの詳細については、http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether-tests-passed-in-team-build.aspx の「Determining Whether Tests Passed in Team Build」を参照してください。

作業項目

- **作業項目を使用してビルドの中断を追跡する**

作業項目を使用してビルドの中断を追跡する

チーム ビルドに失敗した場合は、その不具合を追跡する作業項目が自動的に作成されます。この作業項目は、特に指定しない限りは "アクティブ" に割り当てられ、ビルドに失敗したことがわかるようなタイトルが付きます。失敗したビルドを修正して不具合を解決できるように、この作業項目は、担当の開発者またはビルド マネージャに割り当てることが望ましいです。

TFSBuild.proj 内でこの作業項目を定義するビルド タスクは、以下のような形をしています。

```
<!-- Create WorkItem for build failure -->
<CreateNewWorkItem
BuildId="$(BuildNumber)"
Description="$(WorkItemDescription)"
TeamProject="$(TeamProject)"
TeamFoundationServerUrl="$(TeamFoundationServerUrl)"
Title="$(WorkItemTitle)"
WorkItemFieldValues="$(WorkItemFieldValues)"
WorkItemType="$(WorkItemType)"
ContinueOnError="true" />
```

たとえば、特定の開発者に作業項目を割り当てる場合や、緊急度、優先度を設定する場合のように、作成される作業項目をカスタマイズする場合は、WorkItemFieldValues を変更できます。

参考資料

- ビルド失敗時の作業項目をカスタマイズする詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243778(vs.80).aspx の「Team Foundation ビルドのタスク」を参照してください。

ビルドに関する参考資料

- チーム ビルドの全般については、http://msdn2.microsoft.com/ja-jp/library/ms181710(VS.80).aspx の「Team Foundation ビルドの概要」を参照してください。

ガイドライン : プロジェクト管理

索引

区分およびイテレーション

- 区分を使用してトレーサビリティの向上を図る
- イテレーションを使用してプロジェクト内のマイルストーンを示す
- 未割り当てのシナリオとタスクのために別のイテレーションを作成する
- 適切なイテレーション サイクル期間を決定する

チェックイン ポリシー

- チェックイン ポリシーを使用してコードの品質を維持する
- チェックイン ポリシーを使用して、作業項目をチェックインに関連付けることを開発者に義務付ける
- チェックイン ポリシーを作成してコーディングの基準を守らせる
- 開発者がチェックイン ポリシーに従わなかったとき通知を受信できるようにする

プロセス テンプレート

- 軽量プロセスまたは略式プロセスのみを必要とするプロジェクトで作業するときに MSF Agile プロセス テンプレートを使用する
- 正規のプロセスを必要とするプロジェクト、または CMMI 標準への準拠を必要とするプロジェクトで作業するときに、MSF CMMI プロセス テンプレートを使用する
- 最低限のプロセス テンプレートを使用してみる
- 自身のチームのプロセスに合わせて既存のプロセス テンプレートを変更する

セキュリティ グループと権限

- セキュリティ グループを作成して、特定の権限を与える

- チーム メンバを適切なセキュリティ グループに割り当てる

チーム プロジェクト

- アプリケーションのバージョンから別のバージョンへ作業項目やその他資産を移行する場合に、アプリケーションごとに 1 つずつチーム プロジェクトを作成する
- アプリケーションのバージョンごとに新しい作業項目やその他資産で作業を開始する場合に、バージョンごとに 1 つずつチーム プロジェクトを作成する
- プロジェクト資産に関して、必要な権限のみを与える
- 分岐に対応できる構造をしたソース ツリーを作成する

作業項目

- プロジェクトの開始時にシナリオを取り込む
- サービス品質要件を適切に定義する
- シナリオを分割して、管理可能なモジュラー型開発タスクに分ける
- タスクごとに受け入れ基準を設定する
- 要件とタスクとをシナリオにリンクする
- 作業項目のバルク編集に Microsoft Excel を使用する

区分およびイテレーション

- **区分を使用してトレーサビリティの向上を図る**
- **イテレーションを使用してプロジェクト内のマイルストーンを示す**
- **未割り当てのシナリオとタスクのために別のイテレーションを作成する**
- **適切なイテレーション サイクル期間を決定する**

区分を使用してトレーサビリティの向上を図る

プロジェクトのタスク、バグ、要件、その他作業項目を整理しておきたいときは、チーム プロジェク

トで区分を使用してください。各区分に権限を設定すれば、チーム プロジェクトの各部へのアクセスを制限することも可能です。

論理コンポーネントまたは物理コンポーネントを表すときには区分を使用し、特定の機能を表すときには下位区分を作成してください。こうした構造にしておけば、各作業項目を整理しておくのに役に立つばかりでなく、コンポーネント別または機能別でのトレーサビリティの向上にもつながります。

プロジェクト用に区分を作成するには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトをクリックします。
2. [チーム] メニューで、[チーム プロジェクトの設定] を指してから、[区分およびイテレーション] をクリックします。
3. [区分およびイテレーション] ダイアログ ボックスで、[区分] タブをクリックします。
4. [子ノードの追加] ツールバー ボタンをクリックします。
5. 新しいノードを右クリックし、[名前の変更] をクリックし、必要な区分名を入力します。
6. [区分] ノードをクリックします。
7. 手順 2、3、4 を繰り返して、さらに区分を作成し、プロジェクト構造に対応した階層を作成します。

区分を設けると作業項目の権限を分割することができますが、複雑なツリーの場合は、権限の管理に必要な処理が増えてしまうため、複雑すぎる区分構造の作成には気を付けてください。複雑すぎる構造/権限を他のチーム プロジェクトにコピーすると問題になることがあります。

参考資料

- 区分を使用する詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。

イテレーションを使用してプロジェクト内のマイルストーンを示す

アプリケーションの開発の過程において、計画、実装、テストなどの主要な一連の活動を何回繰り返すかを定義するときは、イテレーションを使用してください。この一連の主要な活動が、完成した機能や完成したコンポーネントなど定量化可能な成果と共に、プロジェクトのマイルストーンになるはずです。

イテレーションを作成するには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトをクリックします。
2. [チーム] メニューで、[チーム プロジェクトの設定] を指してから、[区分およびイテレーション] をクリックします。
3. [区分およびイテレーション] ダイアログ ボックスで、[イテレーション] タブをクリックします。
4. [子ノードの追加] ツールバー ボタンをクリックします。
5. 新しいノードを右クリックし、[名前の変更] をクリックし、イテレーションの名前を入力します。
6. [イテレーション] ノードをクリックします。
7. 手順 2、3、4 を繰り返して、プロジェクトに関係のあるイテレーションをさらに作成します。
8. [閉じる] をクリックします。

注意 ：Microsoft® Solutions Framework (MSF) for Agile Software Development (MSF Agile) プロセス テンプレートには、事前定義されたイテレーションが 3 つあります。このイテレーションは、個々の要件に応じて、削除してもかまいません。新しいイテレーションを作成する代わりとして名前を変更しても、そのまま何も変更しなくてもかまいません。

参考資料

- イテレーションを使用する詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。

未割り当てのシナリオとタスクのために別のイテレーションを作成する

まだどのイテレーションにも割り当てられていないシナリオとタスクを、すべて割り当てることのできる別のイテレーションを作成してください。このようなイテレーションを作成しておけば、イテレーシ

ョンの計画を立てるときに、ペンディングになっているシナリオとタスクを簡単に見分けることができます。

別のイテレーションを作成するには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトをクリックします。
2. [チーム] メニューで、[チーム プロジェクト設定] を指してから、[区分およびイテレーション] をクリックします。
3. [区分およびイテレーション] ダイアログ ボックスで、[イテレーション] タブをクリックします。
4. [子ノードの追加] ツールバー ボタンをクリックします。
5. 新しいノードを右クリックし、[名前の変更] をクリックし、イテレーションの名前を「Iteration 999」と入力します。
6. [閉じる] をクリックします。

参考資料

- イテレーションを作成する詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。

適切なイテレーション サイクル期間を決定する

チーム プロジェクトの設定時に、プロジェクトの規模と複雑さに応じて適切なイテレーション サイクル期間を決定してください。

イテレーション サイクル期間を決定するときは、以下の点に注意してください。

- イテレーション サイクルは、チーム メンバが十分な量の作業を行えるだけの長さにすることが望ましく、かつ、少なくともシナリオ数個分に及ぶことが望ましいです。
- イテレーション サイクルは、変更と優先度に柔軟に適応できるだけの短さにすることが望ましいです。

実際には、イテレーション サイクルを 2 週間に設定すれば、大半のプロジェクトでうまくいきます。

参考資料

- イテレーション サイクル期間については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。
- イテレーション サイクル期間の詳細については、このガイドの「第 11 章 プロジェクト管理の説明」を参照してください。

チェックイン ポリシー

- **チェックイン ポリシーを使用して、コードの品質を維持する**
- **チェックイン ポリシーを使用して、作業項目をチェックインに関連付けることを開発者に義務付ける**
- **チェックイン ポリシーを作成して、コーディング基準を守らせる**
- **開発者がチェックイン ポリシーに従わなかったとき通知を受信できるようにする**

チェックイン ポリシーを使用して、コードの品質を維持する

プロジェクトのチェックインの品質を上げたいときは、コード分析とテスト ポリシーとを組み合わせて使用してください。たとえば、特定のテストを実行し、そのテストに合格してからでないと、ソースが Microsoft Visual Studio® 2005 Team Foundation Server (TFS) ソース管理にチェックインされないようにするときは、付属のテスト ポリシーを使用します。また、セキュリティ、パフォーマンス、移植性、メンテナンス性、信頼性の各ルールを確実にすることにより、コードが一定の品質基準を満たせるようなコード分析ポリシーを設定することも可能です。

この種のチェックイン ポリシーを適用するだけでなく、コーディングの基準とガイドラインとを守らせるポリシーも適用することにより、コードが特定の品質基準に至るようになります。

チーム プロジェクトでコード分析チェックイン ポリシーを適用するには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトを右クリックし、[チーム プロジェクトの設定] を指して、[ソース管理] をクリックします。
2. [チェックイン ポリシー] タブをクリックし、[追加] をクリックし、適切なポリシーを選択して設定します。

参考資料

- カスタム チェックイン ポリシーの作成および使用の詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してください。

チェックイン ポリシーを使用して、作業項目をチェックインに関連付けることを開発者に義務付ける

チェックインを作業項目に関連付けることを開発者に義務付けるときは、作業項目チェックイン ポリシーを設定してください。

ビルドに問題が生じた場合は、どの変更セットがビルドに関連付けられているのかを知り、その変更セットがどの作業項目に関連付けられているのかを知ることが大切です。そうすれば、このコードのチェックインをどの開発者が担当しているかがわかり、その開発者がプロジェクトのどの区分で作業しているかがわかります。

作業項目チェックイン ポリシーを設定して、チェックインを作業項目に関連付けることを開発者に義務付けるには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトを右クリックし、[チーム プロジェクトの設定] を指して、[ソース管理] をクリックします。
2. [チェックイン ポリシー] タブをクリックします。
3. [追加] をクリックし、[作業項目] チェックイン ポリシーを選択して設定します。

参考資料

- チェックインの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181411(VS.80).aspx の「方法 : 保留中の変更をチェックインする」を参照してください。
- 作業項目と変更セットの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181410(VS.80).aspx の「方法 : 作業項目を変更セットに関連付ける」を参照してください。

チェックイン ポリシーを作成して、コーディング基準を守らせる

作業しているプロジェクトが、静的なコード分析や既存のチェックイン ポリシーでは対応しきれないコーディング基準を必要とするプロジェクトである場合があります。たとえば、コードにタブ文字を使用してはならないプロジェクトや、すべてのチェックインにコメントを義務付けるプロジェクトなどです。そうしたシナリオにも対応できる新しいチェックインポリシーを作成することができます。

チーム プロジェクトでコード分析チェックイン ポリシーを適用するには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトを右クリックし、[チーム プロジェクトの設定] を指して、[ソース管理] をクリックします。
2. [チェックイン ポリシー] タブをクリックしてから、[追加] をクリックします。
3. [チェックイン ポリシーの追加] ダイアログ ボックスで、[コード分析] を選択し、[OK] をクリックします。
4. [コード分析ポリシー エディタ] で、[C/C++ コード分析 (/analyze) を強制] または [マネージ コードのコード分析を強制] を選択します。マネージ コードとアンマネージ コードの両方がプロジェクトに含まれている場合は、両方とも選択してください。
5. マネージ コード分析を選択した場合は、必要なコーディング基準に応じて、マネージ コード分析に必要なルールを設定します。

   これにより、どのルールが適用されるかが正確に決まります。

既定では利用できないチェックを実行するため、カスタム チェックイン ポリシーを作成することもできます。たとえば、禁止されているアプリケーション プログラミング インターフェイス (API) 呼び出しなどのコード パターンを許可しないようにすることもできます。また、ソース コード内のどこに中かっこを配置すべきかなど、チームごとのコーディング スタイルに関するガイドラインを適用するポリシーを記述することもできます。

参考資料

- チェックイン ポリシーを作成する詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してください。

開発者がチェックイン ポリシーに従わなかったとき通知を受信できるようにする

Team Foundation Server Version Control には、チェックイン ポリシーの無視を防ぐ手段がありません。ただし、以下の手順を実行すれば、チェックイン ポリシーが無視されたかどうかを検出することができます。

1. チェックイン イベントを捕捉するため、(Team Foundation Core Services API から) Team Foundation Server Eventing Service を使用します。
2. 変更セットの詳細を解析して、チェックイン ポリシーが無視された場合にその対応を行う Notify メソッドを記述します。

また、手作業で変更セットの履歴を細かく調べて、ポリシーが無視されたかどうかを見つけることもできます。

参考資料

- チェックイン ポリシーの無視については、http://msdn2.microsoft.com/ja-jp/library/ms245460(VS.80).aspx の「方法 : チェックイン ポリシーをオーバーライドする」を参照してください。

プロセス テンプレート

- **軽量プロセスまたは略式プロセスのみを必要とするプロジェクトで作業するときに、MSF Agile プロセス テンプレートを使用する**
- **正規のプロセスを必要とするプロジェクト、または CMMI 標準への準拠を必要とするプロジェクトで作業するときに、MSF CMMI プロセス テンプレートを使用する**
- **最低限のプロセス テンプレートを使用してみる**
- **自身のチームのプロセスに合わせて既存のプロセス テンプレートを変更する**

軽量プロセスまたは略式プロセスのみを必要とするプロジェクトで作業するときに、MSF Agile プロセス テンプレートを使用する

テストドリブン開発 (TDD)、または他のアジャイル手法を使用している場合は、MSF for Agile Software Development (MSF Agile) プロセス テンプレートを使用することが望ましいです。これは、アジャイル ソフトウェア プロジェクト用の軽量プロセスです。MSF for CMMI Software Development (MSF CMMI) プロセス テンプレートに別途用意されているプロセス改善機能が特に必要な場合を除き、まずはこの MSF Agile プロセス テンプレートを使用してみてください。

MSF Agile プロセス テンプレートは、簡単に変更できるため、個々のプロセス要件に合わせて修正できます。

参考資料

- 詳細については、このガイドの「第 11 章 プロジェクト管理の説明」を参照してください。
- MSF Agile プロセス テンプレートの詳細については、このガイドの「第 14 章 アジャイル ソフトウェア開発プロジェクトに対する MSF」を参照してください。
- プロセス テンプレートをカスタマイズする詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

正規のプロセスを必要とするプロジェクト、または CMMI 標準への準拠を必要とするプロジェクトで作業するときに、MSF CMMI プロセス テンプレートを使用する

既存プロセスの改善を目指した正規のソフトウェア開発プロセスを使用する場合は、MSF for CMMI Software Development (MS CMMI) プロセス テンプレートを使用することが望ましいです。

このプロセス テンプレートは、個々のプロセス要件に合わせて簡単に編集、変更できます。

参考資料

- 詳細については、このガイドの「第 11 章 プロジェクト管理の説明」を参照してください。
- テンプレートをカスタマイズする詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

最低限のプロセス テンプレートを使用してみる

多くのチームは、標準的なチーム プロジェクトのすべての部分をサポートする必要はありません。たとえば、多くのチームは、チーム プロジェクトのソース管理の部分だけが必要であり、Microsoft Office SharePoint® ポータルは不要です。チーム プロジェクト テンプレートは変更できますので、不要な部分は削除できます。テンプレートを変更する場合でも、Group Permissions セクションと Classifications セクションは残しておく必要があります。それ以外のセクションは、必要に応じて、残しておいてもいいし、削除してもかまいません。

最低限のプロセス テンプレートを作成するときは、プロセス テンプレート マネージャを使用して、目的のテンプレートをローカル コンピュータにダウンロードし、使用しないセクションをテンプレートから削除し、そのテンプレートをまたサーバーにアップロードして戻します。

参考資料

- プロセス テンプレートの詳細については、このガイドの「第 13 章 プロセス テンプレートの説明」を参照してください。

- プロセス テンプレートのカスタマイズについては、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。
- プロセス テンプレートをカスタマイズする詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243782(VS.80).aspx の「プロセス テンプレートのカスタマイズ」を参照してください。
- 最低限のプロセス テンプレートを使用する詳細については、http://blogs.msdn.com/richardb/archive/2007/05/10/how-to-use-tfs-for-source-control-only.aspx の「How to use TFS for source control only」を参照してください。

自身のチームのプロセスに合わせて既存のプロセス テンプレートを変更する

Microsoft Visual Studio Team System (VSTS) に付属しているプロセス テンプレートは、そのままでは、作業しているプロジェクトにうまく適合しない場合があります。別の作業項目の種類が必要な場合もあれば、まったく異なるプロセス手法を使用している場合もあります。この場合は、既存のプロセス テンプレートを変更した方がよいです。自身のプロセス要件を最も満たしているプロセス テンプレートを選択して、必要に応じて変更してください。

一般的に、プロセス テンプレートの以下の部分をカスタマイズする必要があります。

- グループと権限
- 作業項目の種類
- ソース管理のチェックイン メモとポリシー
- 区分およびイテレーション
- レポート
- チーム ポータル
- プロセス ガイダンス

参考資料

- プロセス テンプレートの詳細については、このガイドの「第 13 章 プロセス テンプレートの説明」

を参照してください。

- テンプレートをカスタマイズする詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

セキュリティ グループと権限

- **セキュリティ グループを作成して、特定の権限を与える**
- **チーム メンバを適切なセキュリティ グループに割り当てる**

セキュリティ グループを作成して、特定の権限を与える

Team Foundation Server でプロジェクトを作成すると、選択したプロセス テンプレートが何であっても、そのプロジェクト用に 4 つの既定のグループが作成されます。特に指定しない限り、各グループには一連の権限が付与されています。グループのメンバに何ができて何ができないかは、その権限によって決まります。その 4 つのグループは次のとおりです。

- プロジェクト管理者
- 貢献者
- 閲覧者
- ビルド サービス

チーム プロジェクト用にセキュリティ グループを作成した方が、組織のセキュリティ要件を満たせるようになります。セキュリティ グループを作成すれば、チーム プロジェクトにかかわるユーザー グループに一連の権限を効率よく付与することができます。付与する権限は必要最小限とし、追加するユーザーまたはグループは、この新しいチーム プロジェクト グループに所属しなければならないユーザーまたはグループだけに限ってください。

以下のガイドラインにも従ってください。

- 既定のグループに付与された権限は変更しないこと (変更する場合は、すべてのプロジェクトで同じように変更すること)

- メンバシップ用に Active Directory (AD) グループを使用するのはサーバー レベルのみとすること
- 権限の設定には、AD グループではなく TFS グループを使用すること
- 一切何も拒否せず、拒否する場合は、適切な理由があること (通常、拒否するというのは、分割の仕方が理想的ではないということを示している)

参考資料

- セキュリティ グループを作成する詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。
- TFS 権限の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252587(VS.80).aspx の「Team Foundation Server のアクセス許可」を参照してください。

チーム メンバを適切なセキュリティ グループに割り当てる

既存のチーム プロジェクト グループか、サーバー レベル グループか、自身で作成するカスタム セキュリティ グループか、いずれかを使用して、プロジェクトで作業するチーム メンバを選び、そのメンバの役割を決め、各チーム メンバを TFS に割り当ててください。

メンバをセキュリティ グループに割り当てるときは、そのセキュリティ グループで利用できる権限を必要とするメンバだけを割り当ててください。必要であれば、適切なセキュリティ権限を付与したカスタム セキュリティ グループを作成して、そのセキュリティ グループにユーザーを割り当てることも可能です。

参考資料

- セキュリティ グループの詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。

チーム プロジェクト

- **アプリケーションのバージョンから別のバージョンへ作業項目やその他資産を移行する場合に、アプリケーションごとに 1 つずつチーム プロジェクトを作成する**
- **アプリケーションのバージョンごとに新しい作業項目やその他資産で作業を開始する場合に、バージョンごとに 1 つずつチーム プロジェクトを作成する**
- **複数のプロジェクトに及ぶ大規模なプロジェクトで作業するときに、チームごとに 1 つずつチーム プロジェクトを作成する**
- **プロジェクト資産に関して、必要な権限のみを与える**
- **分岐に対応できる構造をしたソース ツリーを作成する**

アプリケーションのバージョンから別のバージョンへ作業項目やその他資産を移行する場合に、アプリケーションごとに 1 つずつチーム プロジェクトを作成する

ソース コードだけでなく、作業項目や、その他の TFS 資産までも、リリースから別のリリースへ移行する場合は、アプリケーションごとに 1 つずつチーム プロジェクトを使用することを検討してください。アプリケーションの複数のバージョンにチーム プロジェクトを 1 つずつ使用していれば、TFS 資産はすべて自動的に次のリリースへと移行されます。アプリケーションの新バージョンをリリースする準備が整ったときには、そのリリースを示す分岐をプロジェクト内に作成して、コードを分離することができます。

アプリケーション 1 つにつきプロジェクトを 1 つ使用する場合は、以下の点に注意してください。

- 並列リリースの場合は、作業項目、チェックイン ポリシー、プロセス ガイダンスを共有しなければなりません。
- レポートの処理が難しくなります。既定では、レポートはプロジェクト全体が対象となるため、リリースごとにフィルタの処理を追加しなければなりません。
- アプリケーションが何百もあり、それぞれが別々のプロジェクトに含まれている場合は、TFS のパフォーマンスと拡張性が限界に達します。
- リリースを繰り返すうち、”荷物” が増えていきます。この問題を解消する一番簡単な方法は、新規プロジェクトを作成して、次のリリースに移行するコードをその新規プロジェクトに分岐することです。

参考資料

- チーム プロジェクトを使用する詳細については、http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx の「When to use Team Projects」を参照してください。

アプリケーションのバージョンごとに新しい作業項目やその他資産で作業を開始する場合に、バージョンごとに 1 つずつチーム プロジェクトを作成する

作業項目や、その他の TFS 資産を移行せずに、リリースごとに 1 からやり直す場合は、リリースごとに 1 つずつプロジェクトを使用する方法を検討してみてください。リリースごとに新規のプロジェクトを使用すると、前のリリースに影響を及ぼさずに、作業項目スキーマ、ワークフロー、チェックイン ポリシー、その他資産を変更することができます。これは場合によっては特に便利です。たとえば、主要開発チームとはプロセスもワークフローも異なる可能性のある維持管理チームのような別のチームが、前のリリースを管理する場合はなおのこと便利です。

リリースごとにプロジェクトを 1 つ使用する場合は、以下の点に注意してください。

- ソース コードをプロジェクトから別のプロジェクトへ移動するのは非常に簡単ですが、作業項目やその他の TFS 資産をプロジェクトから別のプロジェクトへ移動するのは難しいです。作業項目は、一度に 1 つずつしか別のプロジェクトにコピーできないため、いくつかまとめて作業項目をコピーしたい場合は、ユーティリティを作成する必要があります。
- アプリケーションとリリースが何百もあり、それぞれが別々のプロジェクトに含まれている場合は、TFS のパフォーマンスと拡張性が限界に達します。
- 既存のチーム プロジェクトの構造を変えるのは難しいため、長い期間利用できる構造を選んでください。
- ソースは次のようにチーム プロジェクトどうしで簡単に共有できます。
    - プロジェクトから別のプロジェクトへソースを分岐します。
    - 別のプロジェクトからワークスペースへソースをマップします。
- Team Foundation Server で対応できるプロジェクトの個数は、MSF Agile プロセス テンプレートを使用した場合は 500 個、MSF CMMI プロセス テンプレートを使用した場合は 250 個まで拡張できます。自分だけのプロセス テンプレートを作成する場合、または既存のプロセス テンプ

レートをカスタマイズする場合は、サーバーの拡張性に一番大きく影響するのが作業項目スキーマであることを忘れないでください。スキーマが複雑だと、サーバーで対応できるプロジェクトの個数が減ります。

- 元のプロジェクトの区分はすべて維持しなければならず、場合によっては、ソース管理で権限も変更しなければなりません。

参考資料

- チーム プロジェクトを使用する詳細については、http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx の「When to use Team Projects」を参照してください。

プロジェクト資産に関して、必要な権限のみを与える

チーム プロジェクトの作成時には、プロセスによって作成される既定のセキュリティ グループを見直して、必要なら、適切な権限を付与したセキュリティ グループを作成してください。その後で、プロジェクト資産に関して各メンバに必要な権限のみが付与されるよう、プロジェクト メンバを適切なグループに割り当てます。

参考資料

- 権限を付与する詳細に関しては、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。

分岐に対応できる構造をしたソース ツリーを作成する

ソース ツリー構造を作成するときは、分岐に対応した構造にしてください。フォルダは、ソースと、その他のプロジェクト資産とを別にしてください。別にしておけば、分離開発が将来必要になった場合でも、ソース フォルダを分岐するだけで済みます。また、ソース フォルダの中では、コンポーネントごとにフォルダを分けておいてください。そうすれば、必要な場合に部分分岐が実行できます。

他の実体 (共有コード、単体テスト、ライブラリ依存関係など) についても、フォルダを使用して分けておいてください。そうすれば、分岐のときに適宜除外することも含めることもできます。

以下に示したのは、分岐に対応したソース ツリー構造の一例です。

• **Main :** 製品の出荷に必要なすべての資産用のコンテナ

o **Source :** ビルドに必要なすべてのもの用のコンテナ

--□□-□□-　□

Code : ソース コード用のコンテナ

--□□-□□-　□

Shared Code : 他のプロジェクトから共有されるソース コード用のコンテナ

--□□-□□-　□

Unit Tests : 単体テスト用のコンテナ

--□□-□□-　□

Lib : バイナリ依存関係用のコンテナ

o **Docs :** 製品と一緒に出荷する文書用のコンテナ

o **Installer :** インストーラのソース コードおよびバイナリ用のコンテナ

o **Builds :** チーム ビルド スクリプト用のコンテナ

o **Tests :** テスト チームのテスト ケース用のコンテナ

参考資料

• ソース ツリー構造の詳細に関しては、このガイドの「第 5 章 分岐とマージの方針の定義」を参照してください。

作業項目

- **プロジェクトの開始時にシナリオを取り込む**
- **サービス品質要件を適切に定義する**
- **シナリオを分割して、管理可能なモジュラー型開発タスクに分ける**

- **タスクごとに受け入れ基準を設定する**
- **要件とタスクとをシナリオにリンクする**
- **作業項目のバルク編集に Microsoft Excel を使用する**

プロジェクトの開始時にシナリオを取り込む

プロジェクトの開始時に、一連のプロジェクト シナリオを作成して取り込んでください。そうすれば、プロジェクトの全体像がつかみやすくなり、後で進捗を追うときにも使用できます。開発の過程でも、既存のシナリオを変更したり新しいシナリオ追加したりするなどして、学んだことを反映することができます。

プロジェクトの開始時にシナリオを取り込むには、次の操作を行います。

1. さまざまなステークホルダ (顧客、ビジネス アナリスト、エンド ユーザー、プロジェクト マネージャなど) から得た情報を基に作成した要件文書であるプロジェクト バック ログ (PBL) 文書を使用し、プロジェクトのシナリオをよく調べます。
2. [チーム エクスプローラ] で、プロジェクト ノードを展開し、[作業項目] フォルダを右クリックし、[作業項目の追加] をポイントし、[シナリオ] をクリックします。
3. [新しいシナリオ] ページで、シナリオの詳細を入力します。イテレーションが「Iteration 999」に必ず設定されるようにします。
4. 新しいシナリオを保存します。
5. プロジェクトに関係するすべてのシナリオに対して、上記の各手順を繰り返します。

参考資料

- シナリオを取り込む詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。

サービス品質要件を適切に定義する

イテレーション サイクルの間に処理する個々のシナリオについて、サービス品質 (QoS) 要件を定義してください。そうすれば、シナリオの受け入れ基準を定義するときに便利です。QoS は、プロジェクトの目標と要件によって決まります。仕様書がある場合は、それも QoS 要件を決めるデータとなります。

QoS 要件を定義するには、次の操作を行います。

1. プロジェクト内の [作業項目] フォルダを右クリックし、[作業項目の追加] を指してから、[サービス品質要件] をクリックします。
2. [新しいサービス品質要件] ページで、次の詳細情報を追加します。
    a. [種類] を、パフォーマンス、スケーラビリティ、ストレス、セキュリティなど該当する値に設定します。
    b. [イテレーション] を現在のイテレーション サイクルに設定します。
    c. [リンク] タブから QoS を特定のシナリオにリンクし、追跡しやすいようにします。
3. 新しい QoS 要件を保存します。
4. どのシナリオにも QoS 要件が複数ありうることを忘れずに、品質要件の分野ごとまたは種類ごとに QoS 要件を 1 つずつ作成します。
5. 特定のイテレーション サイクルの間に処理するすべてのシナリオに関して、QoS 要件を作成してください。

重要事項 : QoS 要件は、後でテスト タスクに分割できます。

参考資料

• QoS 要件を定義する詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。
• 作業項目の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181314(VS.80).aspx の「Team Foundation の作業項目の管理」を参照してください。

シナリオを分割して、管理可能なモジュラー型開発タスクに分ける

イテレーション計画の最中に、シナリオをいくつかのユーザー ストーリーに分割し、さらにそのユーザー ストーリーをいくつかの開発タスクに分割してください。作成する開発タスクは、管理可能なモジュール型のタスクにしてください。このタスクは、数日以上は続かないはずです。これよりも大きなタスクの場合は、もっと小さなタスク (サブタスク) に分割する必要があります。そうすれば、プロジェクトのスケジュールを組む自由度が上がり、プロジェクトの管理もしやすくなります。

シナリオを分割して、管理可能なモジュラー型開発タスクに分けるには、次の操作を行います。

1. 選択したシナリオをいくつかの開発者ストーリーに分割します。
2. その開発者ストーリーをさらにいくつかの開発者タスクに分割します。
3. 以下の手順により、開発者タスクをタスク作業項目として TFS に取り込みます。
   a. [チーム エクスプローラ] で、自身のプロジェクト ノードを展開し、[作業項目] フォルダを右クリックし、[作業項目の追加] をポイントし、[タスク] をクリックします。
   b. [新しいタスク] ページで、以下のように詳細情報を追加します。
      i. [作業分野] を [開発] に設定します。
      ii. ii. [イテレーション] を現在のイテレーション サイクルに設定します。
      iii. [リンク] タブで、タスクを特定のシナリオにリンクし、追跡しやすいようにします。 このタブでは、説明が取り込めるほかに、タスクの受け入れ基準も取り込めるので、タスクが正常に完了したかどうかが判定できます。
      iv. [担当者] フィールドを、タスクに従事する開発者に設定します。
   c. 新しいタスクを保存します。
   d. 関係のあるすべてのタスクに対して、上記の各手順を繰り返します。
4. 当該イテレーションに関係するすべてのシナリオに対して、上記の各手順を繰り返します。

参考資料

- シナリオの詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。
- 作業項目の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181314(VS.80).aspx

の「Team Foundation の作業項目の管理」を参照してください。

タスクごとに受け入れ基準を設定する

開発タスクを定義するときには、いつタスクが完了したかを判断する基準となる受け入れ基準をその開発タスクに設定しなければなりません。これには 2 つの方法があります。使用しているプロセス テンプレートによって使い分けてください。

- **MSF Agile :** 作業項目タイプの正規の要件を伴わない MSF Agile を使用している場合は、受け入れ基準をテキストとして作業項目自体の中に埋め込むのが最善の策です。最初は箇条書きから始めて、適宜詳細情報を追加してください。
- **MSF CMMI :** MSF CMMI を使用している場合は、正規の要件を使用してタスクの受け入れ基準を定義することができます。最初の手順は、要件を定義することです。その後、各要件を実装するのに使用される開発タスクを作成し、そのタスクを各要件にリンクします。そうすれば、各要件に照らしてチェックすることができ、要件からタスクへとたどることもできるようになります。

受け入れ基準は、ミニシナリオか QoS 要件かいずれかの形式でユーザー経験要件として定義されることが最も多いです。開発者は、受け入れ基準を満たしていれば、後はそのタスクに完了のマークを付けて、次のタスクへ進むことができます。

参考資料

- 作業項目の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181314(VS.80).aspx の「Team Foundation の作業項目の管理」を参照してください。

要件とタスクをシナリオにリンクする

タスク、バグ、問題点、QoS 要件などの新しい作業項目を作成するときは必ず、その作業項目の作成を主導したシナリオと、その作業項目を関連付けてください。そうすれば、開発イテレーションの過程で、どの作業項目も本当のユーザー シナリオによって主導されるようになり、作業項目を使用してシナリオの進捗状況をより良く把握できるようになります。

タスク、バグ、問題点、QoS など新しい作業項目をシナリオにリンクするには、次の操作を行います。

1. [新しい作業項目] ページで、[リンク] タブをクリックし、[リンク] タブの [追加] ボタンをクリックします。
2. [リンクの追加] ダイアログ ボックスの [リンクの種類] にある [シナリオ] を選択します。
3. [参照] をクリックして、チーム プロジェクトに含まれているシナリオを見つけます。
4. リストの中から、リンク先となるシナリオを選択し、[OK] をクリックします。
5. [コメント] ボックスに、作業項目との関係について述べたコメントを入力します。自動的に [説明] ボックスに入力されます。
6. [**OK**] をクリックします。

参考資料

- 詳細については、このガイドの「Visual Studio Team Foundation Server でプロジェクトを管理する方法」を参照してください。
- 作業項目の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181314(VS.80).aspx の「Team Foundation の作業項目の管理」を参照してください。

作業項目のバルク編集に Microsoft Excel を使用する

Team Foundation Server では、作業項目のバルク編集はできません。そのため、各作業項目を 1 つずつ編集しなければなりません。たとえばトリアージ ミーティングの最中のように短期間に大量の作業項目を編集する必要がある場合は、その編集作業が楽になるよう Microsoft Office Excel® の使用を検討してください。作業項目は、TFS から Excel へエクスポートして、Excel で編集し、その編集内容を維持したまま TFS にインポートし直すことができます。

Excel で作業項目リストを作成して編集するには、次の操作を行います。

1. Microsoft Office Excel の [チーム] メニューで [新しい一覧] をクリックします。

2. [Team Foundation Server に接続] で、接続先のサーバーを選択するか [サーバー] をクリックしてサーバー情報を入力するか、いずれかの操作をします。
3. [チーム プロジェクト] で、Team Foundation Server 上にあるチーム プロジェクトの中から、作業をするチーム プロジェクトを選択します。
   ドキュメントがこのチーム プロジェクトにバインドされます。
4. [**OK**] をクリックします。
5. 必要なリストの種類を選択します。クエリ リストを作成するため、[クエリ リスト] オプションを選択してから、[クエリの選択] ドロップダウン リストからチーム クエリを選択します。
6. 新しい作業項目リストに表示させる列を選択します。
7. 目的の作業項目をインポートします。詳細については、http://msdn2.microsoft.com/en-us/library/ms181676(VS.80).aspx の「How to: Import Work Items in Microsoft Excel or Microsoft Project」を参照してください。
8. これで、作業項目が編集できるようになりました。[チーム] メニューの [変更の発行] をクリックすれば、更新した作業項目から作業項目データベースへ発行できます。

参考資料

- プロジェクト管理タスクに Microsoft Office Excel を使用する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181694(VS.80).aspx の「Microsoft Excel で作業項目リストを操作する」を参照してください。

Team Foundation プロジェクト管理に関する参考資料

- MSF プロセス テンプレートの詳細については、http://msdn2.microsoft.com/en-us/teamsystem/aa718801.aspx の「Process Templates」を参照してください。

ガイドライン : レポート

索引

管理

- 適切なセキュリティ グループにユーザーを所属させる
- プロジェクトの状況と健全性指標の両方を 1 か所で閲覧できるよう、レポート ダッシュボードを作成する

作成/カスタマイズ

- レポートを配置する際にサーバー名が正しいことを確認する
- 一定期間の動向をつかめるスケジュール レポート スナップショットを作成する
- 補足データにアクセスできるよう既存のレポートを変更する

閲覧

- 最新データが必要な場合に、ウェアハウス Web サービスの実行が必ず済んでいるようにする

管理

- **適切なセキュリティ グループにユーザーを所属させる**
- **プロジェクトの状況と健全性指標の両方を 1 か所で閲覧できるようレポート ダッシュボードを作成する**

適切なセキュリティ グループにユーザーを所属させる

ユーザーがレポートを配置できるようにする場合は、そのユーザーにレポート サーバーのコンテンツ管理者ロールを割り当ててください。コンテンツ管理者ロールというのは、レポートとデータソース接続の配置、管理をWeb サーバー上で行うユーザー用にあらかじめ定義してある Report Services ロールのことです。Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) のユーザーからレポート サーバーに宛ててレポートを配置するには、そのユーザーがコンテンツ管理者ロールのメン

バでなければなりません。

ユーザーをコンテンツ管理者ロールに追加するには、次の操作を行います。

1. プロジェクトのレポート サイトを開きます。[チーム エクスプローラ] で、チーム プロジェクトの [レポート] エントリを右クリックし、[レポート サイトの表示] をクリックします。
2. ウィンドウの一番上にある [プロパティ] タブをクリックします。
3. ウィンドウの左側にある [セキュリティ] をクリックします。
4. [新しいロールの割り当て] をクリックします。
5. コンテンツ管理者ロールに追加するユーザーまたはグループの名前を、[グループ名またはユーザー名] フィールドに入力します。
6. [コンテンツ マネージャ] チェック ボックスをオンにします。
7. [OK] をクリックします。

参考資料

- コンテンツ管理者ロールの詳細については、http://technet.microsoft.com/ja-jp/library/ms159693.aspx の「コンテンツ マネージャ ロール」を参照してください。
- データ層でのセキュリティ ロールの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms174839.aspx の「Analysis Services を使用したアクセスのセキュリティ保護」を参照してください。
- アプリケーション層でのセキュリティ ロールの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms157198.aspx の「Reporting Services の保護」を参照してください。

プロジェクトの状況と健全性指標の両方を 1 か所で閲覧できるようレポート ダッシュボードを作成する

レポート ダッシュボードがあると、1 つのページからプロジェクトの重要な情報へすばやくアクセス

できます。MSF for Agile Software Development (Microsoft Solution Framework for Agile Software Development) プロジェクト用の Microsoft Office SharePoint® のデフォルトのポータル ページには、レポートが 1 つあるほか、リンクがいくつか張ってあります。MS Agile プロジェクト用または MSF for CMMI® (MS CMMI) プロジェクト用のポータル ページを変更して、必要な個数だけレポートが表示できるようにすれば、プロジェクト情報が 1 か所で閲覧できるようになります。

たとえばレポート ダッシュボードには、以下のようなレポートを表示するのが便利です。

- 残存作業
- 品質指標
- バグ率
- プロジェクト速度

表示したいレポートごとにレポート ビューア Web パーツを追加すれば、SharePoint のポータル ページに新しいレポートが追加できます。

チーム プロジェクト ポータルを変更し、レポート ダッシュボードを作成するには、次の操作を行います。

1. stsadm.exe ツールと RSWebParts.cab とを使用してレポート ビューア Web パーツをレポート サーバーにインストールします。stsadm.exe ツールと RSWebParts.cab は、SharePoint および Report Services のインストール パッケージに付属しています。
    - STSADM.EXE は、C:¥Program Files¥Common Files¥Microsoft Shared¥web server extensions¥60¥BIN にあります。
    - RSWebParts.Cab は、C:¥ Program Files¥Microsoft SQL Server¥90¥Tools¥Reporting Services¥SharePoint にあります。

    例 ：STSADM.EXE -o addwppack -filename "C:¥ Program Files¥Microsoft SQL Server¥90¥Tools¥Reporting Services¥SharePoint¥RSWebParts.cab" -globalinstall
2. [チーム エクスプローラ] で、プロジェクトを右クリックします。
3. [プロジェクト ポータルの表示] をクリックします。

4. [共有ページの変更] をクリックします。
5. [参照] をポイントし、[Web パーツの追加] をクリックします。
6. [仮想サーバー ギャラリー] をクリックします。
7. [Web パーツの一覧] で [Report Viewer] を選択します。
8. [追加] をクリックします。
9. 「http://<*report server*>/reports」のようにレポート マネージャ名を入力します。
10. 「<*my project*>/Quality Indicators」のように、表示するレポートのパスを入力します。

参考資料

- レポート ビューア Web パーツを追加する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms159772.aspx の「SharePoint 2.0 Web パーツによるレポートの表示」を参照してください。
- チーム プロジェクト ポータルの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms242883(VS.80).aspx の「チーム プロジェクト ポータルの使用」を参照してください。

作成/カスタマイズ

- **レポートを配置する際にサーバー名が正しいことを確認する**
- **一定期間の動向をつかめるスケジュール レポート スナップショットを作成する**
- **補足データにアクセスできるよう既存のレポートを変更する**

レポートを配置する際にサーバー名が正しいことを確認する

レポート サーバーの URL またはターゲット フォルダ名の指定が間違っていると、レポートをレポート サーバーに配置することができません。Visual Studio 2005 からレポートを配置する場合は、レポートの配置先となるサーバーの URL を指定し、さらにそのレポートがどのチーム プロジェクトのレポートであるかを指定してください。レポートの配置先となるレポート サーバーの URL は、http://TeamServerName/**ReportServer** です。この **ReportServer** は、レポート サーバー Web サービスのエンドポイントです。

[配置プロパティ] ダイアログ ボックスの [TargetReportFolder] フィールドで、チーム プロジェクト名を指定してください。この値は大文字と小文字が別の文字として扱われるため、大文字、小文字を間違えてしまうとレポートは配置されますが、[チーム エクスプローラ] を開いても、チーム プロジェクトのレポート一覧にそのレポートは表示されません。

参考資料

- 配置プロパティを設定する詳細については、http://technet.microsoft.com/ja-jp/library/ms155802.aspx の「配置プロパティを設定する方法 (レポート デザイナ)を参照してください。

一定期間の動向をつかめるスケジュール レポート スナップショットを作成する

定期的にプロジェクト データのスナップショットを作成するときは、レポート履歴を使用してください。一定の期間にわたるこうしたスナップショットを見た方が、傾向がよくわかり、プロジェクトの期間全体での重要なデータ ポイントを見逃さずに済みます。

スケジュール レポート スナップショットを作成するには、次の操作を行います。

1. レポート ポータルからレポートを開きます。
2. [プロパティ] タブをクリックします。
3. [履歴] リンクをクリックします。
4. スナップショットを実行するスケジュールを設定します。

スケジュールの設定が済んだ後は、当該レポートの [履歴] タブに各スナップショットが表示されます。[履歴] タブでは、マニュアル スナップショットも作成できます。

補足データにアクセスできるよう既存のレポートを変更する

Visual Studio (Business Intelligence Development Studio) に入っている Microsoft SQL Server™

2005 Reporting Services デザイナを使用してレポートを変更してください。この Reporting Services デザイナは、SQL Server 2005 の各種クライアント ツールに同梱されています。

レポートをカスタマイズすると、新しいレポートを作成しなくても既存のレポートに機能が追加できます。必要なレポートに似ているレポートが既に存在しているのであれば、似ているレポートをカスタマイズしたほうが、新しく作成するよりも時間の節約になります。既存のレポートをカスタマイズするためには、そのレポートをレポート サーバーからエクスポートし、Visual Studio 内の既存のレポート プロジェクトに追加してから、変更を加えた後でレポート ポータルに配置し直さなければなりません。

注意 : チーム レポート サイトから利用できる Report Builder を使用することも可能ですが、このツールは Visual Studio レポート シナリオとの対応が不十分なので、推奨しません。

参考資料

- 詳細な方法の項目については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。
- レポートの詳細については、このガイドの「第 15 章 レポーティングの説明」を参照してください。
- レポート プロジェクトのしくみを解説した詳しいチュートリアルについては、http://msdn2.microsoft.com/ja-jp/library/ms170246.aspx の「Reporting Services のチュートリアル」を参照してください。
- レポートの編集に関する Microsoft MSDN® の記事を読みたいときは、http://msdn2.microsoft.com/ja-jp/library/ms244655(VS.80).aspx の「方法 : レポート デザイナでレポートを編集する」を参照してください。

閲覧

- **最新データが必要な場合にウェアハウス Web サービスの実行が必ず済んでいるようにする**

最新データが必要な場合にウェアハウス Web サービスの実行が必ず済んでいるようにする

レポートのデータを必ず最新のデータにしたい場合は、ウェアハウス Web サービスを手動で実行してください。既定では、ウェアハウス Web サービスは 1 時間に 1 回稼働してレポートのデータを生成します。今レポートを実行しようとしていて、そのレポートに最新のデータを載せたい場合は、ウェアハウス Web サービスを手動で実行することもできます。

ウェアハウス サービスを手動で実行するには、次の操作を行います。

1. インターネット インフォメーション サービス (IIS) マネージャを開きます。
2. [Team Foundation Server] Web サイトを選択します。
3. この Web サイトの中で、Warehouse¥v1.0 ディレクトリを開きます。
   すると、ウェアハウスで利用できる操作リストが表示されたページが開きます。
4. warehousecontroller.asmx を右クリックし、[参照] をクリックします。
5. [Run] をクリックし、[起動] をクリックします。
   すると、実行要求の状況を示したブラウザ ウィンドウがもう 1 つ開きます。値は true になっているはずです。
6. 最初のブラウザ ウィンドウに戻って、各種操作が表示されたページに戻ります。
7. [GetwareHouseStatus] を選択し、[起動] をクリックします。

すると、ウェアハウス Web サービスの現在の状況が表示されます。値が idle になっていれば、ウェアハウスの実行が済んだという意味です。他の値はサービスの状況を示しています。

参考資料

- ウェアハウスのトラブルシューティングについては、http://msdn2.microsoft.com/ja-jp/library/ms244674(vs.80).aspx の「データ ウェアハウスのトラブルシューティング」を参照してください。

Team Foundation のレポートに関する参考資料

- レポートの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194922(VS.80).aspx の「Team Foundation Server のレポート」を参照してください。

ガイドライン ： ソース管理

索引

バージョン管理へのアクセス

- コマンド ライン ツールを使用してみる
- Microsoft® Visual Studio® 2005 Team Foundation Power Tools (TFPT) を使用して変更をアンシェルブする
- Team Foundation Power Tools を使用して変更をロールバックする
- Team Foundation Power Tools を使用してオフラインで作業する
- Team Foundation Power Tools を使用して変更セットを取得する
- Team Foundation Power Tools を使用して、保留中の編集を削除する

管理

- 引き継いだ権限をメンテナンス分岐上でオフにする
- ソースの変更をまだ任せられない開発者には、チェックイン権限を利用させない

分岐/ラベル/マージ

- ビルドにラベルでマークを付けて、後でそのビルドに戻れるようにする
- 分岐を使用して対応リリースを分離する
- マージ パスに沿って分岐構造を計画する
- 構成ファイルやソース ファイルが含まれる高いレベルで分岐する
- あまり深くまで分岐しない
- 必要にならない限り分岐しない
- 基点のないマージは、できる限り避ける
- “チェリーピック” マージよりフル マージを推奨
- 頻繁にマージする

- メイン分岐としての機能が果たせるよう、新規のチーム プロジェクトの場合は必ず最上位フォルダを作成する
- マージの前に二重チェックができるよう、candidate スイッチまたは preview スイッチを使用してみる
- マージの処理に名前変更が含まれる場合は、ツールから推奨されるパスによく注意する
- マージの競合を解決する際には気を付ける
- マージの結果は一度に 1 つずつチェックインする
- マージからチェックインまでの間にビルドしテストを実行する

チェックインとチェックイン ポリシー

- コードを共有する準備が整ったときのみコードをチェックインする
- シェルブセットを使用して、保留中の変更をバックアップまたは共有する
- チェックインのたびに 1 つずつ作業項目を解決する
- チェックイン ポリシーを使用してコーディング基準を守らせる
- チェックイン ポリシーを使用してコード品質ゲートを守らせる
- いつポリシーが無視されたかを検出する
- 競合を避けるよう計画する

チェックアウト、取得、およびロック

- 変更する前に最新のソースを取得する
- lock コマンドは慎重に使用する
- いつファイルをロックするかをチームメイトに伝える

依存関係

- できる限りプロジェクト参照を使用する
- ファイル参照は、どうしても必要な場合のみ使用する
- プロジェクト参照およびファイル参照に copy local = true を使用する

- Web サービスを参照する際、動的 URL を使用する

分散開発/遠隔地開発

- プロキシには、適切なサイズのディスク ドライブを利用する
- 定期的に最新のファイルが取得できるようスケジュール タスクを作成する
- プロキシ パフォーマンス カウンタとイベント ログを定期的に監視する
- ファイル サイズと帯域幅に従って ***executionTimeout*** を設定する
- プロキシが長期間機能停止する場合は、そのプロキシを使用不可にする
- 不要なファイル転送を減らすためワークスペース クローキングを利用してみる

移行

- VSS コンバータを使用して Team Foundation Server ソース管理へ移行する
- 他のソース管理システムから Team Foundation Server ソース管理へ移行する

プロジェクト/ワークスペースの管理

- 分岐ではなくワークスペースを使用して 1 人の開発者を分離する
- Microsoft Windows® エクスプローラではなくソース管理を使用してファイルの削除と名前変更を行う
- ソリューションが開いた状態でのみ削除と名前変更を行う
- アプリケーションのバージョンから別のバージョンへ資産を移動する場合に、アプリケーションごとに 1 つずつチーム プロジェクトを作成する
- アプリケーションのバージョンごとに 1 からやり直す場合に、バージョンごとに 1 つずつチームプロジェクトを作成する
- 統合テストを必要とするコードとバイナリを共有するために分岐を使用する
- プロジェクトどうしの依存関係に対応できるよう、ワークスペース マッピングは避ける
- チーム プロジェクトのルート レベルでワークスペース マッピングを作成する
- 共有コンピュータ上で、他と重複しないローカル フォルダ パスを使用する

- ソース ツリーの一部だけをマッピングしてみる
- 分岐に対応できる構造をしたソース ツリーを作成する

シェルブ

- シェルブを使用して、保留中の変更を共有し、レビューまたはハンドオフできるようにする
- シェルブを使用して、保留中の変更をサーバーにバックアップする
- 優先度の高い作業によって中断された場合にシェルブを使用する

バージョン管理へのアクセス

- **コマンド ライン ツールを使用してみる**
- **Team Foundation Power Tools を使用して変更をアンシェルブする**
- **Team Foundation Power Tools を使用して変更をロールバックする**
- **Team Foundation Power Tools を使用してオフラインで作業する**
- **Team Foundation Power Tools を使用して変更セットを取得する**
- **Team Foundation Power Tools を使用して保留中の編集を削除する**

コマンドライン ツールを使用してみる

Visual Studio から実行できない処理を行う場合や、処理のスケジュールを組む必要がある場合は、Team Foundation Server (TFS) に付属している Team Foundation Power Tools (Tfpt.exe) などのコマンド ライン ツールの使用を検討してください。この Tfpt.exe ツールは、ダウンロードして入手することもできます。こうしたコマンドライン ツールでは、Windows タスク スケジューラを使用して処理のスケジュールを組むことができます。

パスとその他環境変数を正しく設定するため、Visual Studio 2005 のコマンド プロンプト ウィンドウから、Tf.exe または Vsvars32 バッチ ファイルのいずれかを実行してください (Vsvars32 バッチ ファイルは、通常は *DriveLetter*:¥Program Files¥Microsoft Visual Studio 8¥Common7¥Tools にあります)。Tf.exe ツールは、Checkin、Checkout、Get、History、Shelve、Branch、Merge、Label、

Status、Undelete、Undo などほとんどのソース管理コマンドに対応しています。

以下のような処理は、Tf.exe を使用してコマンド ラインから実行することが多いです。

- サーバーとローカル マシンでファイルの同期をとる ：**tf get**
- ファイルをサーバーに追加する ：**tf add**
- 編集できるようファイルをチェックアウトする ：**tf checkout**
- 保留中の変更をチェックインする ：**tf checkin**
- 特定の変更セットをサーバーから取得する ：**tf get /version**

以下のような処理は、コマンド ラインでしか実行できません。

- 別のユーザーのワークスペースを削除する ：**tf workspace /delete**
- 別のユーザーのチェックインを元に戻す ：**tf undo**
- 別のユーザーのロックを解除する ：**tf lock**
- ラベル範囲を定義する ：**tf label**
- 基点のないマージを実行する ：**tf merge**

参考資料

- 詳細については、Microsoft MSDN® Web サイトの http://msdn2.microsoft.com/ja-jp/library/zthc5x3f(VS.80).aspx の「チュートリアル ： コマンド ラインからの Team Foundation ソース管理の操作」を参照してください。

- コマンド ラインでしか利用できないコマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194957(VS.80).aspx の「コマンド ラインからのみ実行できる操作 (Team Foundation ソース管理)」を参照してください。

Team Foundation Power Tools を使用して変更をアンシェルブする

Team Foundation Power Tools (TFPT) には、Visual Studio では利用できないバージョン管理機能があります。TFPT は、たとえばオフラインの作業に便利です。また、変更セットのチェックインを元

に戻す手段として、TFPT を使用してロールバック処理を実行することもできます。変更をアンシェルブする必要のある場合は TFPT の使用を検討してみてください。

TFS でサポートされているアンシェルブの処理では、シェルブされた変更とローカルな変更を一緒にマージすることはできません。保留中の変更 (編集) がローカル ワークスペース内の項目に含まれていて、かつシェルブされた変更も編集である場合は、TFPT を使用して変更セットから変更をアンシェルブすることにより、3 方向マージを用いて TFPT で各変更をマージすることができます。

このコマンドは、Tfpt.exe を使用してコマンド ラインから実行します。

参考資料

• Team Foundation Power Tools をダウンロードするときは、http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。
• Team Foundation Power Tools について意見を交わしているフォーラムを見たい場合は、http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1 にアクセスしてください。

Team Foundation Power Tools を使用して変更をロールバックする

変更をロールバックする必要がある場合は、TFPT の使用を検討してみてください。TFS は、変更セットのチェックインを元に戻す機能には直接は対応していません。TFPT の rollback コマンドを使用すると、指定の変更セットで行われた変更の大半は元に戻すことができます。すべての変更がロールバックできるわけではありませんが、ほとんどのシナリオでロールバックが動作します。

このコマンドは、Tfpt.exe を使用してコマンド ラインから実行します。

参考資料

• Team Foundation Power Tools をダウンロードするときは、
http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。
• Team Foundation Power Tools について意見を交わしているフォーラムを見たい場合は、
http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1 にアクセスしてください。

Team Foundation Power Tools を使用してオフラインで作業する

TFS は、オフラインでの作業にネイティブ対応していません。オフラインで作業する場合は、正確に以下のワークフローに従ってください。

1. 手作業で読み取り専用フラグを外します。
2. ファイルを編集します。
3. ファイルを追加または削除します。
4. TFPT online コマンドを実行します。

上の各手順について以下に詳しく説明します。

重要事項 : オフライン中はどのファイルも名前を変更してはいけません。

1. **手作業で読み取り専用フラグを外します。**
   既定では、ワークスペース内のファイルのうちまだチェックアウトの済んでいないファイルにはすべて、読み取り専用というマークが付いています。サーバーに接続せずに作業する場合は、その読み取り専用というフラグを手作業で外さないと、各ファイルの編集も削除もできません。読み取り専用のフラグを外すときは、Windows エクスプローラで目的のファイルを右クリックし、[プロパティ] をクリックし、[読み取り専用] チェックボックスをオフにし、[OK] をクリックしてください。attrib -r という DOS コマンドを使用しても同じことができます。
2. **ファイルを編集します。**

読み取り専用フラグを外したファイルは編集できます。

3. **ファイルを追加または削除します。**

読み取り専用フラグを外したファイルは追加も削除もできます。TFPT online ツールでは名前変更の処理と追加/削除の処理が区別できないため、ファイルの名前は変更しないでください。

注意 ：Tfpt online コマンドで削除を行う場合はオプションを指定する必要がありますが、処理に時間がかかります。

4. **TFPT online コマンドを実行します。**

オンラインに戻って、コマンド ラインに「TFPT online」と入力して TFPT online コマンドを実行してください。このコマンドを実行すると、書き込み可能なファイルがあるかワークスペースが検索され、どの変更をサーバー上で保留にした方がよいかが決定されます。ファイルを削除している場合は /delete スイッチを使用してください。/delete スイッチを使用した場合は、ローカル ワークスペースに含まれている削除ファイルも検索されます。その後、どの変更をワークスペースに保留するかを選択できるオンライン ウィンドウが表示されます。

参考資料

• Team Foundation Power Tools をダウンロードするときは、http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。

• Team Foundation Power Tools について意見を交わしているフォーラムを見たい場合は、http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1 にアクセスしてください。

Team Foundation Power Tools を使用して変更セットを取得する

変更セットを取得する必要がある場合は TFPT の使用を検討してみてください。TFPT の GetCS コマンドを実行すると、変更セットの当該バージョンの時点で変更セットに列挙されていた項目がすべて取得できます。これは、自身のワークスペースに必要な変更が既に同僚によってチェックインされていた場合は便利ですが、自身のワークスペース全体を最新バージョンに更新することはできません。

このコマンドは、Tfpt.exe を使用してコマンド ラインから実行します。

参考資料

• Team Foundation Power Tools をダウンロードするときは、http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。

• Team Foundation Power Tools について意見を交わしているフォーラムを見たい場合は、http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1 にアクセスしてください。

Team Foundation Power Tools を使用して、保留中の編集を削除する

保留中の編集をファイルから削除する必要がある場合は、TFPT の使用を検討してください。TFPT の Undo Unchanged コマンドを実行すると、実際にはまだ編集されていない保留中の編集がファイルから削除されます。編集用にファイルを大量にチェックアウトしておいて、実際には少数のファイルしか変更しないような場合は、この方法が便利です。TFPT UU ツールを実行することにより、まだ変更されていないファイル上での編集を元に戻すことができます。このツールは、ローカル ワークスペースに含まれているファイルのハッシュと、サーバー上にあるハッシュとを比較して、ファイルが実際に編集されたのかどうかを判定するツールです。

このコマンドは、Tfpt.exe を使用してコマンド ラインから実行します。

参考資料

• Team Foundation Power Tools をダウンロードするときは、http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。

• Team Foundation Power Tools について意見を交わしているフォーラムを見たい場合は、http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1 にアクセスしてください。

管理

- **引き継いだ権限をメンテナンス分岐上でオフにする**
- **ソースの変更をまだ任せられない開発者にはチェックイン権限を利用させない**

引き継いだ権限をメンテナンス分岐上でオフにする

たとえばソフトウェアのいずれかのバージョンを出荷した後のように、分岐がメンテナンスの状態にあるときは、引き継いだ権限をオフにしてツリーをロックすることができます。この操作が済んだ後は、ホット フィックスの必要に応じて個々のユーザーに PendChange 権限と Checkin 権限を付与することができます。

参考資料

- 権限を削除する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400718(VS.80).aspx の「方法 : ソース管理対象のファイルへのアクセスを削除する」を参照してください。

ソースの変更をまだ任せられない開発者にはチェックイン権限を利用させない

新人や研修生のように、まだ確固とした信頼の置けない開発者には、ソース ツリーのチェックイン権限を利用させないことができます。引き継いだ権限をオフにする前に、自身のアカウントに関する権限をはじめとして必要な権限は必ず設定するようにしてください。対象となる開発者は、直接チェックインはできなくても、保留中の変更は可能ですし、そうした変更をシェルブすることもできます。もっと経験を積んだ開発者であれば、そうした変更をアンシェルブして、内容をよく調べて、チェックインすることができます。

参考資料

- 権限を削除する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400718(VS.80).aspx の「方法 : ソース管理対象のファイルへのアクセスを削除する」を参照してください。

分岐/ラベル/マージ

- **ビルドにラベルでマークを付けて、後でそのビルドに戻れるようにする**
- **分岐を使用して対応リリースを分離する**
- **マージ パスに沿って分岐構造を計画する**
- **構成ファイルやソース ファイルが含まれる高いレベルで分岐する**
- **あまり深くまで分岐しない**
- **必要にならない限り分岐しない**
- **基点のないマージは、できる限り避ける**
- **“チェリーピック” マージよりフル マージを推奨**
- **頻繁にマージする**
- **メイン分岐としての機能が果たせるよう、新規のチーム プロジェクトの場合は必ず最上位フォルダを作成する**
- **マージの前に二重チェックができるよう、*candidate* スイッチまたは *preview* スイッチを使用してみる**
- **マージの処理に名前変更が含まれる場合は、ツールから推奨されるパスによく注意する**
- **マージの競合を解決する際には気を付ける**
- **マージの結果は一度に 1 つずつチェックインする**
- **マージからチェックインまでの間にビルドを行いテストを実行する**

ビルドにラベルでマークを付けて、後でそのビルドに戻れるようにする

ラベルでデイリー ビルドにマークを付けておくと、特定のビルドに使用される一連のファイルを見るのも取り出すのも簡単にできます。チーム ビルドでは、作成する各ビルドに関連のある一連のファイルに、自動的にラベルが割り当てられます。チーム ビルドのラベルのフォーマットは、たとえば “Releasex86_20070226.1” のように “リリース タイプ_ビルド番号” というフォーマットになりま

す。

ラベルはすべてのチーム ビルドに付きますが、独自のラベルを作成して、そのラベルを以下のようなビルドに付けておけば、後でそのラベルを目印にして簡単に目的のビルドに戻ることができます。

- 内部のマイルストーン (アルファ リリースなど)
- 分岐後または外部依存関係の統合後に作成したビルド

ラベルに名前を付けるときは、後でビルドの位置とその内容が簡単にわかるような名前にすることが望ましいです。内容がわかりやすいラベルを使用してください。たとえば、“AlphaBuild_20070112” のように “ラベル名_日付” というフォーマットを使用するなどしてください。

後でメンテナンス作業をしなければならない顧客にビルドをリリースするときは、ラベルではなく分岐を使用して、そのリリースに関係して将来発生するメンテナンス作業を区別して扱うようにしてください。その後で、その分岐と、その分岐に含まれている可能性のあるフィックスをメイン ソース ツリーにマージして戻すことができます。

既存のラベルを見つけるときは、[ファイル] メニューで [ソース管理]、[ラベル] をポイントし、[ラベルの検索] をクリックします。見つかったラベルは、変更することも、[ラベルの検索] ダイアログ ボックスから削除することもできます。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181439(VS.80).aspx の「ラベルの操作」を参照してください。
- ラベルの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181440(VS.80).aspx の「方法 : ラベルを適用する」を参照してください。

分岐を使用して対応リリースを分離する

対応リリース ビルドの生成に使用するソース コードを含む分岐を作成してください。この操作をする

と、メイン分岐に沿って継続されるソース コード ベースの継続開発に影響を与えることなく、ソフトウェアの対応リリース バージョンにフィックスと最新データを適用することができます。

アプリケーションの次期バージョンの開発は、対応リリース分岐と並行してメイン分岐に沿って継続されます。対応リリース分岐に対して行った安定化のための変更点は、製品の次期バージョンをリリースする前にメイン分岐にマージすることができます。

ソフトウェアのリリース バージョンのサポートに使用される、メンテナンス分岐の作成が済んだ後に分岐構造がどのような形になるか、その例を以下に示します。

- **Main** – 統合分岐
    - o **Source**
    - o **その他の資産フォルダ**
- **Releases** – メンテナンス分岐用のコンテナ
    - o **Release 1** – メンテナンス分岐
        - □**Source**

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

マージ パスに沿って分岐構造を計画する

ソース管理エクスプローラでは、既存の分岐パスに沿ってマージすることのみ可能です。コマンド ラインを使用すれば、基点のないマージを他のパスに沿って実行することも可能ですが、この種のマージはあまりよい方法ではなく、マージの競合が増えるばかりか、将来マージを行う際にも競合が発生してしまいます。

マージを実行する際には以下の点に注意してください。

- 親から子へ、あるいは子から親へという方向で階層に沿ってマージすれば、階層越しにマージした場合よりも競合が少なくなります。
- 分岐階層が分岐親および分岐子のベースとなります。この分岐階層というのは、ディスク上のソースコードの物理的な構造とは異なる場合があります。以下に例を示します。
- **物理構造 ：**
    - o **Development** – 開発分岐
    - o **Main** – 統合分岐
    - o **Releases** – リリース分岐用のコンテナ
        - □**Release 1** – リリース分岐
- **論理構造 ：**
    - o **Main**
        - □**Development**
        - □**Release 1**

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。

- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

構成ファイルやソース ファイルが含まれる高いレベルで分岐する

作成した分岐をコンパイルできるだけの、十分に高いレベルで分岐してください。たとえば、以下のソース ツリーの場合です：

- **Main：** 製品の出荷に必要なすべての資産用のコンテナ
- **Source：** ビルドに必要なすべてのもの用のコンテナ
    - **Code：** ソース コード用のコンテナ
    - **Shared Code：** 他のプロジェクトから共有されるソース コード用のコンテナ
    - **Unit Tests：** 単体テスト用のコンテナ
    - **Lib：** バイナリ依存関係用のコンテナ
- **Docs：** 製品と一緒に出荷する文書用のコンテナ
- **Installer：** インストーラのソース コードおよびバイナリ用のコンテナ
- **Tests：** テスト チームのテスト ケース用のコンテナ

新規のブランチの中にソース ファイルと構成ファイルがすべて含まれるよう、Source フォルダのレベルで分岐してください。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。

- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

あまり深くまで分岐しない

子分岐から最も上の親まで変更をマージするのに必要な時間が増えるため、あまり深くまで分岐しないでください。以下の分岐構造を例に説明します。

- **Development** – 開発分岐用のコンテナ
  - o **Development Branch**
    - □**Sub-Branch**
      - **Sub-Sub-Branch**
- **Main** – 統合分岐
  - o **Source**
  - o **その他の資産フォルダ**

分岐階層に沿ってマージを実行した方が競合は少なくなります。そのため、Main にマージする Sub-Sub-Branch に変更がある場合は、最初に Sub-Branch と Development Branch とにマージしてからでないと、Main へはマージできません。各マージの完了、競合の解決、ビルド、テストまでの所要時間は、分岐構造で作成した分岐のレベルとマージ時間とを掛け合わせた時間となります。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx

の「方法 ： ファイルおよびフォルダをマージする」を参照してください。

- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

必要にならない限り分岐しない

開発チームが同じファイル セットで同時に作業する必要がある場合を除き、分岐はしないでください。迷った場合は、ビルドにラベルを付けておいて、後でそのビルドから分岐を作成してもかまいません。分岐をマージするのは非常に手間がかかることがあります。分岐どうしの間の変更が大きい場合は特にそうです。

マージを行うためには、1 人以上の開発者がそのマージを実行して競合を解決する必要があります。マージの判断を誤ることによって、ビルドが不安定になることは珍しくないため、マージされたソースは徹底的にテストしなければなりません。

分岐階層越しにマージするのは特に難しく、普通なら自動的に処理される競合の多くを手作業で処理しなければなりません。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation

ソース管理の分岐およびマージ」を参照してください。

基点のないマージは、できる限り避ける

基点のないマージは、できる限り避けてください。基点のないマージを実行するときには、マージされる分岐に含まれているファイルとフォルダの関係を示す情報が TFS にまったく与えられません。そのため、マージの競合が増える結果になるのが一般的であり、将来マージを実行した場合さらに競合が増える事態につながるおそれがあります。

分岐ツリーの構造は、階層越しにではなく階層に沿って (つまり分岐ツリーの上下方向に沿って) だけマージすればよい構造にしてください。階層越しに分岐する場合は、基点のないマージを使用せざるを得ません。

マージを実行するときは以下の点に注意してください。

- 親から子へ、あるいは子から親へという方向で階層に沿ってマージすれば、階層越しにマージした場合よりも競合が少なくなります。
- 分岐階層が分岐親および分岐子のベースとなります。この分岐階層というのは、ディスク上のソースコードの物理的な構造とは異なる場合があります。以下に例を示します。

- 物理構造 :
    - **Development** – 開発分岐
    - **Main** – 統合分岐
    - **Releases** – リリース分岐用のコンテナ
        - **Release 1** – リリース分岐
- 論理構造 :
    - **Main**
        - **Development**
        - **Release 1**

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

“チェリーピック” マージよりフル マージを推奨

個々の変更を選び出して分岐間でマージ (“チェリーピック” マージ) するよりも、分岐全体を一度にマージするようにしてください。分岐に含まれている個々の変更を選び出して別の分岐とマージするのが便利なこともあります。しかし、チェリーピックされた変更セットが、将来のマージ範囲の中に収まってしまうと、その変更セットはマージし直されます。場合によっては、さらにマージの競合が発生しかねない結果となります。

コマンド ラインから実行する /discard マージの場合は特にその傾向が強いです。破棄された変更セットは、将来のマージ範囲の中に収まった場合、拾い出されます。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。

- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

頻繁にマージする

分岐間の待ち時間はなくしてください。特に、各チームがそれぞれに分離された分岐で作業しながら、同じリリースを対象に作業している場合は特にそうです。それによって、変更どうしの互換性が保証されます。

マージ スケジュールは、分岐構造の複雑さと、開発チームの個々のニーズによって左右されます。一例を次に示します。このような適度に複雑な分岐構造の場合、開発分岐はメイン分岐まで毎日マージすることができます。また 2 日に 1 回、統合分岐から変更を逆にマージすることもできます。

- **Development –** 分離された開発分岐用のフォルダ
  - o **External -** 外部依存関係分岐
  - o **Team 1 -** チーム分岐
  - o **Team 2 -** チーム分岐
    - ☐**Feature A –** 機能分岐
    - ☐**Feature B –** 機能分岐
    - ☐**Feature C –** 機能分岐
- **Main** – 統合分岐
- **Releases -** リリース分岐用のフォルダ
  - o **Release 2** – リリース分岐
- **Safe Keeping -** セーフキーピング分岐用のフォルダ
  - o **Release 1 -** セーフキーピング分岐

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「"Team Foundation ソース管理の分岐およびマージ」を参照してください。

メイン分岐としての機能が果たせるよう、新規のチーム プロジェクトの場合は必ず最上位フォルダを作成する

新規のチーム プロジェクトを開始するときは、チーム プロジェクトの下にフォルダ (フォルダ名は Main とすることが多い) を作成することから始めてください。Main 分岐のソースはすべてこのフォルダに入れてください。新しい分岐を作成する必要があるときは、Main フォルダから直接分岐してください。

よくある分岐構造の例を下に示します。

- **Development** – Main から分岐された分離済み開発ブランチ用のフォルダ
  - o **Feature A**
    - □**Source**
  - o **Feature B**
    - □**Source**
- **Main** – 統合分岐
  - o **Source**
  - o **その他の資産フォルダ**
- **Releases** – Main から分岐されたリリース分岐用のフォルダ

o **Release 1** – メンテナンス分岐

□**Source**

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「"Team Foundation ソース管理の分岐およびマージ」を参照してください。

マージの前に二重チェックができるよう、candidate スイッチまたは preview スイッチを使用してみる

マージを実行する前に、candidate スイッチまたは preview スイッチを使用してマージ結果を二重チェックしてください。このオプションはコマンド ラインからしか実行できませんが、merge コマンドを実行するときに、どのファイル、どのバージョンがマージされるかがわかるのは非常に有利です。たとえば、このオプションを使用すれば、マージの範囲が予想を大きく超えることはなくなり、merge コマンドが及ぼす影響も確実に理解できるようになります。大規模なマージの場合、作業を各個人あるいは各チームに振り分けるのにはマージ レポートが役立ちます。

マージの結果をプレビューするときは、Tf.exe コマンド ライン ツールを使用して、preview か candidate のいずれかのスイッチを指定した merge コマンドを実行してください。以下に例を示します。

Tf merge main/source development/feature/source /preview

preview スイッチを指定した場合は、マージのプレビューが表示されます。candidate スイッチを指定した場合は、ソースに含まれている変更セットのうち、まだ移動先にマージされていない変更セットすべてを列挙したリストが表示されます。このリストには、まだマージされていない変更セット ID と、その変更セットに関する基本情報が含まれています。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

マージの処理に名前変更も含まれる場合は、ツールから推奨されるパスによく注意する

マージの処理に名前変更も含まれる場合は、ツールから推奨されるパスによく注意し、適宜変更してください。名前変更にはすべて、競合というマークが付きます。TFS で使用されているマージ アルゴリズムは、名前変更がマージされている最中に、最適なターゲット パスを計算しようとします。場合によっては、デフォルトのターゲット パスが、希望するパスにならないこともありますので、マージされたファイルをコミットする前に必ず二重チェックをかけるようにしてください。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

マージの競合を解決する際には気を付ける

マージを実行するときには気を付けてください。ビルドを不安定にしかねない間違いを犯しやすいからです。ファイルをマージするときは次のようにしてください。

- マージしようとしているコードを二重チェックしてから、そのマージをコミットする。
- 生成されたマージ済みファイルをソース管理にチェックインして戻す前に、そのマージ済みファイルをコンパイルできるかどうかテストする。
- マージ済みファイルをソース管理にチェックインして戻す前に、関連のある単体テストを実行して、そのテストに合格することを確認する。
- 別の開発者が行った変更がどういう性質の変更であるのかを確実に理解する。一部の行について、それをマージすることに迷いがある場合は、変更を行った開発者に話を聞いてみてください。そうすれば、その開発者の目的がよくわかり、マージの影響に関して別の意見も得られます。

マージが終了した後は、生成されたソースをコンパイルし、単体テストを実行して、大きな不具合がないかどうかを調べてください。

参考資料

- マージの競合を解決する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181432(VS.80).aspx の「競合の解決 (Team Foundation ソース管理)」を参照してください。

マージの結果は一度に 1 つずつチェックインする

同じファイルを要するマージを実行する場合は、まず 1 つ目のマージの結果をチェックインしてから、次のマージを実行してください。最初のマージの実行が済んだ後、コードをコンパイルし、単体テストに合格したことを確かめ、それから保留中の変更をチェックインします。その後で次のマージを開始し、同じ処理を繰り返します。

こうすればあまり複雑にならず、各マージを分離しておくことにより、必要に応じて変更を元に戻すこともできます。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

マージからチェックインまでの間にビルドを行いテストを実行する

マージの実行が済んだら、マージ済みファイルをチェックインする前に、コードをコンパイルし、関係

のあるテストを実行してください。これは、マージが原因でビルドが不安定になるのを避けるためです。

マージの結果は、最初はワークスペースの内部で分離され、保留中の変更をチェックインするまでサーバーにはアップロードされません。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「"Team Foundation ソース管理の分岐およびマージ」を参照してください。

チェックインとチェックイン ポリシー

- **コードを共有する準備の整ったときのみコードをチェックインする**
- **シェルブセットを使用して、保留中の変更をバックアップまたは共有する**
- **チェックインのたびに 1 つずつ作業項目を解決する**
- **チェックイン ポリシーを使用してコーディング基準を守らせる**
- **チェックイン ポリシーを使用してコード品質ゲートを守らせる**
- **いつポリシーが無視されたかを検出する**
- **競合を避けるよう計画する**

コードを共有する準備の整ったときのみコードをチェックインする

更新済みコードをソース管理にチェックインするのは、単体テストが完全に済んで、チームの残りのメンバと共有する準備が整ったときのみにしてください。まだ完了していない中間作業にはシェルブセットを使用してください。

コードをチェックインすると、そのコードが次のスケジュールされたビルドの一部として使用されます。コードが不完全だと、それが原因でビルドが不安定になることがあります。また、コードをチェックインすると、get latest コマンドを実行する他の開発者が変更を取り込みます。その変更が不完全だったり、テストが不十分だったりした場合は、それが原因で同僚に問題を引き起こします。

参考資料

- チェックインの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181411(VS.80).aspx の「方法 : 保留中の変更をチェックインする」を参照してください。

シェルブセットを使用して、保留中の変更をバックアップまたは共有する

まだチェックインする準備の整っていない保留中の変更を含むファイルをバックアップするときは、シェルブセットを使用してください。シェルブセットは、コードを共有してコード レビューを行うのにも使用でき、開発タスクを別の開発者に渡す場合にもシェルブセットが使用できます。シェルブセットを使用すれば、ビルドを不安定にしかねない不完全な作業をチェックインしなくても、保留中の変更をサーバーにアップロードすることができます。

保留中の一連の変更をシェルブするときは、まず [ソリューション エクスプローラ] で自身のソリューションを右クリックして変更を表示し、その後 [保留中の変更を表示] をクリックします。シェルブするファイルを選択し、[シェルブ] をクリックします。シェルブセット名を入力し、そのシェルブセットの目的を示したコメントを入力し、その後、[シェルブ] をクリックします。

シェルブセットを取り出すには、次の操作を行います。

1. [ファイル] メニューで [ソース管理] をポイントし、[保留中の変更をアンシェルブ] をクリックします。
2. [所有者の名前] テキスト ボックスに、シェルブセットの作成者名を「Contoso¥JimB」または単に「JimB」などのように入力し、[検索] をクリックします。
3. 結果ペインで、ワークスペースにアンシェルブするシェルブセットを選択し、[詳細] をクリックします。
4. TFS ソース管理サーバーから取り出したシェルブセットを削除する場合は、[シェルブセットをサーバーに保持する] オプションをオフにします。Team Foundation Server によって、シェルブされた各バージョンが保留中の変更として目的のワークスペースに復元されますが、これは、その変更が既にワークスペース内で保留になっている変更と競合しない場合に限ります。
5. 作業項目とチェックイン メモを、復元されたシェルブセットに関連付けたくない場合は、[作業項目およびチェックイン メモの復元] オプションをオフにすることもできます。[詳細] ダイアログボックスが表示されたら、ワークスペースにアンシェルブするシェルブセットまたはシェルブセット項目を選択し、[アンシェルブ] をクリックします。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

チェックインのたびに 1 つずつ作業項目を解決する

作業項目が解決された後は、保留中の変更をチェックインし、その作業項目の状況を更新してください。この手順には以下の目的があります。

- この手順を実行することにより、ソース管理データベース内の記録に矛盾がなくなります。このデータベースは、作業項目別の変更をレビューするときにも使用でき、将来必要になった場合にバックアップから作業項目を取り出すためにも使用できます。
- チェックインから次のチェックインまでそれほど待機しなくても済むようになります。チェックインから次のチェックインまで待機する時間が長くなるほど、手動マージや追加テストを必要とする競合が発生する確率が増えます。

参考資料

- チェックインの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181411(VS.80).aspx の「方法 ： 保留中の変更をチェックインする」を参照してください。
- 作業項目と変更セットの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181410(VS.80).aspx の「方法 ： 作業項目を変更セットに関連付けるを参照してください。
- 作業項目詳細をレビューする詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181401(VS.80).aspx の「方法 ：[保留中の変更] ウィンドウから作業項目の詳細を表示する」を参照してください。

チェックイン ポリシーを作成してコーディング基準を守らせる

開発チーム全体にコーディング基準を守らせたいときは、チェックイン ポリシーを使用してください。TFS ソース管理にチェックインしたすべてのソース コードが確実にコーディング基準を満たすようにしたい場合は、チェックイン ポリシーが便利です。

TFS に同梱されているコード分析チェックイン ポリシーを利用すると、チェックインの前に必ずコード分析の実行を済ませておくことができます。コード分析ポリシーは、さまざまなルールをチェックできるよう微調整が可能です。たとえば、設計、相互運用性、メンテナンス性、移植性、命名規則、信頼性などに適用されるルールをチェックすることができます。

チーム プロジェクトでコード分析チェックイン ポリシーを適用するには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトを右クリックし、[チーム プロジェクトの設定] を指して、[ソース管理] をクリックします。
2. [チェックイン ポリシー] タブをクリックしてから、[追加] をクリックします。
3. [チェックイン ポリシーの追加] ダイアログ ボックスで、[コード分析] をクリックし、[OK] を

クリックします。

4. [コード分析ポリシー エディタ] で、[C/C++ コード分析 (/analyze) を強制] または [マネージ コードのコード分析を強制] をクリックします。マネージ コードとアンマネージ コードの両方がプロジェクトに含まれている場合は、両方とも選択してください。
5. [マネージ コードのコード分析を強制] を選択した場合は、必要なコーディング基準に従って、マネージ コード分析に必要なルール設定を設定します。これにより、どのルールが適用されるかが正確に決まります。

既定では利用できないチェックを実行するため、カスタム チェックイン ポリシーを作成することもできます。たとえば、禁止されているアプリケーション プログラミング インターフェイス (API) 呼び出しなどのコード パターンを禁止することもできます。また、ソース コード内のどこに中かっこを配置すべきかなど、チームごとのコーディング スタイルに関するガイドラインを適用するポリシーを記述することもできます。

参考資料

- カスタム チェックイン ポリシーの作成および使用の詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してください。
- チェックイン ポリシーのカスタマイズ方法を知りたいときは、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインの際に特定のパターンだけを禁止するサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインの際にコメントを義務付けるサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーの登録方法を知りたいときは、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New

Check-In Policy! How Do I Add It?」を参照してください。

チェックイン ポリシーを使用してコード品質ゲートを守らせる

コード品質ゲートを守らせたい場合は、コード分析とテスト ポリシーを組み合わせて使用してください。たとえば、特定のテストを実行し、そのテストに合格してからでないとソースが TFS ソース管理にチェックインされないようにするときは、付属のテスト ポリシーを使用します。また、セキュリティ、パフォーマンス、移植性、メンテナンス性、信頼性の各ルールを確実にすることにより、コードが一定の品質基準を満たせるようなコード分析ポリシーを設定することも可能です。

この種のチェックイン ポリシーを適用したうえで、コーディングの基準とガイドラインを守らせるポリシーも適用することにより、コードが特定のコード品質ゲートを満足するようになります。

チーム プロジェクトでコード分析チェックイン ポリシーを適用するには、次の操作を行います。

1. [チーム エクスプローラ] で、チーム プロジェクトを右クリックし、[チーム プロジェクトの設定] を指して、[ソース管理] をクリックします。
2. [チェックイン ポリシー] タブをクリックし、[追加] をクリックして、適切なポリシーを選択して設定します。

参考資料

- カスタム チェックイン ポリシーの作成および使用の詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム チェックインを作成する方法」を参照してください。
- チェックイン ポリシーのカスタマイズ方法を知りたいときは、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインの際に特定のパターンだけを禁止するサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to

Disallow Certain Patterns」を参照してください。

- チェックインの際にコメントを義務付けるサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーの登録方法を知りたいときは、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。

いつポリシーが無視されたかを検出する

Team Foundation Version Control には、ポリシーの無視を防ぐ手段がありません。ただし、以下の手順を実行すれば、ポリシーが無視されたかどうかを検出することができます。

- チェックイン イベントを捕捉するため、Team Foundation Core Services API から Team Foundation Eventing Service を使用します。
- 変更セットの詳細を解析して、チェックイン ポリシーが無視された場合にその対応を行う Notify メソッドを記述します。

また、手作業で変更セットの履歴を細かく調べて、ポリシーが無視されたかどうかを見つけることもできます。

参考資料

- チェックイン ポリシーの無視については、http://msdn2.microsoft.com/ja-jp/library/ms245460(VS.80).aspx の「方法 : チェックイン ポリシーをオーバーライドする」を参照してください。
- Team Foundation Eventing Service の詳細については、http://msdn2.microsoft.com/en-us/library/bb130154(vs.80).aspx の「Eventing Service」を参照してください。
- ポリシー違反があったときに電子メール メッセージを受信する方法については、

http://blogs.infosupport.com/marcelv/archive/2005/10/18/1635.aspx を参照してください。

競合を避けるよう計画する

複数の開発者が同時に同じソース ファイルの同じ領域で作業しないように計画を立ててください。そうすれば、解決しにくい競合の発生を避けることができます。多くの競合は自動競合解決で解決できますが、それでも、2 人以上の開発者が同じメソッドで作業すること、あるいは同じコード行で作業することは避けるべきです。同じコード行に発生した競合は、手作業で解決しなければならず、そのためマージの処理が複雑になります。チーム内で意思の疎通を図ることが重要です。

ファイルの処理を開始する前に、ソース管理から最新バージョンが得られていることを確認し、作業を開始する前に他の人がそのファイルをチェックアウトしたかどうかを確認してください。既に同僚がそのファイルをチェックアウトしていた場合は、どのような作業をしているのかを尋ねてから、その同僚による変更が完了するのを待機したほうがよいのか、それとも同じファイルを対象に並行して作業を続けても安全かどうかを判断してください。同じファイルの別のソース コード領域で別の機能の処理をしているのであれば、安全です。

だれか他の人によってファイルがチェックアウトされたかどうかを確認するには、次の操作を行います。

1. Visual Studio の [チーム エクスプローラ] ウィンドウで [ソース管理] をダブルクリックします。
2. ソース管理フォルダ階層の中で、確認するファイルを含んでいるフォルダに移動します。
   保留中の変更が、その変更を所有しているユーザーのユーザー名と一緒に表示されます。

現在どのファイルの変更が保留になっているのかを確認するため、Visual Studio 2005 のコマンド プロンプト ウィンドウから以下のコマンドを実行します。

Tf status /format:detailed /user:*

他の人も並行して作業することがわかっているソース ファイルの処理を開始するときは、他のチームメンバに対して、自分も同じファイルの処理をしようとしていることを伝え、さらにどのような点を更新しようとしているのかも伝えてください。

参考資料

- 自身のワークスペースで保留中の変更を表示する詳細については、http://msdn2.microsoft.com/en-us/library/ms181400(VS.80).aspx の「方法 : ワークスペース内のすべての保留中の変更を表示および管理する」を参照してください。

- 他のワークスペースで保留中の変更を表示する詳細については、http://msdn2.microsoft.com/en-us/library/ms181401(VS.80).aspx の「方法 : 他のワークスペース内の保留中の変更を表示する」を参照してください。

チェックアウト、取得、およびロック

- **変更する前に最新のソースを取得する**
- **lock コマンドは慎重に使用する**
- **いつファイルをロックするかをチームメイトに伝える**

変更する前に最新のソースを取得する

作業しているプロジェクトのどのソース ファイルについても、すべて最新バージョンであることを確認するため、ファイルをチェックアウトして編集する前に get latest コマンドを実行してください。このコマンドを実行しないまま、最新ではないソース ファイルを基準にして自身のコードをローカルにビルドしてしまうと、後でサーバーにチェックインしたときに、そのコードが原因でビルドに問題が生じる確率が増えます。

特定のチーム プロジェクトに関連のあるファイル セットの最新のコピーを取得するため、[ソース管理エクスプローラ] で目的のチーム プロジェクトを右クリックし、[最新バージョンの取得] をクリックしてください。現在自身のワークスペース内に、編集が保留になっている書き込み可能ファイルがあっても、このコマンドを実行したことでそのファイルが上書きされることはありません。現在のワークスペースにマップされているディレクトリから Tf get /all コマンドを実行すれば、コマンド ラインか

らも同じコマンドが実行できます。

参考資料

• Get コマンドの詳細については、
http://msdn2.microsoft.com/ja-jp/library/ms181420(VS.80).aspx にある MSDN Web サイトの「Get コマンド」を参照してください。

lock コマンドは慎重に使用する

lock コマンドは慎重に使用してください。作業中のファイルにロックをかけると、他の人がそのファイルを同時に処理できなくなり、開発プロセスが滞るおそれがあるからです。手作業による複雑なマージを行わなければならないような競合が生じるのが心配であれば、ファイルをロックするのは、そのファイルを編集している最中だけに限ることです。可能な場合は、ファイルをチェックアウトするときに [なし] というロック タイプを選択して共有チェックアウトを実行することを推奨します。

競合を避けるためにロックを使用した方が適切な例を以下に示します。

- 画像ファイルのようなバイナリ ファイル フォーマットや、簡単にマージできないドキュメントを修正している場合
- ファイルに加える変更の範囲が広いために、同じファイルを処理している他の開発者も同じ行を変更してしまう確率が高い場合

競合の可能性を一切避けたい場合は、チェックアウト ロックを使用してください。チェックアウト ロックを使用すると、他の人は同じファイルをチェックアウトすることもチェックインすることもできなくなります。ファイルをロックする場合は、いつファイルをロックするのかをチームメイトに知らせて、ロックする理由と、そのファイルにかかる作業のだいたいの所要時間も伝えてください。さらに、いつロックを解除して変更をチェックインし直すのかについても知らせてください。
ファイルをチェックアウトして編集するときはロック タイプを指定しますが、明示的にファイルをロックすることもできます。

チェックアウトの最中にファイルをロックするには、以下の操作を行います。

1. [ソース管理エクスプローラ] で、目的のファイルを右クリックし、[編集用にチェックアウト] をクリックします。
2. ロック タイプを [なし]、[チェックアウト]、[チェックイン] の中から指定します。
3. ファイルを明示的にロックするときは、目的のファイルを右クリックし、[ロック] をクリックし、[チェックアウト] か [チェックイン] かいずれかのロック タイプを選択します。

Microsoft Visual Source Safe® (VSS) の動作とは違って、ファイルをチェックアウトしても、最新バージョンは暗黙的には取得されません。ファイルをロックする前に、[最新バージョンを取得] をクリックして、ワークスペース内に最新バージョンがあることを確認してください。

参考資料

- ロックの詳細については、http://msdn2.microsoft.com/en-us/library/ms181420(VS.80).aspx の「方法 : フォルダまたはファイルをロックおよびアンロックする」を参照してください。
- ロックの種類の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181419(VS.80).aspx を参照してください。

いつファイルをロックするかをチームメイトに伝える

ソース ファイルをロックする場合は、その旨をチームメイトに伝え、そのファイルが利用できない間に各チームメイトがそれぞれの作業計画を立てられるようにしてください。そのファイルの排他的ロックが必要な理由と、いつまでロックしておくことになりそうかを説明してください。ファイルをロックすると、他の開発者が同じソース ファイルを処理したくてもできなくなり、開発サイクルに支障が生じるおそれがあります。

手作業による複雑なマージを行わなければならないような競合が生じることが心配であれば、ファイルをロックするのは、そのファイルを編集している最中だけにしてください。可能な場合は、ファイルをチェックアウトするときに [なし] というロック タイプを選択して、共有チェックアウトを実行することを推奨します。

[ソース管理エクスプローラ] からファイルをロックするときは、目的のファイルを右クリックし、[ロック] をクリックし、[チェックアウト] あるいは [チェックイン] いずれかのロック タイプを選択します。

参考資料

- ロックの種類の詳細については http://msdn2.microsoft.com/ja-jp/library/ms181419(VS.80).aspx を参照してください。

依存関係

- **できる限りプロジェクト参照を使用する**
- **ファイル参照は、どうしても必要な場合のみ使用する**
- **プロジェクト参照およびファイル参照に copy local = true を使用する**
- **Web サービスを参照する際、動的 URL を使用する**

できる限りプロジェクト参照を使用する

以下のような利点があるため、できる限りプロジェクト参照を使用するようにしてください。

- プロジェクト参照は、ソリューションとプロジェクト セットがロードされるどの開発ワークステーションでも機能します。それは、プロジェクトの GUID (Globally Unique Identifier) がプロジェクト ファイル内に配置されているためです。この GUID があるために、現在のソリューションの中で紛れることなく参照先プロジェクトが識別できます。
- プロジェクト参照を使用すれば、Visual Studio のビルド システムにプロジェクトの依存関係を追跡させて、プロジェクトの正しいビルド順序を決定することができます。
- また、参照先アセンブリが特定のコンピュータ上にない状態を避けることができます。
- プロジェクト参照は自動的にプロジェクト構成の変化を追跡します。たとえば、デバッグ構成を使用してビルドを行うときには、どのプロジェクト参照も、参照先プロジェクトによって生成されるデバッグ アセンブリを参照します。同様にリリース構成ではリリース アセンブリを参照します。

これはつまり、参照をリセットしなくても、プロジェクト全体でデバッグからリリース ビルドへと自動的にスイッチできるということです。

- プロジェクト参照を使用すれば、循環依存関係の検出と循環依存関係の防止を Visual Studio に実行させることができます。

アセンブリがソリューションのプロジェクト セット内に存在していれば、プロジェクト参照が使用できます。アセンブリがソリューションのプロジェクト セットの外部にあり、それでもプロジェクト参照を使用したい場合は、おおもとのプロジェクトから自身のプロジェクトへ依存関係を分岐することができます。依存関係の新バージョンを取得するときは、そのおおもとのプロジェクトから自身の分岐へマージを実行してください。

参考資料

- プロジェクト参照およびファイル参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。
- 参照を追加する詳細については、http://msdn2.microsoft.com/ja-jp/library/wkze6zky(VS.80).aspx の「方法 : Visual Studio のリファレンスを追加または削除する」を参照してください。

ファイル参照は、どうしても必要な場合のみ使用する

現在のソリューションのプロジェクト セットの外部にあるアセンブリを参照する必要があるために、プロジェクト参照が使用できない場合や、おおもとのプロジェクトから自身のプロジェクトへ分岐を作成したくない場合は、ファイル参照を設定しなければなりません。

参考資料

- プロジェクト参照およびファイル参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」

を参照してください。

プロジェクト参照およびファイル参照に copy local = true を使用する

どの参照にも copy local 属性が関連付けられています。参照が最初に追加されるときに、Visual Studio がこの属性の初期設定を true または false に決定します。参照先アセンブリがグローバル アセンブリ キャッシュ (GAC) に入っていることが判明した場合は false に設定され、それ以外の場合は true に設定されます。
このデフォルト設定はなるべく変更しないでください。

copy local が true に設定されていると、Visual Studio のビルド システムによって、どの参照先アセンブリも (依存関係の下流にあるアセンブリも含めて)、クライアント プロジェクトの出力フォルダにコピーされます。たとえば、クライアント プロジェクトが Lib1 という名前のアセンブリを参照し、Lib1 が Lib2 と Lib3 とに依存している場合は、ビルド時に Visual Studio によって Lib1、Lib2、Lib3 がプロジェクトのローカル出力フォルダに自動的にコピーされます。

参考資料

- プロジェクト参照およびファイル参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。

Web サービスを参照する際、動的 URL を使用する

Web サービスを呼び出す場合は、最初に Web 参照をプロジェクトに追加しなければなりません。すると、呼び出す Web サービスと対話するときの仲介となるプロキシ クラスが生成されます。初めプロキシ コードには、Web サービスの静的 URL (例 : http://localhost or http://SomeWebServer) が含まれています。

重要事項 : 現在のソリューションに含まれている Web サービスのうち、自身のコンピュータ上で実

行する Web サービスに対しては、参照がどのコンピュータ上でも有効になるように、http://MyComputerName ではなく必ず http://localhost を使用してください。

プロキシに埋め込まれている静的 URL は通常、実稼働環境にもテスト環境にも不要な URL です。一般的に、アプリケーションが開発からテスト、そして実稼働へと移行していくにつれて、必要な URL も変化します。この問題を解消するオプションが 3 つあります。

- プロキシ クラスのインスタンスを作成するときに、Web サービスの URL をプログラムで設定することができます。
- プロキシ内でハードコーディングされた URL を避けるさらに柔軟な方法は、Web サービス参照の [URL の動作] プロパティを [ダイナミック] に設定する方法です。この方法を推奨します。このプロパティを [ダイナミック] に設定すると、コードがプロキシ クラスに追加され、Web サービス URL がアプリケーション構成ファイルのカスタム構成セクションから取得されます (Web アプリケーションの場合は Web.config、Windows アプリケーションの場合は SomeApp.exe.config)。
- WSDL.exe コマンド ライン ツールを使用して /urlkey スイッチを指定しても、プロキシを生成することができます。この場合も、コードがプロキシに追加されて Web サービス URL が取得されるという点では、[URL の動作] プロパティを設定するのと同様に機能しますが、URL はアプリケーション構成ファイルの <applicationSettings> セクションに格納されます。

動的 URL 手法の場合も、ユーザー構成ファイルを提供することができ、そのユーザー構成ファイルの方をメインのアプリケーション構成ファイルよりも優先させることができます。この方法により、個々の開発者 (およびテスト チームのメンバ) が一時的に Web サービス参照を代替ロケーションにリダイレクトすることができます。

参考資料

- 詳細については、このガイドの「第 6 章 Managing Source Control Dependencies in Visual Studio Team System」を参照してください。

分散開発/遠隔地開発

- **プロキシには、適切なサイズのディスク ドライブを利用する**
- **定期的に最新のファイルが取得できるようスケジュール タスクを作成する**
- **プロキシ パフォーマンス カウンタとイベント ログとを定期的に監視する**
- **ファイル サイズと帯域幅に従って *executionTimeout* を設定する**
- **プロキシが長期間機能停止する場合は、そのプロキシを使用不可にする**
- **不要なファイル転送を減らすためワークスペース クローキングを利用してみる**

プロキシには、適切なサイズのディスク ドライブを利用する

ハードウェアに関しては、TFS のインストール ガイドに示した推奨値に従ってください。特に、十分な容量を持ったディスク ドライブを利用するようにしてください。ストレージ容量には、プロキシがファイルをキャッシュに入れるときに利用できる最大値が設定してあります。この最大値に達すると、キャッシュ内の古いファイルが削除されて、ストレージ容量が少し空きます。空いたスペースは、新たに要求されたファイルをキャッシュするために利用されます。クリーンアップをかけると、前回いつアクセスされたかに応じて各ファイルが削除されます。最も長い期間利用されていないファイルから順に削除されます。

参考資料

• Team Foundation Server プロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx にある MSDN Web サイトの「"Team Foundation Server のプロキシとソース管理」を参照してください。

定期的に最新のファイルが取得できるようスケジュール タスクを作成する

定期的に最新ファイルをプロキシ サーバーに取得したいときは、スケジュール タスクを実行してください。こうすれば、必ずプロキシ キャッシュに最新バージョンのファイルが入っていることになり、

そうしたファイルをクライアントが要求した場合に、必ずキャッシュ内に目的のファイルが見つかることになります。

参考資料

• Team Foundation Server プロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx にある MSDN Web サイトの「"Team Foundation Server のプロキシとソース管理」を参照してください。

プロキシ パフォーマンス カウンタとイベント ログを定期的に監視する

定期的にプロキシ パフォーマンス カウンタと Windows イベント ログを監視して、エラーや警告が出ていないかどうかを調べておくと、プロキシの動作状態を正確に把握することができます。プロキシキャッシュの機能を有効にして、キャッシュのパフォーマンスを監視してください。

以下のパフォーマンス カウンタに注意することが望ましいです。

- 現在のキャッシュ サイズ
- キャッシュ ヒットの総数 : 回数と比率
- ダウンロード要求の総数
- キャッシュ内のファイルの総数
- キャッシュ ミスの総数 : 回数と比率

こうしたパフォーマンス カウンタは、プロキシをインストールするときに登録されます。プロキシのパフォーマンス カウンタはマルチインスタンス化されます。マルチインスタンス化というのは、アプリケーション層ごとのカウンタが Proxy.config ファイルに設定されているということです。このデータを収集することにより、プロキシ サーバーが稼働しているときのパフォーマンスが理解できます。

TFS プロキシによって、キャッシュ パフォーマンス統計情報が ProxyStatistics.xml という名前の XML ファイルに保存されますが、保存するときの時間間隔は変更できます。ProxyStatistics.xml ファイルは、プロキシをインストールしたディレクトリ内の App_Data フォルダにあります。

参考資料

• Team Foundation Server プロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx にある MSDN Web サイトの「”Team Foundation Server のプロキシとソース管理」を参照してください。

ファイル サイズと帯域幅に従って executionTimeout を設定する

帯域幅の狭いネットワーク (3 Mbps 未満) 経由で大きなファイルをダウンロードすることになることがわかっている場合は、Web.config の executionTimeout を適切な値に設定してください。そうすると、タイムアウトする確率が減ります。以下のようにデフォルト値は 1 時間です。

```
<httpRuntime executionTimeout="3600"/>
```

参考資料

• Team Foundation Server プロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx にある MSDN Web サイトの「”Team Foundation Server のプロキシとソース管理」を参照してください。

プロキシが長期間機能停止する場合は、そのプロキシを使用不可にする

プロキシが長期間機能停止する場合は、クライアント上でプロキシを無効にしてください。そうすれば、無駄な再接続が防げます。特に指定しない限りは、5 分ごとに再接続が試行されます。

参考資料

• Team Foundation Server プロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx にある MSDN Web サイトの「”Team Foundation Server のプロキシとソース管理」を参照してください。

不要なファイル転送を減らすためワークスペース クローキングを利用してみる

ワークスペースの一部を見えないようにしたい場合、あるいは不要なファイル転送を防止したい場合は、ワークスペース クローキングの使用を検討してください。クローキングを使用して、現在必要でないフォルダおよびファイルがローカル ワークスペースにコピーされないようにすれば、指定したワークスペース フォルダが見えないようになるだけでなく、パフォーマンス帯域幅が増え、ローカル ディスク スペースが節約されます。ワークスペース内の既存のフォルダ マッピングをクロークすることもできますが、もっといい方法は、クローク専用の新しいフォルダ マッピングを作成することです。

ワークスペース内のフォルダをクロークするには、以下の操作を行います。

1. Visual Studio で、[ファイル] メニューから、[ソース管理] を指して、[ワークスペース] をクリックします。
2. [ワークスペースの管理] ダイアログ ボックスで、クローキングを適用するワークスペースを選択し、[編集] をクリックします。
3. [ワークスペースの編集] ダイアログ ボックスの [作業フォルダ] リストで、クロークする [ソース管理フォルダ] および [ローカル フォルダ] の下にあるフォルダ マッピングを強調表示するか、新規に 1 つ作成するか、いずれかの操作をします。
4. [状態] 列をクリックし、設定を [アクティブ] から [クローク] に変更します。
   アクティブにマップされた TFS ソース管理フォルダでしか、フォルダはクロークできないことに注意してください。
5. [OK] をクリックして [ワークスペースの編集] を閉じ、[閉じる] をクリックして [ワークスペースの管理] を閉じます。

ワークスペース全体にさらに get コマンドを実行するまで、ファイルは非表示にはならないことに注意してください。

参考資料

• Team Foundation Server プロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx にある MSDN Web サイトの「”Team Foundation Server のプロキシとソース管理」を参照してください。

移行

- **VSS コンバータを使用して Team Foundation Server ソース管理へ移行する**
- **他のソース管理システムから Team Foundation Server ソース管理へ移行する**

VSS コンバータを使用して Team Foundation Server ソース管理へ移行する

Team Foundation Server には、ファイル、フォルダ、バージョン履歴、ラベル、ユーザー情報を Visual SourceSafe データベースから Team Foundation Server Version Control へ移行できる VSS コンバータ ツールが同梱されています。

このコンバータには、理解しておくべき重要な制約があります。たとえば以下です。

- ファイル共有に関する履歴情報は保存されません。Team Foundation Server は共有には対応していません。共有ファイルを移行すると、そのファイルは移動先フォルダにコピーされます。
- 分岐に関する履歴情報は保存されません。
- Team Foundation Server はピン設定には対応していません。2 つのラベルを作成することにより、ピン設定されたファイルが移行されます。
- 操作の時間を示すタイムスタンプは移行の際には保存されません。

参考資料

• ファイルを移行する詳細については、このガイドの「Visual Source Safe から Visual Studio Team Foundation Server へソース コードを移行する方法」を参照してください。

他のソース管理システムから Team Foundation Server ソース管理へ移行する

ファイルは、移行の際の起点となるバージョン管理システムから Team Foundation Server Version Control へと手動でエクスポートすることができます。移行の際の起点となるバージョン管理システムから、ファイル履歴またはその他属性を保存する必要がある場合は、http://www.codeplex.com/MigrationSyncToolkit にある TFS Migration および Synchronization Toolkit を利用して独自の移行ツールを作成することができます。

Microsoft は現在 ClearCase コンバータに取り組んでいる最中です。このコンバータについては、準備が整いしだい、http://blogs.msdn.com/tfs_migration にある TFS Migration blog で発表します。

Component Software が、GNU RCS、CS-RCS、GNU CVS、Subversion (SVN)、Visual SourceSafe (VSS) と互換性のあるコンバータを作成しました。

参考資料

- TFS による移行の詳細については、http://blogs.msdn.com/tfs_migration にある MSDN Web サイトの「TFS Migration blog」を参照してください。
- CodePlex から TFS Migration および Synchronization Toolkit をダウンロードするときは、http://www.codeplex.com/MigrationSyncToolkit にアクセスしてください。
- Component Software コンバータの詳細については、http://www.componentsoftware.com/Products/Converter/ を参照してください。

プロジェクト/ワークスペースの管理

- **分岐ではなくワークスペースを使用して 1 人の開発者を分離する**
- **Windows エクスプローラではなくソース管理を使用してファイルの削除と名前変更を行う**
- **ソリューションが開いた状態でのみ削除と名前変更を行う**
- **アプリケーションのバージョンから別のバージョンへ資産を移動する場合に、アプリケーションごとに 1 つずつチーム プロジェクトを作成する**
- **アプリケーションのバージョンごとに 1 からやり直す場合に、バージョンごとに 1 つずつチーム プロジェクトを作成する**

- **統合テストを必要とするコードとバイナリを共有するために分岐を使用する**
- **プロジェクトどうしの依存関係に対応できるよう、ワークスペース マッピングは避ける**
- **チーム プロジェクトのルート レベルでワークスペース マッピングを作成する**
- **共有コンピュータ上で、他と重複しないローカル フォルダ パスを使用する**
- **ソース ツリーの一部だけをマッピングしてみる**
- **分岐に対応できる構造をしたソース ツリーを作成する**

分岐ではなくワークスペースを使用して 1 人の開発者を分離する

自身の作業をチームの他のメンバから分離するときは、別のワークスペースを使用し、新しい分岐は作成しないでください。チームの他のメンバによって処理されるファイルおよびフォルダへの参照を含んでいる (つまり共有ソースを含んでいる) 1 次ワークスペースを使用し、分離するファイルおよびフォルダを管理するための 2 次ワークスペースを作成してください。こうしたファイルを分離しておいて、どこか別の場所で行われている処理と並行して特定のファイルを変更した方がいい場合もあります。たとえば、危険のある保留中の変更を処理する場合や、実験的な変更を処理する場合に、この方法が使用できます。2 次ワークスペースを使用することにより、分岐やマージが複雑にならずに済みます。

2 次ワークスペースを作成するには、次の操作を行います。

1. [ソース管理エクスプローラ] で [ワークスペース] ドロップダウン リストをクリックし、[ワークスペース] をクリックします。
2. [ワークスペースの管理] ダイアログ ボックスで、[追加] をクリックします。
3. [ワークスペースの追加] ダイアログ ボックスに、「MyIsolatedWork」のような新しいワークスペースの名前を入力し、そのワークスペースの目的がよくわかるようなコメントを入力します。
4. [作業フォルダ] リストで、ワークスペースの状況を [アクティブ] に設定し、ワークスペースに含まれることになるソース管理フォルダ (チーム プロジェクトのルート フォルダでもサブフォルダでも可) を確認し、ワークスペースからのファイルを含む自身のコンピュータ上でのローカル フォルダ パスを指定します。
5. [OK] をクリックし、[閉じる] をクリックして、分離されたワークスペースを作成します。

最新のソースを取得して、分離したワークスペースで作業を開始するには、次の操作を行います。

1. [ソース管理エクスプローラ] で、分離したワークスペースの名前が [ワークスペース] ドロップダウン リストで選択されていることを確認します。
2. チーム プロジェクトのルート フォルダを選択し (ソース ツリーの一部だけ必要な場合はサブフォルダを選択し)、[最新バージョンを取得] をクリックします。

すると、フォルダ構造と最新のファイル セットが、ソース管理サーバーから、新しいワークスペースにマップした自身のコンピュータ上のローカル ディレクトリへとコピーされます。

参考資料

- ワークスペースを作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ワークスペースを編集する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

Windows エクスプローラではなくソース管理を使用して、ファイルの削除と名前変更を行う

ソース管理に追加されたファイルを削除または名前変更する必要がある場合は、[ソース管理エクスプローラ] を使用するか、コマンド ラインから Tf.exe を使用するか、いずれかの手段により行うことが望ましいです。削除も名前変更も、Windows エクスプローラでは行わないでください。ローカル ワークスペース ファイルと、サーバーで管理されているソース管理ファイルとの同期が外れてしまうためです。

[ソース管理エクスプローラ] を使用してファイルまたはフォルダを削除するには、次の操作を行います。

1. [ソース管理エクスプローラ] で、目的のファイルまたはフォルダを選択します。

2. 選択したファイルまたはフォルダを右クリックし、[削除] をクリックします。
削除される項目であることを示すアイコンが左側に表示され、[保留中の変更] 列に “削除” と表示されます。この項目は、次回そのファイルをチェックインしたときに削除されます。

注意 : 保留中の変更がまだ他に存在している項目は削除できません。たとえば、チェックアウトされたファイルは削除できません。

ファイルまたはフォルダを複数操作する必要がある場合は、Tf move、Tf delete、Tf rename の各コマンドが特に便利です。[ソース管理エクスプローラ] の場合は、ファイルもフォルダも一度に 1 つずつしか、移動、名前変更、削除ができません。

参考資料

- Tf delete コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/k45zb450(VS.80).aspx にある MSDN Web サイトの「Delete コマンド (Team Foundation のソース管理)」を参照してください。
- Tf rename コマンドの詳細については、http://msdn2.microsoft.com/en-us/library/a79bz90w(VS.80).aspx にある MSDN Web サイトの「Rename Command」を参照してください。

ソリューションが開いた状態でのみ削除と名前変更を行う

[ソリューション エクスプローラ] の内部では、ソリューションが開いた状態でのみファイルの削除および名前変更をしてください。この操作はディスクに対して直接は実行しないでください。このことを守れば、保留中の変更を次回チェックインするときに TFS ソース管理の同期が維持されます。

参考資料

- Tf delete コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/k45zb450(VS.80).aspx にある MSDN Web サイトの

「Delete コマンド (Team Foundation のソース管理)」を参照してください。

• Tf rename コマンドの詳細については、http://msdn2.microsoft.com/en-us/library/a79bz90w(VS.80).aspx にある MSDN Web サイトの「Rename Command」を参照してください。

アプリケーションのバージョンから別のバージョンへ資産を移動する場合に、アプリケーションごとに 1 つずつチーム プロジェクトを作成する

ソース コードだけでなく、作業項目や、その他の TFS 資産までも、リリースから別のリリースへ移行する場合は、アプリケーションごとに 1 つずつチーム プロジェクトを使用することを検討してみてください。アプリケーションの複数のバージョンにチーム プロジェクトを 1 つずつ使用していれば、TFS 資産はすべて自動的に次のリリースへと移行されます。アプリケーションの新バージョンをリリースする準備が整ったときには、そのリリースを示す分岐をプロジェクト内に作成して、コードを分離することができます。

アプリケーション 1 つにつきプロジェクトを 1 つ使用することにした場合は、以下のような不利な点もありますので、注意してください。

- 並列リリースの場合は、作業項目、チェックイン ポリシー、プロセス ガイダンスを共有しなければなりません。
- レポートの処理が難しくなります。既定では、レポートはプロジェクト全体が対象となるため、リリースごとにフィルタの処理を追加しなければならないからです。
- アプリケーションが何百もあり、それぞれが別々のプロジェクトに含まれている場合は、TFS のパフォーマンスと拡張性が限界に達します。
- リリースを繰り返すうち、”荷物” が増えていきます。この問題を解消する最も簡単な方法は、新規プロジェクトを作成して、次のリリースに移行するコードをその新規プロジェクトに分岐することです。

参考資料

• チーム プロジェクトを使用する詳細については、

http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx の「When to use Team Projects」を参照してください。

アプリケーションのバージョンごとに 1 からやり直す場合に、バージョンごとに 1 つずつチームプロジェクトを作成する

作業項目や、その他の TFS 資産を移行せずに、リリースごとに 1 からやり直す場合は、リリースごとに 1 つずつプロジェクトを使用する方法を検討してみてください。リリースごとに新規のプロジェクトを使用すると、前のリリースに影響を及ぼさずに、作業項目スキーマ、ワークフロー、チェックイン ポリシー、その他資産を変更することができます。たとえば、主要開発チームとはプロセスもワークフローも異なる可能性のある、維持管理チームのような別のチームが前のリリースを管理する場合は特に便利です。

リリース 1 つにつきプロジェクトを 1 つ使用するときは、以下の点に注意してください。

- ソース コードをプロジェクトから別のプロジェクトへ移動するのは非常に簡単ですが、作業項目やその他の TFS 資産をプロジェクトから別のプロジェクトへ移動するのは難しいです。作業項目は、一度に 1 つずつしか別のプロジェクトにコピーできないため、いくつかまとめて作業項目をコピーしたい場合は、ユーティリティを作成する必要があります。
- アプリケーションとリリースが何百もあり、それぞれが別々のプロジェクトに含まれている場合は、TFS のパフォーマンスと拡張性が限界に達します。
- 既存のチーム プロジェクトの構造を変えるのは難しいため、長い期間利用できる構造を選んでください。
- ソースは次のようにチーム プロジェクトどうしで簡単に共有できます。
    - プロジェクトから別のプロジェクトへソースを分岐します。
    - 別のプロジェクトからワークスペースへソースをマップします。
- Team Foundation Server で対応できるプロジェクトの個数は、MSF Agile (Microsoft Solution Framework for Agile Software Development) プロセス テンプレートを使用した場合は 500 個、MSF CMMI プロセス テンプレートを使用した場合は 250 個まで拡張できます。独自のプロセスを作成する場合、または既存のプロセスをカスタマイズする場合は、サーバーの拡張性に最も大きく影響するのが作業項目スキーマであることを忘れないでください。スキーマが複雑だと、サ

ーバーで対応できるプロジェクトの個数が減ります。

参考資料

- チーム プロジェクトを使用する詳細については、http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx の「When to use Team Projects」を参照してください。

統合テストを必要とするコードとバイナリを共有するために分岐を使用する

共有コードまたは共有バイナリの管理には次の 2 つの手順があります。

1. 依存関係の格納先となるロケーションを決定します。
2. 依存関係を自身のプロジェクトに分岐します。

1. **依存関係の格納先となるロケーションを決定します。**

格納用のオプションは次のとおりです。

- 共有依存関係が特定のチームに所有されていることがはっきりしている場合は、その共有依存関係をそのチームのチーム プロジェクトに格納してください。
- 明確な所有権が共有依存関係にない場合は、共有コード専用のチーム プロジェクトを作成してください。

2. **依存関係を自身のプロジェクトに分岐します。**

依存関係の格納が済んだら、共有ソースまたは共有バイナリを自身のプロジェクトに分岐してください。共有プロジェクトから利用するプロジェクトへマージを実行するたびに、最新のソースが取得されます。この方法だと定期的に変更が取得可能で、メインのソース ツリーに影響が及ぶ前に統合テストが実行できます。

参考資料

- 分岐およびマージの概要については、

http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。

- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐とマージの実行方法の補足については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「”Team Foundation ソース管理の分岐およびマージ」を参照してください。

プロジェクトどうしの依存関係に対応できるよう、ワークスペース マッピングは避ける

一般的に、複数のチーム プロジェクトを横断するような依存関係は避け、関連のあるソリューション/プロジェクトもしくは依存しているソリューション/プロジェクトはすべて、同じチーム プロジェクトの下に置いておく方がよいです。そうすれば、ビルド スクリプトをカスタマイズする必要性が減ります。依存関係がある場合は、その依存関係の定義にプロジェクト参照を使用するか、共有プロジェクトから自身のプロジェクトへその依存関係を分岐するか、いずれかを行ってください。ファイル参照は管理が難しいため、使用しない方がよいです。例外は、依存プロジェクトを並行して開発していて、リアルタイムに変更が必要な場合です。この場合は、ワークスペース マッピングの使用を検討してもよいです。しかし、依存コードが原因で大きな変更が多く発生しすぎるような場合であれば、分岐を使用して分離バッファを作成することは可能です。

参考資料

- ワークスペースを作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 ： ワークスペースを作成する」を参照してください。
- ワークスペースを編集する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 ： ワークスペースを編集する」を参照してください。

チーム プロジェクトのルート レベルでワークスペース マッピングを作成する

新しいチーム プロジェクトの場合は、チーム プロジェクトのルート ($/MyTeamProject) をローカルドライブ上で同じ名前のフォルダ (C:¥TeamProjects など) にマップしてください。マッピングは再帰的に行われるため、ローカル フォルダ構造は、全体が自動的に作成され、ソース管理における構造とまったく同じになります。

参考資料

- ワークスペースを作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ワークスペースを編集する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

共有コンピュータ上で、他と重複しないローカル フォルダ パスを使用する

1 台のコンピュータを 2 人のユーザーで使用している場合、その 2 人のユーザーは、同じワークスペース マッピングを共有することはできません。たとえば、自分と同僚の 2 人は、同じチーム プロジェクト ($/MyTeamProject) をローカル コンピュータ上の同じフォルダにマップすることはできません。その代わりに、(ロケーション パスが長くなりますが) [マイ ドキュメント] の下にマッピングを作成するか、ローカル コンピュータ上でチーム フォルダの命名規則 (C:¥TeamProjects¥User1、C:¥TeampProjects¥User2 など) を設定するか、いずれかを行ってください。

参考資料

- ワークスペースを作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。

- ワークスペースを編集する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

ソース ツリーの一部だけをマッピングしてみる

パフォーマンスを上げてディスクサイズの要件を減らすため、開発プロジェクトに必要なファイルだけをマップしてください。一般的に、必要になるのは作業を行うソリューションに関連付けられているファイルと、プロジェクトだけです。

参考資料

- ワークスペースを作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ワークスペースを編集する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

分岐に対応できる構造をしたソース ツリーを作成する

ソース ツリー構造は、フォルダ構造、ファイル構造、分岐構造の 3 つで構成されています。メイン分岐の内部では、以下のフォルダとファイルの構造がさまざまな規模のチームで機能することが既に証明されています。

- **Main :** 製品の出荷に必要なすべての資産用のコンテナ
    - o **Source :** ビルドに必要なすべてのもの用のコンテナ
        - □**Code :** ソース コード用のコンテナ
        - □**Shared Code :** 他のプロジェクトから共有されるソース コード用のコンテナ
        - □**Unit Tests :** 単体テスト用のコンテナ

- □**Lib :** バイナリ依存関係用のコンテナ
- o **Docs :** 製品と一緒に出荷する文書用のコンテナ
- o **Installer :** インストーラのソース コードおよびバイナリ用のコンテナ
- o **Tests :** テスト チームのテスト ケース用のコンテナ

Main を起点に作成した分岐によって、このフォルダとファイルの構造が、たとえば次のように新しい分岐へとコピーされます。

- **Development :** 開発分岐
  - o **Source :** ビルドに必要なすべてのもの用のコンテナ
    - □**Code :** ソース コード用のコンテナ
    - □**Shared Code :** 他のプロジェクトから共有されるソース コード用のコンテナ
    - □**Unit Tests :** 単体テスト用のコンテナ
    - □**Lib :** バイナリ依存関係用のコンテナ
- **Main :** 統合分岐
  - o **Source :** ビルドに必要なすべてのもの用のコンテナ
    - □**Code :** ソース コード用のコンテナ
    - □**Shared Code :** 他のプロジェクトから共有されるソース コード用のコンテナ
    - □**Unit Tests :** 単体テスト用のコンテナ
    - □**Lib :** バイナリ依存関係用のコンテナ
  - o **Docs :** 製品と一緒に出荷する文書用のコンテナ
  - o **Installer :** インストーラのソース コードおよびバイナリ用のコンテナ
  - o **Tests :** テスト チームのテスト ケース用のコンテナ

参考資料

- ワークスペースを作成する詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ワークスペースを編集する詳細については、

http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

シェルブ

- **シェルブを使用して、保留中の変更を共有し、レビューまたはハンドオフできるようにする**
- **シェルブを使用して、保留中の変更をサーバーにバックアップする**
- **優先度の高い作業によって中断された場合にシェルブを使用する**

シェルブを使用して、保留中の変更を共有し、レビューまたはハンドオフできるようにする

作業中のコードについて、遠隔地にいるチーム メンバを相手に議論がしたい場合、あるいはそのメンバにコードをレビューしてほしい場合は、シェルブを使用してください。コードを電子メールで送信しなくても、コードをシェルブして (つまり “棚” に載せて)、遠隔地にいる同僚はその ”棚” から目的のファイルを取り出すことができます。

同じように、一部完了した作業を別の開発者に渡して完成させる場合も、シェルブを使用してコードを渡すことができます。

参考資料

- 保留中の変更をシェルブする詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

シェルブを使用して、保留中の変更をサーバーにバックアップする

終業時になってもまだ作業が終らず、現在の作業をサーバーにバックアップしておきたい場合は、シェルブを使用してください。現在の変更をシェルブすると、その変更が TFS サーバーに適用され、翌日その変更を自分 (または自分以外) が取り出すことができます。

参考資料

- 保留中の変更をシェルブする詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

優先度の高い作業によって中断された場合にシェルブを使用する

優先度の高い新しい作業を割り当てるときに、一連のソース コードを変更している途中だった場合は、シェルブを使用してください (たとえば、直ちにバグを修正しなければならないような場合)。このとき、安定したバージョンのコードに戻る必要がありますが、変更内容は失いたくありません。操作をする前に、コードをシェルブしておけば、後で簡単に取り出すことができます。

参考資料

- 保留中の変更をシェルブする詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

ソース管理に関する参考資料

- Team Foundation Server ソース管理の詳細については、http://msdn2.microsoft.com/en-us/library/ms181237(VS.80).aspx の「Team Foundation Source Control」を参照してください。

プラクティス: チーム ビルド

索引

管理

- ビルド サーバーを保護する方法
- ビルドの削除方法
- ビルドの種類の削除方法
- 作業項目をビルドに関連付ける方法

チェックイン ポリシー

- チェックイン品質の改善のためのチェックイン ポリシーの使用方法
- 作業項目をビルドに関連付けるためのチェックイン ポリシーの使用方法

継続的な統合ビルド

- 継続的な統合 *(CI)* ビルドを自動的に実行する方法
- ローリング ビルドが必要かどうかの判別方法
- ローリング ビルドのサイクル間隔の判別方法

カスタマイズ

- ビルド番号の変更方法
- ツリーのサブセットの取得と作成のためのワークスペースのマッピングのセットアップ方法
- 別のチーム プロジェクトへの依存関係を持ったプロジェクトの作成方法
- ビルド構成の変更方法 *(Release/Debug)*

配置

- ビルド サーバーのセットアップ方法
- 複数のビルド サーバーが必要かどうかの判別方法

一般

- チーム ビルドを使って *ASP.NET* アプリケーションをビルドおよび配置する方法
- チーム ビルドを使った *Microsoft® .NET 1.1* アプリケーションのビルド方法
- チーム ビルドを使ったセットアップ/配置プロジェクトのビルド方法
- チーム ビルドの作成方法
- 複数のビルドの種類の作成方法
- 別のプロジェクトのアセンブリを参照するプロジェクト用のチーム ビルドの作成方法
- 電子メールのビルド イベントのサブスクライブ方法
- ビルドの失敗時に通知を受け取る方法
- ビルドの開始方法
- ビルドの正常完了の検証方法
- ビルド出力の表示方法
- ビルド サーバーの場所の変更方法
- ビルドの出力場所の変更方法
- どの変更セットがビルドの一部を成すかの判別方法
- 報告されたビルドの品質の変更方法

プロジェクト

- シングル ソリューション戦略の使用方法
- パーティション ソリューション戦略の使用方法
- 複数ソリューション戦略の使用方法

レポート

- ビルド品質の表示方法
- ビルドのすべてのチェックインの表示方法
- ビルドの終了済み作業項目またはバグの表示方法
- ビルドの未完了の作業項目またはバグの表示方法
- ビルドからビルドへの進行速度のトラッキング方法
- テスト ケースでのビルドの合格/不合格結果のトラッキング方法
- ビルド状況の検討方法 *(BVT* 結果*)*

スケジュール ビルド

- ナイトリー ビルドを自動的に実行する方法
- プロジェクトでのビルドの頻度と種類の決定方法

テストを軸とした開発

- 「*hello world*」受け入れテストの作成方法
- ビルドの一環としての自動テストの実行方法
- ビルドの一環としてのコード分析の実行方法
- テストの失敗時にビルドを失敗させる方法

管理

- **ビルド サーバーを保護する方法**
- **ビルドの削除方法**
- **ビルドの種類の削除方法**
- **作業項目をビルドに関連付ける方法**

ビルド サーバーを保護する方法

ビルド サーバーを保護するには

1. ビルド サービスは、別個のサーバーに配置します。つまり、Microsoft Visual Studio® 2005 Team Foundation Server (TFS) のアプリケーション層やデータ層とサーバーを共有しないようにします。
2. ビルド ディレクトリへの読み取り/書き込みアクセス権をビルド プロセスに認可します。ビルドを実行するアカウントが、必ずこのディレクトリにアクセスできるようにします。
3. ビルド ドロップ ネットワーク共有への読み取り/書き込みアクセス権をビルド プロセスに認可します。ビルドを実行するアカウントが、必ずこの共有にアクセスできるようにします。
4. チーム ビルドの実行で使用するアカウントが、チーム プロジェクトの **Build Services** グループのメンバであることを確認します。

Team Foundation Server のセキュリティを向上するためには、アプリケーション層やデータ層上ではなく、独自のコンピュータ上にビルド サーバーをインストールする必要があります。配置またはビルドのステップによっては、上位特権が必要になることがあります。たとえば、Web アプリケーションを配置するための仮想ディレクトリを作成するには、ビルド サーバーでの管理者権限が必要です。つまり、ビルドを実行する Microsoft Windows® アカウントが、この権限を持っている必要があるということです。ビルド コンピュータがアプリケーション層上に置かれていると、セキュリティ上のリスクを負う可能性があります。同様に、ビルド コンピュータがデータ層上に置かれていると、ビルド アカウントは、この層上のデータベースにアクセスして変更することができます。

注意: セキュリティ上の理由から、チーム ビルドを実行するアカウントを SERVER¥Service アカウント グループに加えないでください。このグループのメンバは、TFS における完全な管理者権限を持つからです。

参考資料

- TFS グループと許可の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253077(VS.80).aspx の「Team Foundation Server の既定のグループ、アクセス許可、およびロール」を参照してください。

ビルドの削除方法

ビルドを削除するには、TFSBuild コマンド ライン ツールを使用します。TFS サーバーのアドレス、チーム プロジェクトの名前、およびビルド名を指定します。以下に例を示します。

TfsBuild delete http://mytfsserver:8080 myproject build20070606.4

参考資料

- 完了したビルドの削除の詳細については、http://msdn2.microsoft.com/ja-jp/library/aa337656(VS.80).aspx の「方法 : 完了したビルドを削除する」を参照してください。
- Delete コマンドの詳細については、

http://msdn2.microsoft.com/ja-jp/library/ms244360(VS.80).aspx の「Delete コマンド（Team Foundation ビルド)」を参照してください。

ビルドの種類の削除方法

チーム エクスプローラを使用して、チーム ビルドの種類を削除することはできません。代わりに、ビルドの種類をソース管理から除去する必要があります。

既存のビルドの種類を削除するには

1. [ソース管理エクスプローラ] を開きます。
2. [ソース管理エクスプローラ] で、チーム プロジェクト フォルダを展開します。
3. [TeamBuildTypes] フォルダを展開します。
4. 削除したいチーム ビルドの種類を表すチーム ビルド フォルダを右クリックしてから、[**削除**] をクリックします。
5. チーム ビルド フォルダをもう一度クリックしてから、[**保留中の変更のチェックイン**] をクリックします。
6. [チーム エクスプローラ] を開きます。
7. チーム ビルド フォルダを右クリックしてから、[**最新の情報に更新**] をクリックします。
8. チーム ビルド フォルダを展開し、チーム ビルドが削除済みであることを確認します。

参考資料

- チーム ビルドの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181710(vs.80).aspx の「Team Foundation ビルドの概要」を参照してください。

作業項目をビルドに関連付ける方法

作業項目をチェックインに関連付けるには、[**チェックイン**] ダイアログ ボックスを使用します。それによって、その作業項目が次のビルドに自動的に関連付けられます。

作業項目をビルドに関連付けるには

1. ビルドに組み入れて作業項目に関連付けようとしている変更をコードに加えます。
2. 保留中の変更をチェックインします。
3. [**チェックイン**] ダイアログ ボックスで[**作業項目**] をクリックします。
4. そのチェックインに関連付けたい 1 つ以上の作業項目を選択します。

最後に正常に完了したビルド以降に発生したすべての変更セットが、次のビルドに関連付けられます。次のビルド後に、このビルドに関連した変更セットとしてこの変更セットがチーム ビルドにリストされます。また、その変更セットに関連付けられた作業項目として、選択した作業項目が組み入れられます。

参考資料

- 保留中の変更チェックインの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181411(VS.80).aspx の「方法 : 保留中の変更をチェックインする」を参照してください。

チェックイン ポリシー

- **チェックイン品質の改善のためのチェックイン ポリシーの使用方法**
- **作業項目をビルドに関連付けるためのチェックイン ポリシーの使用方法**

チェックイン品質の改善のためのチェックイン ポリシーの使用方法

チェックインの品質を改善するには、コード分析とテスト ポリシーを組み合わせて使用します。たとえば、TFS ソース管理にソースをチェックインする前に、ある特定のテストが必ず実行されて合格するようにするには、既定のテスト チェックイン ポリシーを使用します。また、コード分析ポリシーを構成して、ある特定の品質基準をコードが必ず満たすようにすることもできます。それには、セキュリティ、パフォーマンス、移植性、保守容易性、および信頼性の規則が必ず守られるようにします。

チーム プロジェクトでコード分析チェックイン ポリシーを実施するには

1. [チーム エクスプローラ] でチーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] を選択してから、[**ソース管理**] をクリックします。
2. [**チェックイン ポリシー**] タブをクリックします。
3. [**追加**] をクリックしてから、コード分析およびテスト ポリシーを構成します。

参考資料

- カスタム チェックイン ポリシーの作成と使用方法の詳細については、このガイドの「方法 ：TFS のカスタム チェックイン ポリシーを作成するステップ」を参照してください。
- チェックイン ポリシーをカスタマイズする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインで特定のパターンを禁止するサンプル コードの詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインでコメントを強制するサンプル コードについては、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーを登録する方法の詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。

作業項目をビルドに関連付けるためのチェックイン ポリシーの使用方法

各チェックインが必ず作業項目に関連付けられるようにするには、チェックイン ポリシーを使用します。開発者は [**チェックイン**] ダイアログ ボックスを使用して、作業項目をチェックインに関連付けます。それによって、その作業項目が次のビルドに自動的に関連付けられます。

開発者によって必ずチェックインが作業項目に関連付けられるように作業項目チェックイン ポリシーを設定するには

1. [チーム エクスプローラ] でチーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] を選択してから、[**ソース管理**] をクリックします。
2. [**チェックイン ポリシー**] タブをクリックします。
3. [**追加**] をクリックしてから、[**作業項目**] チェックイン ポリシーを選択して構成します。

参考資料

- カスタム チェックイン ポリシーの作成と使用方法の詳細については、このガイドの「方法 ：TFS のカスタム チェックイン ポリシーを作成するステップ」を参照してください。
- チェックイン ポリシーをカスタマイズする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。

継続的な統合ビルド

- **継続的な統合 (CI) ビルドを自動的に実行する方法**
- **ローリング ビルドが必要かどうかの判別方法**
- **ローリング ビルドのサイクル間隔の判別方法**

継続的な統合ビルドを自動的に実行する方法

Visual Studio 2005 Team Foundation Server には、すぐに使用できる CI ソリューションは用意されていませんが、独自の継続的な統合 (CI) ソリューションを実装するためのフレームワークは用意されています。

CI ビルド ソリューションをセットアップするには

1. **ビルドを作成してテストします。**使用できるチーム ビルドの種類があって、エラーなしで実行できることを確認します。

2. **継続的な統合アドオンをインストールします。**CI アドオンを http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx からインストールします。
3. **継続的な統合アドオンを構成します。**CI Web アプリケーションの仮想ルートが **TFSAppPool** アプリケーション プールで稼働していることを確認します。CI Web アプリケーションの Web.config ファイルを更新して、チーム サーバーおよびチーム ビルドと一緒に稼働するようにします。そのためには、次のようなキーを追加します。

```
<add key="1"
value="TeamServer=http://TFSRTM:8080;TeamProjectName=AdventureWorks;BuildType=Test
Build"/>
```

4. **CheckinEvent イベントの登録を行います。BisSubscribe.exe** ツールを使用して、チェックイン イベントの登録を行います。それには、次のようなコマンド ラインを使用します。

```
Bissubscribe /eventType CheckinEvent /address http://TFSRTM:8080/ci/notify.asmx /deliveryType Soap
/domain http://TFSRTM:8080
```

5. **CI ビルドをテストします。**ビルドをテストするには、チェックインを完了します。ビルドが完了するまで数分待ってから、ビルド ページを表示して、ビルドが正常に完了したかどうかを確認します。
6. **電子メール アラートをセットアップします。**ビルドの完了時に関係者に通知されるようにアラートをセットアップします。チーム プロジェクトのコンテキスト メニューから [**プロジェクト警告**] ダイアログ ボックスを開いてから、[**ビルドが完了したとき**]アラート チェック ボックスを選択します。

さらに詳細な情報と拡張ステップについては、このガイドの「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。

参考資料

- 詳細については、このガイドの「第 8 章 – チーム ビルドでの継続的な統合のセットアップ」を参照してください。
- CI ビルドのセットアップの詳細については、このガイドの「Visual Studio Team Foundation

Server でスケジュール ビルドをセットアップする方法」を参照してください。

- Visual Studio Team System の CI ソリューションの使用方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms364045(VS.80).aspx の「Continuous Integration Using Team Foundation Build」を参照してください。
- Visual Studio Team System の CI ソリューション MSI をダウンロードするには、http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi にアクセスしてください。
- TFS におけるアジャイル開発および CI の詳細については、http://www.microsoft.com/japan/msdn/msdnmag/issues/06/03/TeamSystem/default.aspx の「常時結合を可能にするための Team Foundation Server の拡張」を参照してください。

ローリング ビルドが必要かどうかの判別方法

チェックインごとに、直後にビルドを行うのが最も簡単な CI 戦略であり、一般的にその場合に、最も迅速なフィードバックが得られます。ただし、ビルド サーバーにまったく余裕がなくなるほど頻繁にチェックインが発生する場合は、ローリング ビルド アプローチをとる必要があります。つまり、指定数のチェックインの後か、または指定期間の後で、ビルドを行います。ローリング ビルドを使用する必要があるかどうかを確かめるには、以下の点を判別します。

- チーム ビルドに要する分単位の時間
- チェックインの分単位の平均頻度
- チェックインが頻繁に発生する時間枠

ビルドに要する時間がチェックインの平均頻度より大きい場合、ビルドは連続して実行されています。なぜなら、最初のビルドが完了しないうちに次のチェックインが発生し、それによって別のビルドが開始されているからです。各ビルドが完了する前にチェックインの発生が継続的に起きる場合、ビルド サーバーのパフォーマンスが影響を受けるので、スケジュール ビルドなどの他のビルドの開始が妨げられます。チェックインが頻繁に発生する時間枠を見直して、スケジュール ビルドやその他の重要なチーム ビルドの配信に、CI ビルドが影響を与えていると見なしてよいかどうかを判別します。

参考資料

- 詳細については、このガイドの「第 8 章 – チーム ビルドでの継続的な統合のセットアップ」を参照してください。

ローリング ビルドのサイクル間隔の判別方法

ローリング ビルドのサイクル間隔を究明して、ビルド プロセスの効率化を図ることが重要です。ビルド プロセスに負荷をかけすぎずに、タイミングよくビルドを行うことができます。

ローリング ビルドの理想的なサイクル間隔を判別するには、チェックインの平均頻度をビルドの所要時間で割り算します。たとえば、10 分を要するビルドがあったとして、チェックインの平均頻度が 5 分ごとであると仮定すると、チェックインの間隔を 2 つのチェックインに設定し、タイムアウト期間を 10 分に設定することができます。このようにすれば、ビルドが完了してから、次のビルドが開始されるようにすることができます。ビルド サーバーに負荷がかかり過ぎていると判明した場合には、この値を増大することができます。

参考資料

- 詳細については、このガイドの「第 8 章 – チーム ビルドでの継続的な統合のセットアップ」を参照してください。

カスタマイズ

- **ビルド番号の変更方法**
- **ツリーのサブセットの取得と作成のためのワークスペースのマッピングのセットアップ方法**
- **別のチーム プロジェクトへの依存関係を持ったプロジェクトの作成方法**
- **ビルド構成の変更方法 (Release/Debug)**

ビルド番号の変更方法

コンパイル済みのアセンブリでビルド番号をカスタマイズするには、置換するビルド番号を生成してか

ら、それを assemblyinfo ソース ファイルに書き込む必要があります。
チーム ビルド インターフェイスに表示されたビルド番号のプロパティをカスタマイズするには、**BuildNumberOverride** ターゲット内の $(**BuildNumber**) プロパティをオーバーライドする必要があります。

アセンブリとチーム ビルド インターフェイスの両方でビルド番号をカスタマイズするには

1. **BuildNumberOverride** ターゲット内の **$(BuildNumber)** をオーバーライドします。
2. **BeforeCompile** ターゲットをオーバーライドして、AssemblyInfo.cs または .vb ファイルを書き込みます。

例

```
<Target Name="BuildNumberOverrideTarget">
 <Message Importance="High" Text="$(BuildNumber)" />
 <ConvertTFSBuildNumberToSolutionBuildNumber
  MajorAndMinorVersion="1.0"
   TFSBuildNumber="$(BuildNumber)"
   TFSLastBuildNumber="$(LastBuildNumber)">
   <Output TaskParameter="SolutionBuildNumber" PropertyName="SolutionBuildNumber" />
   <Output TaskParameter="TFSBuildNumber" PropertyName="BuildNumber" />
  </ConvertTFSBuildNumberToSolutionBuildNumber>
  <Message Importance="High" Text="$(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>
```

参考資料

- アセンブリ バージョン変更の別の方法の詳細については、http://blogs.msdn.com/aaronhallberg/archive/2007/06/08/team-build-and-the-assemblyinfo-task.aspx の Aaron Halberg 氏のブログ ポスト「Team Build and the AssemblyInfo Task」を参照してください。
- 上記のブログ ポストで参照されている AssemblyInfo タスクには、GotDotNet 上に新しいホームがあります。それは、http://www.gotdotnet.com/Community/UserSamples/Details.aspx?SampleGuid=5C455590-332C-433B-A648-E368B9515580 に置かれています。
- Team Foundation ビルドのカスタマイズの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400688(VS.80).aspx の「Team Foundation ビルドのカスタマイズ」を参照してください。

ツリーのサブセットの取得と作成のためのワークスペースのマッピングのセットアップ方法

ワークスペース マッピング ファイルには、ビルド サーバーによって取り出されるソース管理フォルダが定義されています。ビルドの実行のために、常にすべてのファイルをチェックアウトする必要があるとは限りません。ワークスペースの定義を変更して、取り込むフォルダ数を減らすことができます。あるいは、不要なファイルをクローク (覆い隠) して、ビルドの一環として取得されないようにすることもできます。

たとえば、新しいプロジェクトの既定のマッピングは $/TeamProject となります。ソース ファイルがすべて $/TeamProject/foo/bar/foobar/sources の下にあるとすると、このディレクトリをマップするだけで済みます。

ファイルおよびフォルダをクロークするには

1. WorkspaceMapping.xml をソース管理からチェックアウトします。
2. 該当するクローク項目をこのファイルに追加します。
3. WorkspaceMapping.xml をチェックインします。

WorkspaceMapping.xml ファイルは次のようになります。

```
<Mappings>
  <InternalMapping ServerItem="$/MyTeamProject"
LocalItem="c:¥projects¥teamproject” Type="Map" />
  <InternalMapping ServerItem="$/MyTeamProject/documentation"
Type="Cloak" />
</Mappings>
```

参考資料

- ワークスペースでのフォルダのクロークの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181378(VS.80).aspx の「方法 : ワークスペースでフォルダをクロークまたはクローク解除する」を参照してください。
- チーム ビルドでフォルダを無視する方法の詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/10/13/480584.aspx の「How to make 'Team Build' skip getting certain folders?」を参照してください。
- WorkspaceMapping スキーマの詳細については、http://blogs.msdn.com/buckh/archive/2007/02/28/schema-for-the-workspacemapping-xml-file.aspx の「Schema for the WorkspaceMapping.xml file」を参照してください。

別のチーム プロジェクトへの依存関係を持ったプロジェクトの作成方法

チーム ビルドは、複数のチーム プロジェクトを対象とするソリューションのビルドをサポートしません。それを可能にするには、ビルドが依存する他のプロジェクトから、必要なコードをチェックアウトできるように、TFSBuild.proj ファイルをカスタマイズする必要があります。

別のチーム プロジェクトに対する依存関係を持ったプロジェクトをビルドするには、そのプロジェクトからビルド サーバー上のワークスペースにソースまたはアセンブリを取り込む必要があります。そのためには、TFSBuild.proj ファイルを編集してアセンブリまたはソリューションの参照を追加し、**BeforeGet** イベントをオーバーライドして、依存先の各チーム プロジェクトからアセンブリまたはソリューションを取り込む必要があります。

別のチーム プロジェクトへの依存関係を持ったプロジェクトを作成するには

1. TFSBuild.proj スクリプトをソース管理エクスプローラからチェックアウトします。
2. 以下の構成設定を **PropertyGroup** セクションの下に追加します。

```
<!-- PropertyGroup の下に追加します -->
<TfCommand>$(TeamBuildRefPath)¥..¥tf.exe</TfCommand>
<SkipInitializeWorkspace>true</SkipInitializeWorkspace>
```

   SkipInitializeWorkSpace を使用すれば、ビルド マシン上でワークスペースを削除および再作成する既定のタスクを起動しなくて済みます。新しいプロパティが、カスタム ターゲット **BeforeGet** で使用されています (以下を参照)。
3. チーム プロジェクトとソリューションの両方をマップする **ItemGroup** 項目に、以下の構成設定を追加します。ビルド マシンの正しいローカル パスを間違いなく指定します。複数のマッピングによって、同じローカル フォルダを共有することはできません。そのようにした場合、CreateWorkspace タスクで **MappingConflictException** 例外になってしまいます。

```
<ItemGroup>
 <!-- 参照する各ソリューションごとに項目を 1 つずつ追加します -->
 <SolutionToBuild Include="$(SolutionRoot)¥DependentApp¥DependentApp.sln" />
 <SolutionToBuild Include="$(SolutionRoot)¥YourApp¥YourApp.sln" />
</ItemGroup>

<ItemGroup>
  <!-- 参照する各チーム プロジェクトごとに項目を 1 つずつ追加します -->
  <Map Include="$/YourApp/YourApp">
```

```
    <LocalPath>$(SolutionRoot)¥YourApp</LocalPath>
   </Map>
   <Map Include="$/DependentApp/DependentApp">
    <LocalPath>$(SolutionRoot)¥DependentApp</LocalPath>
   </Map>
  </ItemGroup>
```

4. 次のように、**BeforeGet** イベントをオーバーライドして、各チーム プロジェクトのワークスペースを取り出します。

```
<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:$(WorkSpaceName)
&quot;$(SolutionRoot)&quot;"/>
<Exec
WorkingDirectory="$(SolutionRoot)"
Command="&quot;$(TfCommand)&quot; workfold /map /workspace:$(WorkSpaceName)
/server:$(TeamFoundationServerUrl) &quot;%(Map.Identity)&quot;
&quot;%(Map.LocalPath)&quot;"/>
</Target>
```

5. ビルド スクリプトをチェックインし、ビルドを実行します。

参考資料

- 詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の Manish Agarwal 氏のブログ エントリ「Working with multiple team projects in Team Build」

を参照してください。

ビルド構成の変更方法 *(Release/Debug)*

既存のビルドにおいてビルド構成を変更するには、TFSBuild.proj 内の **<ConfigurationToBuild>** タグを変更する必要があります。

ビルド構成を変更するには

1. [ソース管理エクスプローラ] を開きます。
2. [ソース管理エクスプローラ] で、チーム プロジェクト フォルダを展開します。
3. [TeamBuildTypes] フォルダを展開します。
4. コード分析をオンにするチーム ビルド フォルダを選択します。
5. TFSBuild.proj ファイルをソース管理からチェックアウトします。必要により先にフォルダに対して [**最新バージョンの取得**] 操作を実行してください。
6. [ソース管理エクスプローラ] で、TFSBuild.proj をダブルクリックして開きます。
7. **<ConfigurationToBuild>** タグを新しいビルド構成に変更します。
8. TFSBuild.proj を上書き保存し、元のソース管理にチェックインします。

参考資料

- Team Foundation ビルドのカスタマイズの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400688(VS.80).aspx の「Team Foundation ビルドのカスタマイズ」を参照してください。

配置

- **ビルド サーバーのセットアップ方法**
- **複数のビルド サーバーが必要かどうかの判別方法**

ビルド サーバーのセットアップ方法

ビルド サーバーは、TFS のインストールとは別にインストールします。ビルド サーバーは、コードのコンパイル、テストの実行、およびコード分析の実行に対応できる必要があるので、ビルド プロセスで必要になるすべてのツールをインストールする必要があります。

ビルド サーバーをセットアップするには

1. Visual Studio をインストールします。
    - どのようなビルド シナリオにも必要なツールをすべて揃えておきたい場合、Team Suite 全体をインストールする必要があります。
    - チーム ビルドを実行する予定はあっても、テスト ケースを実行する必要はない場合、Visual Studio Team Developer Edition をインストールします。
    - 自動化されたテスト ケースを、ビルド プロセスの一環として実行したい場合、Visual Studio Team Test Edition をインストールします。
2. Team Foundation Server の DVD 上の ¥build フォルダを開きます。
3. セットアップ ウィザードを実行します。

ビルド サーバーの実行で使用するアカウントの詳細については次のとおりです。

- TFS コンピュータに**ローカルでログオンする**許可を持っていなければなりません。
- TFS コンピュータ上のローカル管理者アカウントであってはなりません。
- ドメイン上の Microsoft Active Directory® で [**アカウントは重要なので委任できない**] のマークが付いていなければなりません。

参考資料

- 「*Team Foundation Server* インストール ガイド」を http://www.microsoft.com/downloads/details.aspx?familyid=e54bf6ff-026b-43a4-ade4-a690388f310e&displaylang=en からダウンロードすることができます。

複数のビルド サーバーが必要かどうかの判別方法

すべて 1 つのビルド サーバー上で実行する複数のビルドの種類を扱う場合、ビルド サーバーをオーバーロードすることができます。これが問題になった場合、それぞれ異なるビルド サーバーでのそれぞれ異なるビルドの種類の実行を検討してみてください。

ビルドの実行には長時間を要します。大型プロジェクトのビルドの場合は特にそうです。継続的な統合または頻度の高いスケジュールされたビルドを使用する場合、生成されるビルド量にビルド サーバーが追いつけなくなる可能性があります。複数のビルド サーバーをインストールして、負荷を分散することができます。それぞれ異なるビルドの種類を各サーバーに割り当てて、ビルドの負荷を均一化します。

一般

- **チーム ビルドを使用して ASP.NET アプリケーションをビルドおよび配置する方法**
- **チーム ビルドを使用した .NET 1.1 アプリケーションのビルド方法**
- **チーム ビルドを使用したセットアップ/配置プロジェクトのビルド方法**
- **チーム ビルドの作成方法**
- **複数のビルドの種類の作成方法**
- **別のプロジェクトのアセンブリを参照するプロジェクト用のチーム ビルドの作成方法**
- **電子メールのビルド イベントのサブスクライブ方法**
- **ビルドの失敗時に通知を受け取る方法**
- **ビルドの開始方法**
- **ビルドの正常完了の検証方法**
- **ビルド出力の表示方法**
- **ビルド サーバーの場所の変更方法**
- **ビルドの出力場所の変更方法**
- **どの変更セットがビルドの一部を成すかの判別方法**
- **報告されたビルドの品質の変更方法**

チーム ビルドを使用して *ASP.NET* アプリケーションをビルドおよび配置する方法

ASP.NET Web アプリケーションのみの入ったソリューションをビルドしたい場合、それに適した構成を必ず選択する必要があります。ビルドの種類を作成する場合に、構成およびプラットフォームを選択するには、プラットフォームが **.NET** に設定されていて、[**構成**] が [**デバッグ**] に設定されていることを確認します。

ASP.NET Web アプリケーション プロジェクトのみの入ったソリューションをビルドするには

1. [**新しいチーム ビルドの種類の作成ウィザード**] を実行します。
2. ビルドに名前を付けます。
3. ASP.NET Web アプリケーションのみが入ったソリューションを選択します。
4. 該当する構成を選択します。
5. [**プラットフォーム**] ドロップダウン リストをクリックし、プラットフォームとして **.NET** を選択します。
6. ビルドの種類の場所情報を指定します。
7. 実行するテストおよびコード分析ルールを選択します。
8. [**完了**] を選択して、ビルドの種類を上書き保存します。

ASP.NET Web アプリケーションとその他の .NET プロジェクトの両方が入っているソリューションをビルドする場合、プラットフォーム タイプを [**Mixed Platforms**] に設定する必要があります。

ASP.NET Web アプリケーションとその他の .NET プロジェクトの入ったソリューションをビルドするには

1. [**新しいチーム ビルドの種類の作成ウィザード**] を実行します。
2. ビルドに名前を付けます。
3. ASP.NET Web アプリケーションとその他のプロジェクトのみが入ったソリューションを選択します。
4. 該当する構成を選択します。
5. [**プラットフォーム**] ドロップダウン リストをクリックし、プラットフォームとして **Mixed** を選択します。
6. ビルドの種類の場所情報を指定します。
7. 実行するテストおよびコード分析ルールを選択します。

8. [**完了**] を選択して、ビルドの種類を上書き保存します。

コンパイル済みのバイナリは、{BuildType}¥{Configuration Name}¥{Platform}¥_PublishedWebsites の下のドロップ場所に置かれます。

インターネット インフォメーション サービス (IIS) への Web アプリケーションの配置は、チーム ビルドではもともとサポートされていません。チーム ビルドの一環としてアプリケーションを IIS に配置したい場合には、2 つの選択肢があります。1 つは、ビルドの種類へのカスタム ステップの追加であり、もう 1 つは、Web 配置プロジェクトの使用です。

チーム プロジェクトの開始時点で、Web 配置プロジェクト オプションを考察して、自身の開発でこのオプションが利用できるかどうかを確かめてください。既存の Web サイトがある場合に、Web 配置プロジェクトを使用すると、アプリケーション開発が混乱する可能性があります。それに代えて、MSBuild のビルド後ステップを使用してください。どちらの場合も、ビルドの実行で使用するサービス アカウントが、IIS 内で仮想ディレクトリを作成できるローカルの Administrators グループのメンバであることを確認する必要があります。

ビルド後ステップを使用して Web アプリケーションを配置するには

1. http://www.codeplex.com/sdctasks の Codeplex から SDC Tasks Library をダウンロードします。
2. Web アプリケーションの配置で使用しようとしているチーム ビルドの種類をチェックアウトします。
3. ダウンロードした zip ファイルからファイル Microsoft.Sdc.Tasks.dll を、ビルドの種類のチェックアウト先のフォルダに展開します。
4. この DLL をソース管理に追加してチェックインします。
5. TFSBuild.proj ファイルを修正することによって、ビルドが正しいディレクトリにファイルをコピーしてから、そのディレクトリを仮想ディレクトリとして作成するようにします。
   a. 次のような、コンパイル済み Web アプリケーションの場所を指定する **<PropertyGroup>** 要素を追加します。

```
<PropertyGroup>

<WebBinariesLocation>$(SolutionRoot)¥..¥Binaries¥.NET¥Release¥_PublishedWebSites¥MyWebSite</W
ebBinariesLocation>
</PropertyGroup>
```

b. CreateVirtualDirectory および DeleteVirtualDirectory タスクの参照を追加する 2 つの UsingTask 要素を追加します。

```
<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" />
```

c. AfterCompile ターゲットを追加して、仮想ディレクトリを作成し、ファイルをこのディレクトリにコピーします。

```
<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. ファイルを上書き保存して、ソース管理リポジトリに対してコミットします。

チーム ビルドを実行すると、次に Web アプリケーションが作成されて、仮想ディレクトリが作成され、その Web アプリケーションがこの仮想ディレクトリにコピーされます。

Web 配置プロジェクトを使用するには

1. Visual Studio 2005 Web Deployment Projects をダウンロードし、クライアント コンピュータにインストールします。
2. Visual Studio 2005 Web Deployment Projects をダウンロードし、ビルド サーバーにインスト

ールします。

3. Web アプリケーションを格納しているソリューションを開きます。
4. [**ビルド**] メニューをクリックしてから、[**Web 配置プロジェクトの追加...**] を選択します。
5. 配置プロジェクトの名前と場所を指定します。
6. ソリューション エクスプローラで、Web Deployment Project を右クリックしてから、[**プロパティ ページ**] を選択します。
7. ダイアログ ボックスで、チーム ビルドでビルドする必要のある [**構成**] (Debug または Release) を選択します。
8. [**配置**] セクションで、[**出力フォルダの IIS 仮想ディレクトリを作成する**] チェック ボックスを選択してから、仮想ディレクトリ名を指定します。
9. [**OK**] をクリックします。
10. ソリューションの変更をソース管理にチェックインします。

このソリューションを収めたビルドを実行すると、Web アプリケーションが作成され、Web アプリケーションのビルド先のディレクトリ内で仮想ディレクトリが作成されます。これは、ビルド ディレクトリ}¥{チーム プロジェクト名}¥{ビルドの種類}¥バイナリ¥{構成名}¥{プラットフォーム}¥_PublishedWebSite¥{Web Deployment Project 名となります。

参考資料

- SDC Tasks は、http://www.codeplex.com/sdctasks からダウンロードします。
- チーム ビルドを使用した ASP.NET アプリケーションのビルドの詳細については、http://msdn2.microsoft.com/ja-jp/teamsystem/aa718894.aspx の「TN_1600: Building ASP.NET projects with Team Build」を参照してください。
- Web Deployment Projects の使用の詳細については、http://msdn2.microsoft.com/ja-jp/teamsystem/aa718895.aspx の「TN_1601: Team Build and Web Deployment Projects」を参照してください。
- Web Deployment Projects の詳細については、http://msdn2.microsoft.com/ja-jp/asp.net/aa336619.aspx の「Visual Studio 2005 Web Deployment Projects」を参照してください。

チーム ビルドを使用した *.NET 1.1* アプリケーションのビルド方法

.NET 1.1 は、MSBuild で直接サポートされていないので、チーム ビルドでもサポートされていません。.NET 1.1 アプリケーションのビルドをサポートする MSBuild Extras (MSBee) という名称の、CodePlex 上のプロジェクトが Microsoft からリリースされました。

.NET 1.1 アプリケーションをビルドするには、プロジェクト ファイルを .NET 2.0 プロジェクト ファイルにアップグレードする必要があります。さらに、チーム ビルド プロジェクト ファイルを編集して、.NET 2.0 ツールではなく .NET 1.1 ツールを使用してビルドできるようにする必要もあります。

チーム ビルドを使用して .NET 1.1 アプリケーションをビルドするには

1. .NET 1.1 ソリューションを .NET 2.0 にアップグレードします。それには、Visual Studio 2005 でソリューションを開いて、変換ウィザードを実行するか、または devenv [projectname] /upgrade を実行します。
2. .NET 1.1 ソフトウェア開発キット (SDK) がビルド サーバーにインストール済みであることを確認します。
3. MSBuild Extras を http://www.codeplex.com/MSBee からダウンロードしてインストールします。
4. BuildingFx11inTB.targets を http://blogs.msdn.com/gautamg/attachment/578915.ashx からダウンロードします。
5. .NET 1.1 プロジェクトをビルドするソース管理からビルドの種類をチェックアウトします。
6. ビルドの種類を格納しているディレクトリに BuildingFx11inTB.targets をコピーし、このファイルをソース管理にチェックインします。
7. 次のようにして、TFSBuild.proj ファイルを編集します。
   a. 次の BuildingFx11inTB.targets ファイルをインポートします。

```
<Import Project="$(MSBuildProjectDirectory)¥BuildingFx11inTB.targets" />
```

   b. 次のような、CSharp ターゲットを定義するプロパティを追加します。

```
<PropertyGroup>

<AdditionalPropertiesForBuildTarget>CustomAfterMicrosoftCommonTargets=$(ProgramFiles)¥MSBuild¥MSBee¥MSBuildExtras.Fx1_1.CSharp.targets</AdditionalPropertiesForBuildTarget>
</PropertyGroup>
```

8. TFSBuild.proj をソース管理にチェックインします。

参考資料

- MSBuild Extras の詳細については、http://www.codeplex.com/MSBee の「MSBuild Extras - Toolkit for .NET 1.1 "MSBee"」を参照してください。
- MSBuild Extras を使用して .NET 1.1 アプリケーションをビルドする方法の詳細については、http://blogs.msdn.com/gautamg/archive/2006/04/19/578915.aspx の「Building .NET 1.1application using Team Build」を参照してください。

チーム ビルドを使用したセットアップ/配置プロジェクトのビルド方法

チーム ビルドは、既定ではセットアップ プロジェクトをサポートしません。次のようにして、カスタムのビルド後ステップを使用し、セットアップ プロジェクトをコンパイルして、ビルド ドロップ場所にバイナリをコピーします。

1. ビルドをテストします。

セットアップ プロジェクトで使用する予定のチーム ビルドが作動することを確認します。作動しない場合、先に進む前に修正を加えます。

ヒント: たいていのビルドは、メイン プロジェクト ビルドとセットアップ ビルドで構成されています。セットアップ プロジェクト専用の新しいチーム ビルドを作成する場合、ステップ 2 に進む前に

これを行います。

2. セットアップ プロジェクトが既定でビルドされることを確認します。

1. ソリューション エクスプローラで、チーム ビルドを作成する対象のインストーラ プロジェクトを右クリックします。
2. [**プロパティ**] をクリックします。
3. [**構成マネージャ...**] をクリックします。
4. たとえば、Debug や Release といった、ビルドしようとしている 1 つ以上のビルド構成を選択します。
5. インストーラ プロジェクトの [**ビルド**] チェック ボックスを選択します。

3. プロジェクト ファイル内のすべてのビルド パスが相対パスであることを確認します。

1. インストーラ プロジェクトのソリューション フォルダを開きます。
2. Visual Studio 以外のエディタ内で、.vdproj ファイルを開きます。
3. 編集用として .vdproj ファイルをチェックアウトします。
4. **SccLocalPath**、**SccAuxPath**、**OutputFileName**、および **SourcePath** の各項目を検索します。
5. 各項目のパスが、絶対パスではなく相対パスであることを確認します(これは、セットアップ プロジェクトの作成時の既定値になります）。絶対パスは、ドライブ名から始まり、相対パスは、二重の円記号 (‘¥¥’) で始まるか、または前に何も付いていません。
6. 絶対パスが見つかった場合、相対パスに置き換えます。どの定数式も変更しないでください。式は、後でインストーラによって拡張されます。以下に例を示します。

   "OutputFileName" = "8:c:¥¥temp¥¥SetupProject.msi" は、"OutputFileName" = "8:debug¥¥SetupProject.msi" に置き換えられます。

   ヒント: 相対パスは、プロジェクト フォルダから直接始まります。

   ヒント: パスを作成するときは、常に二重の円記号 ('¥¥') を使用します。なぜなら、これはコードに引き渡されて、単一の円記号 ('¥') にデコードされるからです。

7. 変更を加える必要がある場合、.vdproj ファイルをチェックインして元に戻します。

4. ポストコンパイル タスクをチーム ビルドに追加します。

1. セットアップ プロジェクトで使用する予定のチーム ビルドを開きます。
2. 次のようにして、ビルドの種類をソース管理からチェックアウトします。
   a. ビルドの種類は、ソース管理内のチーム プロジェクトの下にある、TFSBuild.proj という名前の TeamBuildTypes サブフォルダ内に置かれています。
   b. 必要により、先にフォルダに対して [**最新バージョンの取得**] 操作を実行してください。
3. [ソース管理エクスプローラ] で、TFSBuild.proj をダブルクリックして開きます。
   注意: チーム エクスプローラからビルドの種類を表示しようとしても、うまくいきません。そこでは、ビルド ファイルの読み取り専用コピーしか開かれないからです。
4. このファイルの末尾の、最後の </ItemGroup> タグおよび最後の </Project> タグの間に、次のようなコードを追加します。

```
<Target Name="AfterCompile">
<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;"/>
<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. 貼り付けたコード内のパスをチェックし、それがビルド サーバーに対して正確であることを確認します。
   ヒント: コマンド ラインを使用して、exec コマンド タグ内のパスをテストします。コマンド ラインでのテストでは、すべての " を引用符に置き換えます。以下に例を示します。

```
"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"

ヒント: SourceFiles パスをテストするときにも、コマンド ラインを使用します。

5. ビルドの変更をテストします。

1. チーム エクスプローラで、ビルドの種類を右クリックしてから、[**チーム プロジェクトのビルド**] をクリックします。
2. ビルドの概要を見直して、ビルドが成功または失敗のどちらであったかを判別します。
3. ビルドに失敗した場合、ビルド ログへのリンクをクリックします。失敗には一般的に、次のような原因があります。
   a. **exec コマンドのパスが誤っています。**
   b. **出力ファイルをビルド ディレクトリにコピーする許可が設定されていません。**tfsservice ユーザー アカウントが、バイナリ フォルダからのコピーと、ビルド フォルダへのコピーの許可を受けていることを確認してください。バイナリ フォルダは、ソリューションのビルド後に msi ファイルが置かれる場所です。ビルド フォルダは、<BuildDirectoryPath> タグ内の tfs.proj に指定します。
   c. **インストーラがビルドを行うための許可が設定されていません。**tfsserver ユーザー アカウントが、インストーラ プロジェクト用と、インストーラに関連付けられたアプリケーション用の .vdproj ファイルとソース ファイルを読み取る許可を受けていることを確認します。また、tfsserver ユーザー アカウントが、たとえば Debug や Release などの、バイナリ ファイルを出力ディレクトリに書き込む許可を受けていることを確認します。
   d. **ビルド構成が誤っています。**プロジェクト用として、**exec** コマンドで指定したビルド構成が存在することを確認します。たとえば、"Debug|Any CPU" を持っていても "Debug" は持っていないプロジェクトもありえます。これをチェックするには、ソリューション エクスプローラでソリューションのプロパティを調べます。
4. ビルド ログに十分な情報が示されていない場合、exec コマンドの出力ファイルを作成してからログを見直して、詳細情報を確認します。以下に例を示します。

   <Exec Command=""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" > c:¥temp¥output.txt"/>

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms404859(VS.80).aspx の「チュートリアル: Visual Studio セットアップ プロジェクトをビルドするために Team Foundation ビルドを構成する」を参照してください。

チーム ビルドの作成方法

チーム エクスプローラ内のチーム ビルド フォルダから、新しいチーム ビルドを作成することができます。

チーム ビルドを作成するには

1. [チーム エクスプローラ] を開きます。
2. ビルドを作成する対象のチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを右クリックします。
4. [**新しいチーム ビルドの種類...**] を選択します。
5. 新しいチーム ビルドの種類の名前を指定し、[**次へ**] をクリックします。
6. ビルドしようとしているプロジェクトを選択します。これには、単体テストを備えたプロジェクトが組み込まれていなければなりません。
7. ビルド構成 (たとえば、Release または Debug) を選択し、[**次へ**] をクリックします。
8. たとえば **TFSRTM** などの、ビルド サーバーの名前を指定します。
9. たとえば **c:¥builds** などの、ローカル ビルド ディレクトリをビルド サーバー上に指定します。
10. たとえば ¥¥**TFSRTM¥NightlyBuilds** などの、ビルド出力のドロップ場所を指定してから、[**次へ**] をクリックします。
11. [**テストの実行**] チェック ボックスをクリックし、ビルドに関連付けたいテスト リストを選択してから、[**次へ**] をクリックします。
12. [**完了**] をクリックし、チーム ビルドの種類を作成します。

複数のビルドの種類の作成方法

たとえば Release for Customers や Debug for Test Team などの、複数のビルドの種類を作成するには、該当するビルドの種類ごとに別々のチーム ビルドを使用します。

チーム ビルドを作成するには

1. [チーム エクスプローラ] を開きます。
2. ビルドを作成する対象のチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを右クリックします。
4. [**新しいチーム ビルドの種類...**] を選択します。
5. 新しいチーム ビルドの種類の名前を指定し、[**次へ**] をクリックします。
6. ビルドしようとしているプロジェクトを選択します。これには、単体テストを備えたプロジェクトが組み込まれていなければなりません。
7. ビルド構成 (たとえば、Release または Debug) を選択し、[**次へ**] をクリックします。
8. たとえば **TFSRTM** などの、ビルド サーバーの名前を指定します。
9. たとえば **c:¥builds** などの、ローカル ビルド ディレクトリをビルド サーバー上に指定します。
10. たとえば ¥¥**TFSRTM¥NightlyBuilds** などの、ビルド出力のドロップ場所を指定してから、[**次へ**] をクリックします。
11. [**テストの実行**] チェック ボックスをクリックし、ビルドに関連付けたいテスト リストを選択してから、[**次へ**] をクリックします。
12. [**完了**] をクリックし、チーム ビルドの種類を作成します。

別のプロジェクトのアセンブリを参照するプロジェクト用のチーム ビルドの作成方法

別のチーム プロジェクトに対する依存関係を持ったプロジェクトをビルドするには、そのプロジェクトからビルド サーバー上のワークスペースにソースまたはアセンブリを取り込む必要があります。そのためには、TFSBuild.proj ファイルを編集してアセンブリまたはソリューションの参照を追加し、**BeforeGet** イベントをオーバーライドして、必要な各チーム プロジェクトからアセンブリまたはソリューションを取り込む必要があります。

別のチーム プロジェクトのアセンブリを参照するプロジェクトを作成するには

1. TFSBuild.proj スクリプトをソース管理エクスプローラからチェックアウトします。
2. 以下の構成設定を PropertyGroup セクションの下に追加します。

```
<!-- PropertyGroup の下に追加します -->
<TfCommand>$(TeamBuildRefPath)¥..¥tf.exe</TfCommand>
<SkipInitializeWorkspace>true</SkipInitializeWorkspace>
```

SkipInitializeWorkSpace を使用すれば、ビルド マシン上でワークスペースを削除および再作成する既定のタスクを起動しなくて済みます。新しいプロパティが、カスタム ターゲット **BeforeGet** で使用されています (以下を参照)。

3. チーム プロジェクトとソリューションの両方をマップする ItemGroup 項目に、以下の構成設定を追加します。ビルド マシンの正しいローカル パスを指定したことを確認します。複数のマッピングによって、同じローカル フォルダを共有することはできません。そのようにした場合、CreateWorkspace タスクで MappingConflictException 例外になってしまいます。

```
<ItemGroup>
<!-- 参照する各ソリューションごとに項目を 1 つずつ追加します -->
<SolutionToBuild Include="$(SolutionRoot)¥DependentApp¥DependentApp.sln" />
<SolutionToBuild Include="$(SolutionRoot)¥YourApp¥YourApp.sln" />
</ItemGroup>

<ItemGroup>
<!-- 参照する各チーム プロジェクトごとに項目を 1 つずつ追加します -->
<Map Include="$/YourApp/YourApp">
<LocalPath>$(SolutionRoot)¥YourApp</LocalPath>
</Map>
<Map Include="$/DependentApp/DependentApp">
<LocalPath>$(SolutionRoot)¥DependentApp</LocalPath>
</Map>
</ItemGroup>
```

4. 次のように、BeforeGet イベントをオーバーライドして、各チーム プロジェクトのワークスペースを取り出します。

```
<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:$(WorkSpaceName)
&quot;$(SolutionRoot)&quot;"/>
<Exec
WorkingDirectory="$(SolutionRoot)"
Command="&quot;$(TfCommand)&quot; workfold /map /workspace:$(WorkSpaceName)
/server:$(TeamFoundationServerUrl) &quot;%(Map.Identity)&quot; &quot;%(Map.LocalPath)&quot;"/>
</Target>
```

5. ビルド スクリプトをチェックインし、ビルドを実行します。

参考資料

- 詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の「Working with multiple team projects in Team Build」を参照してください。

電子メールのビルド イベントのサブスクライブ方法

電子メールのビルド イベントをサブスクライブするための新しいプロジェクト通知を作成することができます。

電子メールのビルド通知をセットアップするには

1. チーム エクスプローラで、該当するチーム プロジェクトを右クリックします。

2. [**プロジェクト警告**] を選択します。
3. 受け取りを希望する各通知オプションを選択し、通知用の電子メール アドレスを入力します。

参考資料

- プロジェクト警告の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181334(VS.80).aspx の「警告の設定」を参照してください。

ビルドの失敗時に通知を受け取る方法

ビルドの失敗時にのみ電子メールを受信するための、フィルタを使用した新しいプロジェクト通知を作成することができます。

ビルドの失敗時に通知を受けるためのプロジェクト警告を作成するには

1. **ビルドを作成してテストします。**使用できるチーム ビルドの種類があって、エラーなしで稼働していることを確認します。
2. **BuildCompletionEvent イベントの登録を行います。BisSubscribe.exe** ツールを使用して、ビルドの失敗時にのみ電子メール通知の受け取りを指定するフィルタを使って、**BuildCompletionEvent** イベントへの登録を行います。それには、以下のようなコマンド ライン構文を使用します。

```
bissubscribe /eventType BuildCompletionEvent /address myemail@domain.com /deliveryType EmailPlaintext
/server tfsserver1 /filter "TeamProject = 'MyTeamProject' AND CompletionStatus=‘Failed’”
```

3. **ビルドをテストします。**ビルドをテストするには、コンパイルに失敗したコードをチェックインし、ビルドを実行し、そして電子メール通知の受信を確認します。

参考資料

- ビルドのフィルタリング完了イベントの詳細については、http://blogs.msdn.com/jpricket/archive/2006/09/05/how-to-filter-the-build-completion-e

vent.aspx の「How to Filter the Build Completion Event」を参照してください。

- BuildCompletionEvent フィルタの詳細については、http://blogs.msdn.com/jpricket/archive/2006/09/05/useful-buildcompletionevent-filters.aspx の「Useful BuildCompletionEvent Filters」を参照してください。

ビルドの開始方法

チーム エクスプローラ内のチーム ビルド フォルダから、ビルドの種類を開始することができます。

ビルドを手動で開始するには

1. [チーム エクスプローラ] を開きます。
2. ビルドを開始する対象のチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. 開始したいチーム ビルドの種類を右クリックします。
5. [**チーム プロジェクトのビルド**] を選択します。

ビルドの正常終了の確認方法

チーム エクスプローラからアクセス可能な [ビルド] ウィンドウで、ビルドの状況をチェックすることができます。

ビルドが正常完了したことを検証するには

1. [チーム エクスプローラ] を開きます。
2. 結果を表示したいチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. 結果を表示したいチーム ビルドの種類をダブルクリックします。

ビルド出力の表示方法

チーム エクスプローラからアクセス可能な [ビルド] ウィンドウで、ビルド出力を表示することができます。

ビルド出力を表示するには

1. [チーム エクスプローラ] を開きます。
2. ビルド出力を表示したいチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. ビルド出力を表示したいチーム ビルドの種類をダブルクリックします。
5. 出力を表示したいビルド番号のチーム ビルドの結果項目をダブルクリックします。
6. ビルド出力フォルダを表示したい場合、[**ビルド名**] リンクをクリックします。
7. ビルド ログを表示したい場合、[**ログ**] リンクをクリックします。

ビルド サーバーの場所の変更方法

既存のチーム ビルド用のビルド サーバーの場所を変更するには、TFSBuild.proj 内の **<BuildMachine>** タグを変更します。

既存のチーム ビルドの種類のビルド サーバーの場所を変更するには

1. [ソース管理エクスプローラ] を開きます。
2. [ソース管理エクスプローラ] で、チーム プロジェクト フォルダを展開します。
3. [TeamBuildTypes] フォルダを展開します。
4. コード分析をオンにするチーム ビルド フォルダを選択します。
5. TFSBuild.proj ファイルをソース管理からチェックアウトします。必要により、先にフォルダに対して [**最新バージョンの取得**] 操作を実行してください。
6. [ソース管理エクスプローラ] で、TFSBuild.proj をダブルクリックして開きます。
7. 新しいサーバーを指すように **<BuildMachine>** タグを変更します。
8. TFSBuild.proj を上書き保存し、元のソース管理にチェックインします。

参考資料

- Team Foundation ビルドのカスタマイズの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400688(VS.80).aspx の「Team Foundation ビルドのカスタマイズ」を参照してください。

ビルド出力場所の変更方法

既存のチーム ビルド用のビルド出力場所を変更するには、TFSBuild.proj 内の **<DropLocation>** タグを変更します。

既存のチーム ビルドの種類のビルド サーバーの場所を変更するには

1. [ソース管理エクスプローラ] を開きます。
2. [ソース管理エクスプローラ] で、チーム プロジェクト フォルダを展開します。
3. [TeamBuildTypes] フォルダを展開します。
4. コード分析をオンにするチーム ビルド フォルダを選択します。
5. TFSBuild.proj ファイルをソース管理からチェックアウトします。先にフォルダに対して [**最新のバージョンを取得**] 操作を実行する必要があるかもしれません。
6. [ソース管理エクスプローラ] で、TFSBuild.proj をダブルクリックして開きます。
7. 新しい場所を指すように **<DropLocation>** タグを変更します。
8. TFSBuild.proj を上書き保存し、元のソース管理にチェックインします。

参考資料

- Team Foundation ビルドのカスタマイズの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400688(VS.80).aspx の「Team Foundation ビルドのカスタマイズ」を参照してください。

どの変更セットがビルドの一部であるかの判別方法

チーム エクスプローラからアクセス可能な [ビルド] ウィンドウで、各ビルドに関連付けられている変更セットを表示することができます。

ビルドに関連付けられている変更セットを表示するには

1. [チーム エクスプローラ] を開きます。
2. 変更セットを表示したいチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. 変更セットを表示したいチーム ビルドの種類をダブルクリックします。
5. 変更セットを表示したいビルド番号のチーム ビルドの結果項目をダブルクリックします。

6. [**関連付けられた変更セット**] 項目を展開します。
7. 特定の変更セットのさらに詳しい情報を見たい場合、[**ID**] リンクをクリックします。

レポートされたビルドの品質の変更方法

チーム エクスプローラからアクセス可能な [ビルド] ウィンドウで、ビルド品質を変更することができます。

ビルド品質を変更するには

1. [チーム エクスプローラ] を開きます。
2. ビルド品質を変更する対象のチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. ビルド品質を変更したいチーム ビルドの種類をダブルクリックします。
5. [**ビルドの品質評価**] ドロップダウン リストで、ビルド品質を設定したいビルドを選択してから、ビルド品質の値を設定します。

プロジェクト

- **シングル ソリューション戦略の使用方法**
- **パーティション ソリューション戦略の使用方法**
- **複数ソリューション戦略の使用方法**

シングルソリューション戦略の使用方法

小型のシステムで作業する場合、単一の Visual Studio ソリューションを使用して、すべてのプロジェクトを収容することを検討してみてください。この戦略をとれば、ソリューションを開いたときにすべてのコードが揃っているので、開発が容易になります。この戦略ではまた、ソリューション内のプロジェクトどうしの間ですべての参照が行われるので、参照を簡単にセットアップすることもできます。ただし、ソリューション外にあるサード パーティのアセンブリ (別売りコンポーネントなど) を参照するために、やはりファイル参照を使用する必要はあります。

参考資料

- 詳細については、このガイドの「第 3 章 – ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

パーティション ソリューション戦略の使用方法

大規模なシステムで作業する場合、それぞれがアプリケーションのサブシステムを表す複数の Visual Studio ソリューションの使用を検討してみてください。このようなソリューションを開発者が使用すれば、システムを細分化して作業することができ、すべてのプロジェクトに対してすべてのコードをロードしなくて済みます。依存しあう関係にあるすべてのプロジェクトが 1 つのグループにまとまるように、ソリューション構造を設計します。そうすれば、ファイル参照ではなく、プロジェクト参照を使用できるようになります。このやり方でアプリケーション全体をビルドしたい場合は、すべてのプロジェクトを収容するマスタ ソリューション ファイルの作成を検討してみてください。
複数のソリューションを使用して作業する場合、すべてのプロジェクトでフラットなファイル構造を使用します。典型的な例としては、Microsoft Windows フォーム プロジェクト、ASP.NET プロジェクト、Windows サービス、およびこれらのプロジェクトの一部または全部で共用される一連のクラス ライブラリ プロジェクトを擁するアプリケーションがあります。
どのプロジェクトでも、以下のフラット構造を使用することができます。

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

構造をフラットに保つことによって、高い柔軟性と、プロジェクトのさまざまなビューを示すためのソ

リューションを使用する機能を獲得することができます。ソリューション ファイルを軸とした物理的なフォルダ構造の設計は変更が困難です。別のソリューションからクラス ライブラリを再利用する必要のある場合は特にそうです。

注意: チーム ビルドを使用してビルドする (この場合、MSBuild に依存します) 場合、参照するプロジェクトをすべては組み込んでいないソリューション構造を作成することは可能です。ソリューション全体がまずビルドされてしまえば (各ソリューションからバイナリ出力を生成して)、MSBuild は、ソリューションの境界外でプロジェクト参照を追随して、正常にビルドを行うことができます。このやり方で作成されたソリューションを Visual Studio の build コマンドからビルドすることはできません。このアプローチは、チーム ビルドおよび MSBuild でのみ効果があります。

参考資料

- 詳細については、このガイドの「第 3 章 – ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

複数ソリューション戦略の使用方法

多数のプロジェクトを必要とする非常に大きいソリューションで作業する場合、ソリューションのスケーラビリティ制限を受けることがあります。そのようなシナリオでは、アプリケーションを複数のソリューションに分割する必要が生じます。ただし、アプリケーション全体用のマスタ ソリューションは作成しません。なぜなら、各ソリューション内のどの参照も、プロジェクト参照であるからです。各ソリューション外部のプロジェクト (たとえば、サード パーティのライブラリや、別のサブソリューション内のプロジェクト) に対する参照は、ファイル参照です。すなわち、「マスタ」ソリューションは存在しえないということです。そのため、ソリューションをビルドする順序を理解しているスクリプトを使用する必要があります。複数ソリューション構造に関連した保守タスクの一環として、ソリューションからソリューションへの循環参照が開発者によって不用意に作成されないようにします。この構造では、複合ビルド スクリプトと、依存関係の明示的なマッピングが必要になります。またこの構造では、Visual Studio だけでアプリケーション全体をビルドすることはできません。つまり、TFS チーム ビルドを使用します。

参考資料

- 詳細については、このガイドの「第 3 章 – ソース管理におけるプロジェクトとソリューションの構造化」を参照してください。

レポート

- **ビルド品質の表示方法**
- **ビルドのすべてのチェックインの表示方法**
- **ビルドの終了済み作業項目またはバグの表示方法**
- **ビルドの未完了の作業項目またはバグの表示方法**
- **ビルドからビルドへの進行速度のトラッキング方法**
- **テスト ケースでのビルドの合格/不合格結果のトラッキング方法**
- **ビルド状況の検討方法 (BVT 結果)**

ビルド品質の表示方法

チーム エクスプローラからアクセス可能な [ビルド] ウィンドウで、ビルド品質を表示することができます。

ビルド品質を表示するには

1. [チーム エクスプローラ] を開きます。
2. ビルド品質を表示したいチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. ビルド品質を表示するためのチーム ビルドの種類をダブルクリックします。

Microsoft Solutions Framework (MSF) for CMMI® Process Improvement (MSF CMMI) プロセス テンプレートを使用する場合、ビルド レポートを表示して、ビルド品質のほかに、テスト結果、コード カバレッジ、およびコード チャーンに関する追加情報も表示することができます。

MSF CMMI でビルド品質を表示するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを拡張します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. このレポート サイトで、[**ビルド**] レポートを選択します。

ビルドのすべてのチェックインの表示方法

チーム エクスプローラからアクセス可能な [ビルド] ウィンドウで、各ビルドに関連付けられているチェックインを表示することができます。

ビルドに関連付けられているチェックインを表示するには

1. [チーム エクスプローラ] を開きます。
2. ビルドのチェックインを表示する対象のチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. ビルドのチェックインを表示したいチーム ビルドの種類をダブルクリックします。
5. ビルドのチェックインを表示したいチーム ビルド結果項目をダブルクリックします。
6. [**関連付けられた変更セット**] 項目を展開し、ビルドに関連したすべてのチェックインを表示します。
7. 特定の変更セット (チェックインを表します) のさらに詳しい情報を見たい場合、[**ID**] リンクをクリックします。

ビルドの終了済み作業項目またはバグの表示方法

チーム エクスプローラからアクセス可能な [ビルド] ウィンドウで、各ビルドの終了した作業項目とバグを表示することができます。

ビルドに関連付けられている作業項目を表示するには

1. [チーム エクスプローラ] を開きます。
2. 作業項目を表示したいチーム プロジェクトを展開します。
3. ツリー内のチーム ビルド フォルダを展開します。
4. 作業項目を表示したいチーム ビルドの種類をダブルクリックします。
5. 作業項目を表示したいビルド番号のチーム ビルドの結果項目をダブルクリックします。

6. [**関連付けられた作業項目**] を展開します。

ビルドの未完了の作業項目またはバグの表示方法

MSF CMMI プロセス テンプレートを使用している場合、[懸案事項およびブロックされた作業項目] レポートを開いて、ある一定期間中の実施中の作業項目、解決された作業項目、終了された作業項目を表示することができます。ただし、このレポートは、ビルド別ではなく日付別に情報を提示するので、ビルドが作成された日付に従って、結果をビルドに置き換える必要があります。

ビルドの実施中の作業項目またはバグを表示するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを拡張します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. レポート サイトで、[**懸案事項およびブロックされた作業項目**] レポートを選択します。

ビルドからビルドへの進行速度のトラッキング方法

[プロジェクト速度] レポートを使用して、ビルドからビルドへと作業項目が完了されていく進行状況と速度を追跡することができます。このレポートは、MSF CMMI および MSF for Agile Software Development (MSF Agile) の両方で利用できます。

ビルドからビルドへの進行速度をトラッキングするには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを拡張します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. レポート サイトで、[**プロジェクト速度**] レポートを選択します。

テスト ケースでのビルドの合格/不合格結果のトラッキング方法

[品質指標] レポートを使用して、ある一定の期間中に合格または不合格になったテスト ケースの数を追跡することができます。このレポートは、ビルド別ではなく日付別に情報を提示します。したがって、ビルドが作成された日付に従って、結果をビルドに変換する必要があります。このレポートは、MSF

CMMI および MSF Agile の両方で利用できます。

テスト ケースでのビルドの合格/不合格をトラッキングするには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを拡張します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. このレポート サイトで、[**品質指標**] レポートを選択します。

ビルド状況の検討方法 *(BVT 結果)*

MSF CMMI プロセス テンプレートを使用する場合、ビルド レポートを表示して、BVT 結果を見ることができます。

ビルド状況を検討するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを拡張します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. このレポート サイトで、[**ビルド**] レポートを選択します。

スケジュールされたビルド

- **ナイトリー ビルドを自動的に実行する方法**
- **プロジェクトでのビルドの頻度と種類の決定方法**

ナイトリー ビルドを自動的に実行する方法

TFS のチーム ビルド機能では、ユーザー インターフェイスからのスケジュールされたビルドはサポートされません。その代りに、Microsoft Windows タスク スケジューラを使用して、TFSBuild コマンド ユーティリティを実行し、事前に取り決めた時刻にビルドを開始することができます。

スケジュールされたビルドを作成するには

1. 以下のようにして、TFSBuild コマンド ラインを作成します。

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

2. コマンド ラインをバッチ ファイルに入れます。
3. バッチ ファイルを任意のサイクル間隔で実行する Windows のスケジュールされたタスクを作成します。

さらに詳細な情報と拡張ステップについては、このガイドの「Visual Studio Team Foundation Server でスケジュール ビルドをセットアップする方法」を参照してください。

参考資料

- 詳細については、このガイドの「第 9 章 – チーム ビルドにおけるスケジュール ビルドのセットアップ」を参照してください。

プロジェクトでのビルドの頻度と種類の決定方法

ビルドの頻度は、スケジュールされたビルドの作成時に下す最も重要な決定事項の 1 つです。

1 時間以内で大幅な変更の原因となるのに十分な数のチェックインを持つプロジェクトで作業する場合に、継続的な統合 (CI) を使用しないのであれば、頻度として時間単位ビルドを選択することができます。時間単位ビルドの場合、フィードバックがすぐに開発者に戻されます。また、これをテスト担当者やその他のチーム メンバが使用できるようにして、それぞれのフィードバックを求めることもできます。

1 日以内で大幅な変更の原因となるのに十分な数のチェックインを持つプロジェクトで作業する場合、頻度としてデイリービルドを選択することができます。なぜならこれによって、テストおよび開発のチームに毎朝新しいビルドが提供され、そこには前日の変更内容が組み入れられていて、テスト準備が整っているからです。

ビルド期間が数日にわたる大型の複雑なプロジェクトで作業する場合、ウィークリービルドを選ぶ必要があります。それによって、テスト チームは各週の初めに、前の週の変更内容が組み入れられたビル

ドが提供され、テストの準備が整えられています。

参考資料

- スケジュール ビルドのセットアップの詳細については、このガイドの「Visual Studio Team Foundation Server でスケジュール ビルドをセットアップする方法」を参照してください。
- 詳細については、このガイドの「第 9 章 – チーム ビルドにおけるスケジュール ビルドのセットアップ」を参照してください。

テストを軸とした開発

- **「hello world」受け入れテストの作成方法**
- **ビルドの一環としての自動テストの実行方法**
- **ビルドの一環としてのコード分析の実行方法**
- **テストの失敗時にビルドを失敗させる方法**

「*hello world*」受け入れテストの作成方法

「hello world」受け入れテストは、最もシンプルなテストです。これを使用して、単体テストを作成し、ビルド プロセスにフックできることを確認できます。

ビルドに関連付けるテスト リストを作成するには、Visual Studio Test Edition または Visual Studio Team Suite をインストールしている必要があります。ビルド サーバー上で自動テストを実行するには、Visual Studio Test Edition、Visual Studio Developer Edition、または Visual Studio Team Suite をビルド サーバーにインストールしている必要があります。

hello world テストを作成するには

1. [**テスト**] メニューで、[**新しいテスト...**] をクリックします。
2. [**単体テスト**] をクリックしてから、[**OK**] をクリックします。
3. テスト プロジェクトの名前を入力し、[**作成**] をクリックします。
4. 新しいプロジェクトをコンパイルします。

5. プロジェクトをソース管理にチェックインします。
6. Visual Studio で単体テスト ソリューションを開いたまま、[**テスト**] メニューで [**新しいテスト リストの作成...**] をクリックします。
7. テスト リストの名前を [**新しいテスト リストの作成**] ダイアログに指定してから、[**OK**] をクリックします。
8. [テスト マネージャ] で、[**読み込まれたすべてのテスト**] ノードをクリックします。
9. 利用できるテストの中からテストをドラッグし、ツリー内のテスト リスト ノードにドロップします。
10. 変更済みの単体テスト プロジェクト VSMDI ファイルをソース管理にチェックインします。

作成したテスト リストは、新しいチーム ビルドの作成時に利用することができ、ビルド プロセスの一環として自動的に実行されます。

参考資料

- ビルド検証テストの自動実行の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms182465(VS.80).aspx の「方法 : ビルド検証テスト (BVT) を構成および実行する」を参照してください。
- Visual Studio Test Edition または VSMDI ファイルなしで自動ビルド テストを実行する方法の詳細については、http://blogs.msdn.com/buckh/archive/2006/11/04/how-to-run-tests-without-test-metadata-files-and-test-lists-vsmdi-files.aspx の「How to run tests in a build without test metadata files and test lists (.vsmdi files)」を参照してください。

ビルドの一環としての自動テストの実行方法

自動テストを実行して、各ビルド後にビルドの品質に関するフィードバックを自動的に取得することができます。自動テストを実行するには、Visual Studio Developer Edition のほかに Visual Studio Test Edition をビルド サーバーにインストールしているか、または Team Suite 全体をインストールしている必要があります。Developer Edition は、ビルドの実行に必要であるのに対して、Test Edition は、

実行可能なテストおよびテスト一覧をセットアップするのに必要です。

ビルドの一環として自動テストを実行するには

1. ビルドに対して実行したい 1 つ以上の自動テストを作成します。
2. [**テスト**] メニューで [**新しいテスト リストの作成**] をクリックします。
3. [テスト マネージャ] を使用して、テストをグループとして新しいテスト一覧にまとめます。それには、テスト マネージャ内のテスト ビューからテストをドラッグしてテスト一覧にドロップします。
4. 新しいチーム ビルドの種類を作成します。
5. 自動テストを実行するチェック ボックスを選択します。
6. テストおよびテスト一覧を作成したテスト プロジェクトを選択します。
7. 実行したいテスト一覧を選択します。

参考資料

- ビルド検証テストの自動実行の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms182465(VS.80).aspx の「方法 : ビルド検証テスト (BVT) を構成および実行する」を参照してください。
- Visual Studio Test Edition または VSMDI ファイルなしで自動ビルド テストを実行する方法の詳細については、http://blogs.msdn.com/buckh/archive/2006/11/04/how-to-run-tests-without-test-metadata-files-and-test-lists-vsmdi-files.aspx の「How to run tests in a build without test metadata files and test lists (.vsmdi files)」を参照してください。

ビルドの一環としてのコード分析の実行方法

ビルドの種類のコード分析をオンにするには、新しいチーム ビルドの種類の作成時に、新しいチーム ビルドの種類の作成ウィザードの [コード分析] チェック ボックスを選択するか、またはビルドの種類の作成後に TFSBuild.proj ファイルを修正します。

TFSBuild.proj ファイルでコード分析を有効にするには

- プロジェクトの設定に関係なくすべてのプロジェクトでコード分析を実行したい場合、**<RunCodeAnalysis>** タグを **Always** に変更します。
- プロジェクト設定に基づいて、プロジェクトごとにコード分析を実行したい場合、**<RunCodeAnalysis>** タグを **Default** に変更します。

参考資料

- ビルドの一環としての自動コード分析についての詳細については、このガイドの「Visual Studio Team Foundation Server でチーム ビルドを使用してコード分析を自動実行する方法」を参照してください。
- コード分析ツールの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms182023(VS.80).aspx の「コード分析ツールを使用するためのガイドライン」を参照してください。

テストの失敗時にビルドを失敗させる方法

コンパイル エラーが原因でビルドが失敗した場合、障害を追跡する作業項目が作成されて、ビルドには失敗済みのマークが付けられます。ただし、自動テストが失敗しても、ビルドは失敗しません。テストの失敗は警告に変換されて、ビルドは続行されます。

関連した自動テストが失敗した場合には、ビルドも失敗させることができます。また、障害を追跡する作業項目が自動的に生成されるようにすることもできます。

テストの失敗と同時にビルドも失敗させるには

1. Program Files¥MSBuild¥Microsoft¥VisualStudio¥v8.0¥TeamBuild から Microsoft.TeamFoundation.Build.targets ファイルを開きます。
2. テストの失敗と同時に失敗させたいチーム ビルドの種類用として、TFSBuild.proj ファイルを編集目的でチェックアウトして開きます。

3. TFSBuild.proj ファイルの末尾の閉じる </Project> タグの直前に、**RunTestWithConfiguration** ターゲットを Microsoft.TeamFoundation.Build.targets からコピーします。
4. **ContinueOnError** 属性を **true** から **false** に変更します。
   注意: テスト ツール タスクは 2 通りあります。ビルド サーバー上でビルドの動作変更のみの場合は、エンドツーエンド タスクを変更します。開発者のデスクトップ上でのビルド時には、デスクトップ ビルドを使用します。

別の方法として、テストの失敗と同時にすべてのチーム ビルドの種類を失敗させたい場合、Microsoft.TeamFoundation.Build.targets を直接変更してもかまいません。そうすると、すべてのチーム ビルドの種類の動作が変更されます。

上記の推奨ソリューションは、簡単に実装できますが、今後のバージョンの Visual Studio でも引き続き動作するとは限りません。アップグレード後も確実に引き続き動作するリューションを実装したい場合、http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether-tests-passed-in-team-build.aspx の Aaron Hallberg 氏のブログ エントリ「Determining Whether Tests Passed in Team Build」を参照してください。

参考資料

- テストの失敗と同時に作業項目を作成するためのビルドのセットアップの詳細については、http://blogs.msdn.com/nagarajp/archive/2005/10/14/481290.aspx の「Create Workitems for Test Failures in TeamBuild」を参照してください。
- Visual Studio のアップグレード後も確実に引き続き動作するソリューションの詳細については、http://blogs.msdn.com/aaronhallberg/archive/2006/09/21/determining-whether-tests-passed-in-team-build.aspx の「Determining Whether Tests Passed in Team Build」を参照してください。

チーム ビルドの資料

- Team Foundation ビルドの概要は、http://msdn2.microsoft.com/ja-jp/library/ms181710(vs.80).aspx の「Team Foundation ビルドの概要」を参照してください。

プラクティス: プロジェクト管理

索引

チェックイン ポリシー

- コード品質の実施のためのチェックイン ポリシーのセットアップ方法
- 開発者によって作業項目がチェックインに関連付けられるようにするためのチェックイン ポリシーのセットアップ方法
- コード基準の実施のためのチェックイン ポリシーのセットアップ方法

プロジェクト管理

- *Microsoft Project* を使用したプロジェクトの管理方法
- *Microsoft Excel* を使用したプロジェクトの管理方法
- 最小限のプロセス テンプレートの作成方法
- プロセス テンプレートのカスタマイズ方法
- プロセス テンプレートでの作業項目の種類のカスタマイズ方法
- 既存のチーム プロジェクトでの作業項目の種類のカスタマイズ方法
- イテレーションの作成方法
- 領域の作成方法
- チェックイン イベント通知の追加方法
- レポート ダッシュボードのセットアップ方法
- ソース管理リポジトリでのフォルダの作成方法
- *Team Foundation Server* からのプロジェクトの削除方法

チェックイン ポリシー

- **コード品質の実施のためのチェックイン ポリシーのセットアップ方法**
- **開発者によって作業項目がチェックインに関連付けられるようにするためのチェックイン ポリシーのセットアップ方法**
- **コード基準の実施のためのチェックイン ポリシーのセットアップ方法**

コード品質の実施のためのチェックイン ポリシーのセットアップ方法

コード品質基準を強制するには、コード分析とテスト ポリシーを組み合わせて使用します。たとえば、Microsoft® Visual Studio® 2005 Team Foundation Server (TFS) ソース管理にソースをチェックインするために、事前にある特定のテストが必ず実行されて合格するようにするには、VSTS ですぐに使用できるテスト ポリシーを使用します。また、コード分析ポリシーを構成して、ある特定の品質基準をコードが必ず満たすようにすることもできます。それには、セキュリティ、パフォーマンス、移植性、保守容易性、および信頼性の規則が必ず守られるようにします。

コーディングの基準およびガイドラインについて定めたポリシーに加えて、この種のチェックイン ポリシーを実施することによって、コードが必ず規定のコード品質レベルを満たすようにします。

チーム プロジェクトでコード分析チェックイン ポリシーを実施するには

1. [チーム エクスプローラ] でチーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] をポイントしてから、[**ソース管理**] をクリックします。
2. [**チェックイン ポリシー**] タブをクリックします。
3. [**追加**] をクリックしてから、該当するポリシーを選択して構成します。

参考資料

- カスタム チェックイン ポリシーの作成と使用方法の詳細については、このガイドの「方法 ：TFS

のカスタム チェックイン ポリシーを作成するステップ」を参照してください。

- チェックイン ポリシーをカスタマイズする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインでの選択済みパターンを禁止するサンプル コードの詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインでコメントを実施するサンプル コードは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーを登録する方法の詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。

開発者によって作業項目が必ずチェックインに関連付けられるようにするためのチェックイン ポリシーのセットアップ方法

すぐに使用できる [**作業項目**] チェックイン ポリシーを構成して、開発者によって作業項目がチェックインに必ず関連付けられるようにします。この関連付けによって、ソース コードに加えられた変更と、バグおよびタスクをトラッキングする作業項目との間で追跡を続けることができます。

開発者によって必ずチェックインが作業項目に関連付けられるように作業項目チェックイン ポリシーを構成するには

1. [チーム エクスプローラ] でチーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] を選択してから、[**ソース管理**] をクリックします。
2. [**チェックイン ポリシー**] タブをクリックします。
3. [**追加**] をクリックしてから、[**作業項目**] チェックイン ポリシーを選択して構成します。

参考資料

- カスタム チェックイン ポリシーの作成と使用方法の詳細については、このガイドの「方法 ：TFS のカスタム チェックイン ポリシーを作成するステップ」を参照してください。
- チェックイン ポリシーをカスタマイズする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。

コーディング基準の実施のためのチェックイン ポリシーのセットアップ方法

Team Foundation Server に付属しているコード分析チェックイン ポリシーを使用すれば、コードのチェックインごとに静的コード分析を自動的に実行して、関連規則が必ず守られるようにすることができます。コード分析ポリシーに手を加えて、多種多様な規則をチェックすることができます。たとえば、設計、相互運用性、保守容易性、移動性、命名規則、信頼性などについて定めた規則のチェックを行うことができます。

チーム プロジェクトでコード分析チェックイン ポリシーを実施するには

1. [チーム エクスプローラ] でチーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] をポイントしてから、[**ソース管理**] をクリックします。
2. [**チェックイン ポリシー**] タブをクリックしてから、[**追加**] をクリックします。
3. [**チェックイン ポリシーの追加**] ダイアログ ボックスで、[**コード分析**] を選択してから、[**OK**] をクリックします。
4. [**コード分析ポリシー エディタ**] で、[**C/C++ コード分析 (/analyze) を強制**] または [**マネージ コードのコード分析を強制**] を選択します。マネージ コードとアンマネージ コードを組み合わせて構成されているプロジェクトの場合は、両方とも選択します。
5. マネージ コード分析を選択した場合、必要とするコーディング基準に基づいて、マネージ コード分析に対して義務付ける規則設定を構成します。これにより、どの規則を実施するかが正確に決まります。

また、既定では利用できないチェックを実行するために、カスタム チェックイン ポリシーを作成する

こともできます。たとえば、禁止されている API 呼び出しなどのコード パターンを禁止することも、ソース コード中のどこにかっこを記入すればよいかなどの、チームの具体的なコーディング スタイルガイドラインを実施するポリシーを作成することもできます。

参考資料

- カスタム チェックイン ポリシーの作成と使用方法の詳細については、このガイドの「方法 ：TFS のカスタム チェックイン ポリシーを作成するステップ」を参照してください。
- チェックイン ポリシーをカスタマイズする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインでの選択済みパターンを禁止するサンプル コードの詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインでコメントを実施するサンプル コードは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーを登録する方法の詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。

プロジェクト管理

- **Microsoft Project を使用したプロジェクトの管理方法**
- **Microsoft Excel を使用したプロジェクトの管理方法**
- **最小限のプロセス テンプレートの作成方法**
- **プロセス テンプレートのカスタマイズ方法**
- **プロセス テンプレートでの作業項目の種類のカスタマイズ方法**
- **既存のチーム プロジェクトでの作業項目の種類のカスタマイズ方法**
- **イテレーションの作成方法**

- **領域の作成方法**
- **チェックイン イベント通知の追加方法**
- **レポート ダッシュボードのセットアップ方法**
- **ソース管理リポジトリでのフォルダの作成方法**
- **Team Foundation Server からのプロジェクトの削除方法**

Microsoft Project を使用したプロジェクトの管理方法

タスクの作成とスケジュール設定、タスクの依存関係の展開、リソースの負荷分散、および終了日の見積もりを行うには、Microsoft Office プロジェクトを使用します。この機能を使用して、プロジェクトをトラッキングすることができます。

プロジェクトをトラッキングするには、以下のような大まかなステップを実行する必要があります。

- プロジェクト計画を作成します。
- 一連のタスクを作成し、そのスケジュールをたて、それを Team Foundation Server に対して発行します。
  - それらのタスクは、該当する開発者の作業項目キュー内に示されます。
  - チーム メンバはそれぞれのタスクに従事し、作業項目の状況を設定することによって、進行状況を [チーム エクスプローラ] でレポートします。
- プロジェクト計画を最新の情報に更新して、最新情報を取り出します。これで、Microsoft Project でのプロジェクトの進行状況をトラッキングできるようになります。

TFS に対してプロジェクト計画を発行するには

1. 新しいプロジェクト計画を作成する場合、Microsoft Office プロジェクトで通常行うのと同様に、タスク、継続時間、リソースの割り当て、依存関係、およびその他の詳細をセットアップします。
2. Microsoft Office プロジェクトで、[**チーム**] メニューの [**チーム プロジェクトの選択**] をクリックします。
3. チーム プロジェクト用の Team Foundation Server を選択します。
4. チーム プロジェクトを選択します。

5. [**OK**] をクリックします。
6. [**作業項目の種類**] 列で、TFS に対して発行する予定の作業項目ごとに、作業項目の種類を設定します。
7. TFS に対して発行したくないサマリ タスクに対しては、[**同期**] 列で [**発行しない**] を選択します。
8. 複数のリソースに割り当てられているタスクがある場合、1 つのリソースに割り当て可能な別々のタスクに分割します(Team Foundation Server は現在、複数のリソースへの作業項目の割り当てをサポートしていません)。必要であれば、別々のタスクをサマリ タスクにグループ化して、計算を自動化する機能を利用することもできます。
   タスクをグループ化するには、TFS で区分のセットを作成し、[**区分**] 列を設定してタスクをグループ化します。TFS で区分を作成する方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181479(VS.80).aspx の「方法 : チーム プロジェクトの区分を変更する」を参照してください。
9. [**作業項目**] ツール バーの [**発行**] をクリックして、プロジェクト計画を TFS に対して発行します。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms244368(VS.80).aspx の「Microsoft Project で作業項目を操作する」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181676(VS.80).aspx の「作業項目を Microsoft Excel や Microsoft Project でインポートする」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms244373(VS.80).aspx の「チーム プロジェクトを Microsoft Excel や Microsoft Project でトラッキングする」を参照してください。

Microsoft Excel を使用したプロジェクトの管理方法

Microsoft Office Excel® を使用して、要件、シナリオ、問題点、バグ、リスク、および作業項目の保管、並べ替え、フィルタ、および管理を行います。

作業項目リストを作成する方法は 2 通りあります。一方のアプローチでは、チーム エクスプローラから作業項目クエリを選択してから、データバインドの新しいスプレッドシートを作成することができます。新しいスプレッドシートには、クエリから得たデータを取り込んだ作業項目リストが含まれています。また、アドインを使用してプロジェクトを選択し、作業項目をインポートすることによって、Excel 内で作業項目を作成することもできます。

Excel を使って作業項目を作成するには

1. Microsoft Office Excel で、[**チーム**] メニューの [**新しい一覧**] をクリックします。
2. [**Team Foundation Server に接続**] で接続先のサーバーを選ぶか、または [**サーバー**] をクリックしてサーバーの情報を入力します。
3. [**チーム プロジェクト**] で、TFS サーバー上の処理したいチーム プロジェクトを選択します。ドキュメントがそのチーム プロジェクトにバインドされます。
4. [**OK**] をクリックします。
5. 目的のリストの種類を選択します。クエリ リストを作成するには、[**クエリ リスト**] オプションを選択してから、[**クエリの選択**] ドロップダウン リストのチーム クエリをクリックします。
6. 新しい作業項目リストに表示したい列を選択します。
7. 任意の作業項目をインポートします。
   詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181676(VS.80).aspx の「作業項目を Microsoft Excel や Microsoft Project でインポートする」を参照してください。
8. 次に、スプレッドシートを保管するか、または [**チーム**] メニュー上の [**変更の発行**] をクリックして、作業項目データベースに対して新しい作業項目を発行します。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181694(VS.80).aspx の「Microsoft Excel で作業項目を操作する」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181695(VS.80).aspx の「方法 ： 作業項目リストを作成する」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181676(VS.80).aspx の「作

業項目を Microsoft Excel や Microsoft Project でインポートする」を参照してください。

最小限のプロセス テンプレートの作成方法

ソース管理以外で、TFS のプロジェクト管理の関連機能のサポートを必要としない場合、最小限のプロセス テンプレートで十分要件は満たされます。最小限のプロセス テンプレートを作成するには、プロセス テンプレート マネージャを使用してテンプレートをローカル コンピュータにダウンロードし、使用する予定のないセクションを除去するためそのテンプレートを編集して、テンプレートを元のサーバーにアップロードして戻します。

ソース管理のみをサポートするカスタムのプロジェクト テンプレートを作成するには

1. [チーム エクスプローラ] で、サーバー名を右クリックし、[**Team Foundation Server の設定**] をクリックしてから、[**プロセス テンプレート マネージャ**] をクリックします。
2. 編集したいテンプレートを選択して、[**ダウンロード**] を選択します。
3. テンプレートを保存したフォルダから、編集用として Process Template.xml を開きます。
4. **<name>** 要素の編集によって、プロセス名を任意の名前に変更します。
5. <plugins> セクションで、**Reporting**、**Portal**、および **WorkItemTracking** のプラグインを削除します。
6. <groups> セクションで、**Reporting**、**Portal**、および **WorkItemTracking** のグループを削除します。
7. **VersionControl** グループで、<dependencies> セクションを見つけ出して、**WorkItemTracking** の依存関係を削除します。
8. テンプレートの保存先のフォルダで、Reports、Windows SharePoint Services、および WorkItem Tracking フォルダを削除します。
9. 変更内容をすべて保存します。
10. [チーム エクスプローラ] で、サーバー名を右クリックし、[**Team Foundation Server の設定**] をクリックしてから、[**プロセス テンプレート マネージャ**] をクリックします。
11. [**アップロード**] を選択し、アップロードしたいテンプレートを選択します。

変更後のプロセス テンプレートをアップロードし終わったら、そのテンプレートを使用して新しいチーム プロジェクトを作成することができます。

参考資料

- 詳細については、このガイドの「第 13 章 – プロセス テンプレートの説明」を参照してください。
- 詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243782(VS.80).aspx の「プロセス テンプレートのカスタマイズ」を参照してください。

プロセス テンプレートのカスタマイズ方法

プロセス テンプレートをカスタマイズすることによって、新しいチーム プロジェクトの作成時にセットアップされる既定の作業項目の種類、セキュリティ設定、ソース管理の設定、およびレポートを、社内のプロセス要件にあわせて変更します。

プロセス テンプレートをカスタマイズするには

1. すぐに使えるプロセス テンプレートを見直して、社内プロセスに最も適したものを選択します。
2. 選択したプロセス テンプレートをダウンロードします。
3. そのプロセス テンプレートをカスタマイズし、必要に応じて作業項目の種類、セキュリティ設定、およびソース管理設定を変更します。
4. カスタマイズ後のテンプレートを TFS にアップロードします。
5. 加えた変更が、プロセス要件を満たすことを確認します。

上記のプロセスを実装する場合、次の 2 つのオプションが提供されます。

- **XML ファイルを手動でカスタマイズします。**これは手動プロセスなので、エラーを生じやすいですが、きめ細かく制御しながらプロセス テンプレートをカスタマイズすることができます。詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243782(VS.80).aspx の「プロセ

ス テンプレートのカスタマイズ」を参照してください。

- **Microsoft Visual Studio 2005 Team Foundation Server Power Tool に付属しているプロセス テンプレート エディタを使用します。**最新バージョンの TFS Power Tool には、プロセス テンプレートの表示およびカスタマイズのためのグラフィカル ツールが用意されています。TFS への接続時にこのツールを使用して、作業項目の種類の定義や、アクティブになっているプロジェクトのグローバル リストをカスタマイズすることができます。詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

参考資料

- 詳細については、このガイドの「第 13 章 – プロセス テンプレートの説明」を参照してください。
- 詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243782(VS.80).aspx の「プロセス テンプレートのカスタマイズ」を参照してください。

プロセス テンプレートでの作業項目の種類のカスタマイズ方法

最新バージョンの TFS Power Tool で利用できる Process Editor ツールを使用して、作業項目の種類をカスタマイズします。TFS Power Tool は、http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspxからダウンロードできます。

作業項目の種類をカスタマイズするには

1. 次のようにして、自身の要件を最も的確に満たすプロセス テンプレートをダウンロードします。
    a. Visual Studio で、[**チーム**] メニューから [**Team Foundation Server の設定**] を選択します。
    b. [**プロセス テンプレート マネージャ**] をクリックします。
    c. [**プロセス テンプレート マネージャ**] ダイアログ ボックスで、変更するプロセス テンプ

レートを選択してから、[**ダウンロード**] をクリックします。

   d. [**プロセス テンプレートのダウンロード**] ダイアログ ボックスで、テンプレートの保存先としてローカル ドライブのフォルダ場所を選択し、[**保存**] をクリックします。

2. 次のようにして、Process Editor でプロセス テンプレートを開きます。
   a. Visual Studio で、[**チーム**] メニューをクリックします。
   b. [**Process Editor**] をクリックしてから、[**Open Process Template**] をクリックします。
   c. [**Open Process Template fileset**] ダイアログ ボックスで、ダウンロードしたプロセス テンプレートに移動し、[**Open**] をクリックします。
   d. ProcessTemplate.xml ファイルが Visual Studio によって読み込まれます。
   e. カスタマイズの対象の methodology の [**Name**] を設定します。
3. プロセス テンプレート エクスプローラで、[**Work Item Tracking**] をクリックします。
4. 右側のペインで [**Type Definitions**] タブをクリックします。
5. 新しい作業項目を作成するには、右側のペインのツールバーの [**Add**] をクリックします。
6. [**New Work Item Type**] ダイアログ ボックスで、作業項目の種類の名前を入力し、既存の作業項目の種類を [**Copy From**] ドロップダウン リストで選択します。
   新しい作業項目の種類が作成されて、右側のペインの [**Item List**] タブ内の [**Type Definitions**] 中で有効になります。
7. 変更を保存するには、[**File**] メニューの [**Save**] をクリックします。
8. 必要に応じて、新しい作業項目の種類または既存の作業項目の種類を対象に、属性またはフィールドの追加または除去を行いたい場合は、次のようにします。
9. [**Type Definitions**] タブで、編集したい作業項目の種類を右クリックしてから、[**Open**] タブをクリックします。
   選択した作業項目の種類が、新しい Visual Studio ウィンドウに開かれます。
10. 必要に応じて、属性を追加または削除します。

参考資料

- TFS Power Tool をダウンロードするには、http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx にアクセスしてください。

- 作業項目の詳細については、このガイドの「第 12 章 – 作業項目の説明」を参照してください。
- Process Editor ツールを使用して作業項目の種類をカスタマイズする方法の詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

既存のチーム プロジェクトでの作業項目の種類のカスタマイズ方法

既存の作業項目の種類の編集には、2 つのオプションがあります。つまり、コマンド ラインを使用するか、または、TFS Power Tool の一部として利用できるプロセス テンプレート エディタ ツールを使用します。

コマンド ラインから作業項目の種類を編集するには、witexport ツールと witimport ツールを使用します。これらのツールは、チーム エクスプローラのインストール先のコンピュータ上の "%programfiles%¥Program Files¥Microsoft Visual Studio 8¥Common7¥IDE" に置かれています。

作業項目の種類をプロジェクトからエクスポートし、編集し、同じ種類を再インポートするには

1. 次のようにして **witexport** を実行し、作業項目の種類をエクスポートします。
   witexport /f task.xml /t http://TFSServer:8080 /p MyTeamProject /n Task
2. 作業項目の種類の定義を編集します。
3. 次のようにして **witimport** を実行し、変更後のこの種類をインポートします。
   witimport /f task.xml /t http://TFSServer:8080 /p MyTeamProject

上記のコマンド ラインは、TFSServer というサーバーへ MyTeamProject から **Task** 作業項目の種類をエクスポートする方法を示しています。

プロセス テンプレート エディタ ツールを使って作業項目の種類を編集するには

1. 次のようにして、編集したい作業項目の種類をエクスポートします。
   a. Visual Studio で、[**チーム**] メニューから [**Process Editor**] を選択し、[**Work Item**

Types] をクリックしてから、[**Export WIT**] を選択します。

b. [**Connect to Team Foundation Server**] ダイアログ ボックスに、サーバーの URL を入力します。

c. [**Select Work Item Type**] ダイアログ ボックスで、エクスポートしたい作業項目の種類を選択します。

d. 作業項目の種類を保存します。

e. 同じように編集したいすべてのグローバル リストを保存します。

f. 作業項目の種類を編集してから、編集した内容を保存します。

2. 次のようにして、編集後の作業項目の種類をインポートします。

a. Visual Studio で、[**チーム**] メニューから [**Process Editor**] を選択し、[**Work Item Types**] をクリックしてから、[**Import WIT**] を選択します。

b. [**Connect to Team Foundation Server**] ダイアログ ボックスに、サーバーの URL を入力します。

c. [**Import Work Item Type**] ダイアログ ボックスで、編集した作業項目の種類を見つけてから、変更後の作業項目の種類のインポート先のチーム プロジェクトを選択します。

d. [**OK**] をクリックし、新しい作業項目の種類の定義をインポートします。

参考資料

- TFS Power Tool をダウンロードするには、http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx にアクセスしてください。
- Process Editor ツールを使用して作業項目の種類をカスタマイズする方法の詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。
- witimport の使用法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253163(VS.80).aspx の「witimport」を参照してください。
- witexport の使用法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253051(VS.80).aspx の「witexport」を参照してください。

イテレーションの作成方法

イテレーションは、作業項目をグループにまとめるのに使用します。したがって、作業項目の作成や作業項目のクエリとレポートに影響を与えます。

イテレーションを作成するには

1. [チーム エクスプローラ] で、チーム プロジェクトをクリックします。
2. [**チーム**] メニューで [**チーム プロジェクトの設定**] をポイントし、次に [**区分およびイテレーション**] をクリックします。
3. [**区分およびイテレーション**] ボックスの [**イテレーション**] タブをクリックします。
4. ツール バーの [**子ノードの追加**] ボタンをクリックします。
5. 新しいノードを右クリックし、[**名前の変更**] をクリックしてから、イテレーション名を入力します。
6. [**イテレーション**] ノードをクリックします。
7. ステップ 2、3、および 4 を繰り返して、プロジェクトで識別した追加のイテレーションを作成します。
8. [**閉じる**] をクリックします。

注意: Microsoft Solution Framework (MSF) for Agile Software Development (MS Agile) プロセステンプレートには、3 種類の事前定義されたイテレーションが組み込まれています。これらのイテレーションを削除しても、または新たに作成する代わりにその名前を変更してもかまいません。あるいは、未変更のままにすることもできます。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/dhedaeb2(VS.80).aspx の「チュートリアル : 新しいチーム プロジェクトの作成」を参照してください。

区分の作成方法

チーム プロジェクトの作業をサブ区分に編成するには、区分を作成します。たとえば、プロジェクト作業を UI、アプリケーション、およびデータベースなどの区分に編成することができます。区分を作成し終わったら、シナリオや作業項目をその区分に割り当てることができます。

単一のルート レベルの区分から開始し、その後プロジェクトが完成に近づくにつれて、必要時にはいつでも区分を作成することができます。

プロジェクトの区分を作成するには

1. [チーム エクスプローラ] で、チーム プロジェクトをクリックします。
2. [**チーム**] メニューで [**チーム プロジェクトの設定**] を指し、次に [**区分およびイテレーション**] をクリックします。
3. [**区分およびイテレーション**] ボックスの [**区分**] タブをクリックします。
4. ツール バーの [**子ノードの追加**] ボタンをクリックします。
5. 新しいノードを右クリックし、[**名前の変更**] をクリックしてから、任意の区分名を入力します。
6. [**区分**] ノードをクリックします。
7. ステップ 2、3、および 4 を繰り返して、追加の区分を作成します。このようにして、プロジェクト構造の階層を作成することができます。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/dhedaeb2(VS.80).aspx の「チュートリアル : 新しいチーム プロジェクトの作成」を参照してください。

チェックイン イベント通知の追加方法

チェックイン通知をサブスクライブするには、TFS サーバーの %programfiles%¥Microsoft Visual Studio 2005 Team Foundation Server¥TF Setup¥BisSubscribe.exe から利用できる **bissubscribe** ツールを使用します。

チェックイン イベント通知をセットアップするには

1. コマンド プロンプトを開いて、以下の場所 C:¥Program Files¥Microsoft Visual Studio 2005 Team Foundation Server¥TF Setup¥ に変更します。
2. 電子メール通知をセットアップしたい場合、次のようなコマンドを使用します。
   bissubscribe /eventType CheckinEvent /address someone@domain.com /deliveryType EmailHtml /domain http://TFSRTM:8080
3. Web サービス通知をセットアップしたい場合、次のようなコマンドを使用します。
   bissubscribe /eventType CheckinEvent /address http://TFSRTM:8080/ci/notify.asmx /deliveryType Soap /domain http://TFSRTM:8080
4. エラーが表示された場合や、正しく登録したことを確認したい場合、次のようにします。
   a. SQL Server Management Studio を開きます。
   b. **tfsIntegration** データベースを開きます。
   c. **tbl_subscription** テーブルを見直します。

tbl_subscription テーブルは、すでにサブスクライブ済みのイベントをすべて一覧表示しています。このテーブルを参照し、サブスクライブ済みのイベントのエントリを探してください。このテーブルからエントリを削除することにより、登録済みのイベントのサブスクライブを中止することができます。また、次のように、**bissubscribe** を実行し、 **/unsubscribe** コマンド ライン パラメータおよびイベントの **id** を渡して、イベントのサブスクライブを中止することもできます。

bissubscribe /delete/id **[id]** /server http://TFSRTM:8080

参考資料

- 詳細については、このガイドの「Visual Studio Team Foundation Server で継続的な統合ビルドをセットアップする方法」を参照してください。
- 詳細については、http://blogs.msdn.com/buckh/archive/2006/09/29/checkinevent-path-filter.aspx の「Adding a path filter to a CheckinEvent subscription using bissubscribe」を参照してくださ

い。

- 詳細については、http://blogs.msdn.com/khushboo/archive/2006/01/04/509122.aspx の「Continuous Integration using Team Build」を参照してください。

レポート ダッシュボードのセットアップ方法

チーム プロジェクト ポータル Microsoft Office SharePoint® サイトを変更して、1 つの場所でさまざまなプロジェクト情報を提供できるレポート ダッシュボードを作成します。

たとえば、次のようなレポートをレポート ダッシュボードに揃えておけば便利です。

- 残存作業
- 品質指標
- バグ率
- プロジェクト速度

SharePoint ポータル ページに新しいレポートを追加することができます。それには、ページに表示したいレポートごとにレポート ビューア Web パーツを追加します。

チーム プロジェクト ポータルを変更してレポート ダッシュボードを作成するには

1. Microsoft Office SharePoint およびレポート サービスのインストール パッケージに付属している stsadm.exe ツールと RSWebParts.cab を使用して、レポート サーバーにレポート ビューア Web パーツをインストールします。
   - STSADM.EXE は、パス C:¥Program Files¥Common Files¥Microsoft Shared¥web server extensions¥60¥BIN に置かれています。
   - RSWebParts.Cab は、パス C:¥ Program Files¥Microsoft SQL Server¥90¥Tools¥Reporting Services¥SharePoint に置かれています。
     例: STSADM.EXE -o addwppack -filename "C:¥ Program Files¥Microsoft SQL Server¥90¥Tools¥Reporting Services¥SharePoint¥RSWebParts.cab" -globalinstall
2. [チーム エクスプローラ] で、プロジェクトを右クリックしてから、[**プロジェクト ポータルの表**

示] をクリックします。

3. [**共有ページの変更**] をクリックし、[**参照**] をポイントしてから、[**Web パーツの追加**] をクリックします。
4. [**仮想サーバー ギャラリー**] をクリックします。
5. [**Web パーツの一覧**] で、[**Report Viewer**] を選択します。
6. [**追加**] をクリックします。
7. たとえば、http://<*report server*>/reports などの、レポート マネージャ名を入力します。
8. <*my project*>/Quality Indicators などの、表示するレポートのパスを入力します。

参考資料

- レポート ビューア Web パーツの追加の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms159772(SQL.90).aspx の「SharePoint 2.0 Web パーツによるレポートの表示」を参照してください。
- チーム プロジェクト ポータルの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms242883(VS.80).aspx の「チーム プロジェクト ポータルの使用」を参照してください。

ソース管理リポジトリでのフォルダの作成方法

チーム エクスプローラを使用して、ソース管理リポジトリ内にフォルダを作成できますが、クライアント上のワークスペースでフォルダ構造を作成してから保留中の変更をチェックインすることもできます。

サーバー上でフォルダ構造を作成するには

1. [チーム エクスプローラ] で、チーム プロジェクトを展開します。
2. チーム プロジェクトの下の [**ソース管理**] をダブルクリックします。
3. ソース管理エクスプローラでルート ノードを選択し、[ローカル パス] ペイン内を右クリックしてから、[**新しいフォルダ**] をクリックします。

4. ルート フォルダの名前を入力し、Enter キーを押します。
5. ステップ 3 と 4 を繰り返して、必要なフォルダをソース管理リポジトリ内に作成します。

クライアント上でフォルダ構造を作成するには

1. ローカル コンピュータ上でワークスペースのマッピングを作成します。
2. プロジェクトで必要なフォルダ構造を作成します。
   保留中の変更をはじめてチェックインすると、フォルダ構造がサーバーにコピーされます。

参考資料

- ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ソース ツリーの作成の詳細については、このガイドの「Visual Studio Team Foundation Server でソース ツリーを作成する方法」を参照してください。

Team Foundation Server からのプロジェクトの削除方法

チーム エクスプローラを使用して、プロジェクトを削除することはできません。その代りに、**TfsDeleteProject** コマンド ライン ツールを使用する必要があります。このツールは、チーム エクスプローラのインストール先のコンピュータの Program Files¥Microsoft Visual Studio 8¥Common7¥IDE に置かれています。

チームT TFS からプロジェクトを削除するには

1. コマンド プロンプトを開いて、以下の場所
   C:¥Program Files¥Microsoft Visual Studio 2005 Team Foundation Server¥TF Setup¥ に切り替えます。
2. 次のように、**TfsDeleteProject** を実行して、プロジェクトを削除します。

TfsDeleteProject /server:TfsServer TeamProjectName

参考資料

- プロジェクトの削除の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181482(VS.80).aspx の「TFSDeleteProject」を参照してください。

Team Foundation プロジェクト管理の参照資料

- MSF プロセス テンプレートの詳細については、http://msdn2.microsoft.com/ja-jp/teamsystem/aa718801.aspx の「Process Templates」を参照してください。

プラクティス: レポート

索引

管理

- レポート ダッシュボードのセットアップ方法
- レポートに対する許可の設定方法

作成/カスタマイズ

- 既存のレポートのカスタマイズ方法
- *Visual Studio* での新しいレポートの作成方法
- *Excel* での新しいレポートの作成方法
- スケジュールされたレポート スナップショットの作成方法
- レポート サブスクリプションの作成方法

- 既存のプロセス テンプレートへの新しいレポートの追加方法

表示

- プロジェクトの状況の分析方法
- アプリケーション品質の分析方法
- 残存作業の検討方法
- ビルド状態の検討方法
- バグおよびテスト結果の検討方法
- イテレーションでの実際の作業との比較によるスケジュール済み作業の検討方法
- ファイルの最終編集者の判別方法
- 開発者が行ったすべてのコード変更の発見方法
- ファイルに加えられたすべてのコード変更の発見方法
- 特定の作業項目に関連したすべてのコード変更の発見方法
- コード チャーン メトリックの生成方法
- ファイル数、コードの行数、およびプロジェクト数などのワークスペース メトリックの生成方法

管理

- **レポート ダッシュボードのセットアップ方法**
- **レポートに対する許可の設定方法**

レポート ダッシュボードのセットアップ方法

チーム プロジェクトの Microsoft® Office SharePoint® ポータル サイトを変更して、1 つの場所でさまざまなプロジェクト情報を表示できるレポート ダッシュボードを作成します。

たとえば、次のようなレポートをレポート ダッシュボードに揃えておけば便利です。

- 残存作業
- 品質指標

- バグ率
- プロジェクト速度

SharePoint ポータル ページに新しいレポートを追加することができます。それには、ページに表示したい各レポートごとにレポート ビューア Web パーツを追加します。

チーム プロジェクト ポータルを変更してレポート ダッシュボードを作成するには

1. SharePoint およびレポート サービスのインストール パッケージに付属している stsadm.exe ツールと RSWebParts.cab を使用して、レポート サーバーにレポート ビューア Web パーツをインストールします。
   - STSADM.EXE は、パス C:¥Program Files¥Common Files¥Microsoft Shared¥web server extensions¥60¥BIN に置かれています。
   - RSWebParts.Cab は、パス C:¥ Program Files¥Microsoft SQL Server¥90¥Tools¥Reporting Services¥SharePoint に置かれています。
   - 例: STSADM.EXE -o addwppack -filename "C:¥ Program Files¥Microsoft SQL Server¥90¥Tools¥Reporting Services¥SharePoint¥RSWebParts.cab" -globalinstall
2. [チーム エクスプローラ] で、プロジェクトを右クリックします。
3. [**プロジェクト ポータルの表示**] をクリックします。
4. [**共有ページの変更**] をクリックします。
5. [**参照**] をポイントし、[**Web パーツの追加**] をクリックします。
6. [**仮想サーバー ギャラリー**] をクリックします。
7. [**Web パーツの一覧**] で、[**Report Viewer**] を選択します。
8. [**追加**] をクリックします。
9. たとえば、http://<*report server*>/reports などの、レポート マネージャ名を入力します。
10. <*my project*>/Quality Indicators などの、表示するレポートのパスを入力します。

参考資料

- レポート ビューア Web パーツの追加の詳細については、

http://msdn2.microsoft.com/ja-jp/library/ms159772(SQL.90).aspx の「SharePoint 2.0 Web パーツによるレポートの表示」を参照してください。

- チーム プロジェクト ポータルの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms242883(VS.80).aspx の「チーム プロジェクト ポータルの使用」を参照してください。

レポートに対する許可の設定方法

レポートのアクセス許可を変更して、レポートを編集または表示できるメンバを定義できます。レポートのアクセス許可を設定するには、Microsoft SQL Server™ Reporting Services Content Manager のロールのメンバでなければなりません。

チーム プロジェクト内のすべてのレポートのアクセス許可を設定するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. [**プロパティ**] タブをクリックします。
4. [**セキュリティ**] をクリックします。
5. [**アイテムのセキュリティを編集**] をクリックします。
6. 既にレポートに対して定義されているロールのセキュリティを編集する場合、[**編集**] をクリックします。
7. 一覧中にないロールのセキュリティを定義したい場合、[**新しいロールの割り当て**] をクリックします。

単一のレポートのアクセス許可を設定するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. レポート サイトで、アクセス許可を設定するレポートを選択します。
4. [**プロパティ**] タブをクリックします。

5. [**セキュリティ**] をクリックします。
6. [**アイテムのセキュリティを編集**] をクリックします。
7. 既にレポートに対して定義されているロールのセキュリティを編集する場合、[**編集**] をクリックします。
8. 一覧中にないロールのセキュリティを定義したい場合、[**新しいロールの割り当て**] をクリックします。

参考資料

- レポートのアクセス許可の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181645(VS.80).aspx の「方法 : レポートのアクセス許可を設定する」を参照してください。

作成/カスタマイズ

- **既存のレポートのカスタマイズ方法**
- **Visual Studio での新しいレポートの作成方法**
- **Excel での新しいレポートの作成方法**
- **スケジュールされたレポート スナップショットの作成方法**
- **レポート サブスクリプションの作成方法**
- **既存のプロセス テンプレートへの新しいレポートの追加方法**

既存のレポートのカスタマイズ方法

SQL Server 2005 クライアント ツールに付属している Visual Studio (Business Intelligence Development Studio) 内の SQL Server 2005 Reporting Services Designer を使用してレポートを変更することができます。多くの場合、既存のレポートの変更のほうが、新しいレポートの作成よりも簡単です。

TFS で既存のレポートをカスタマイズするには

1. 次のようにして、レポート プロジェクトを作成します。
    a. Visual Studio で、[**ファイル**] をクリックし、[**新規作成**] をクリックしてから、[**プロジェクト**] をクリックします。
    b. **ビジネス インテリジェンス プロジェクト** の種類を選択します。
    c. [**レポート サーバー プロジェクト**] テンプレートを選択します。
    d. プロジェクトの [**名前**] および [**場所**] を設定してから、[**OK**] をクリックします。
2. 次のようにして、カスタマイズするレポートをエクスポートします。
    a. チーム プロジェクトを右クリックしてから、[プロジェクト ポータルの表示] をクリックします。
    b. ポータル Web サイトの左側のクイック起動バー内の [レポート] をクリックします。
    c. カスタマイズするレポートをクリックします。
    d. [プロパティ] をクリックします。
    e. [編集] を選択します。
    f. レポート .rdl ファイルを、ステップ 1 で作成したレポート プロジェクト フォルダに保管します。
3. 次のようにして、データ ソースを追加します。
    a. ウェアハウス データ ソースを作成するには
        i. Visual Studio ソリューション エクスプローラで、[**共有データ ソース**] を右クリックしてから、[**新しいデータ ソースの追加**] をクリックします。
        ii. [**全般**] タブの [**名前**] ボックスに **TfsReportDS** と入力します。
        iii. [**種類**] ボックスの一覧の [**Microsoft SQL Server**] をクリックします。
        iv. [**編集**] をクリックします。
        v. データ層サーバー名を入力します。
        vi. [**TFSWarehouse**] データベースを選択します。
        vii. [**OK**] ボタンをダブルクリックして、データ ソースを追加します。
    b. OLAP データ ソースを作成するには
        i. ソリューション エクスプローラで、[共有データ ソース] を右クリックしてから、[新しいデータ ソースの追加] をクリックします。
        ii. [全般] タブの [名前] ボックスに TfsOlapReportDS と入力します。

iii. [種類] ボックスの一覧の [Microsoft SQL Server Analysis Services] をクリックします。

iv. [編集] をクリックします。

v. データ層サーバー名を入力します。

vi. [TFSWarehouse] データベースを選択します。

vii. [OK] ボタンをダブルクリックして、データ ソースを追加します。

4. 次のようにして、レポートをプロジェクトに追加します。
   a. [ソリューション エクスプローラ] で、[レポート] 右クリックしてから、[追加] → [既存の項目] をクリックします。
   b. ステップ 2 でエクスポートした rdl ファイルを見つけます。

5. 次のようにして、レポートを変更します。
   a. [データ ペイン] 内のクエリ ステートメントを変更します。
   b. 新しいメジャーまたはメンバを [データ ペイン] 内にドラッグします。
   c. [レイアウト ペイン] でレポート レイアウトを変更します。

注意: チーム レポート サイトに用意されているレポート ビルダを使用できますが、このツールは、Visual Studio シナリオの場合は十分にサポートされていないので、推奨されていません。

参考資料

- 詳細については、このガイドの「Visual Studio Team Foundation Server でレポートをカスタマイズする方法」を参照してください。

Visual Studio での新しいレポートの作成方法

SQL Server 2005 クライアント ツールに付属している Visual Studio (Business Intelligence Development Studio) 内の SQL Server 2005 Reporting Services Designer を使用して、レポートを作成することができます。

変更すれば自身のニーズを満たせる既存のレポートがない場合、新しいレポートを作成します。多くの場合、既存のレポートの変更のほうが、新しいレポートの作成よりも簡単です。

TFS で新しいレポートを作成するには

1. 次のようにして、レポート プロジェクトを作成します。
    a. Visual Studio で、[ファイル] をクリックし、[新規作成] をクリックしてから、[プロジェクト] をクリックします。
    b. ビジネス インテリジェンス プロジェクトの種類を選択します。
    c. [レポート サーバー プロジェクト] テンプレートを選択します。
    d. プロジェクトの [**名前**] および [**場所**] を設定してから、[**OK**] をクリックします。
2. 次のようにして、データ ソースを追加します。
    a. ウェアハウス データ ソースを作成するには
        i. Visual Studio ソリューション エクスプローラで、[共有データ ソース] を右クリックしてから、[新しいデータ ソースの追加] をクリックします。
        ii. [全般] タブの [名前] ボックスに TfsReportDS と入力します。
        iii. [種類] ボックスの一覧の、[Microsoft SQL Server] をクリックします。
        iv. [編集] をクリックします。
        v. データ層サーバー名を入力します。
        vi. [TFSWarehouse] データベースを選択します。
        vii. [OK] ボタンをダブルクリックして、データ ソースを追加します。
    b. オンライン分析処理 (OLAP) データ ソースを作成するには
        i. ソリューション エクスプローラで、[共有データ ソース] を右クリックしてから、[新しいデータ ソースの追加] をクリックします。
        ii. [全般] タブの [名前] ボックスに TfsOlapReportDS と入力します。
        iii. [種類] ボックスの一覧の [Microsoft SQL Server Analysis Services] をクリックします。
        iv. [編集] をクリックします。
        v. データ層サーバー名を入力します。
        vi. [TFSWarehouse] データベースを選択します。

vii. [OK] ボタンを ダブルクリックして、データ ソースを追加します。

3. 次のようにして、新しいレポートを作成します。
    a. [ソリューション エクスプローラ] で、[レポート] を右クリックし、[追加] をポイントしてから、[新しい項目] をクリックします。
    b. [レポート] テンプレートを選択します。
    c. レポートに名前を付けてから、[OK] をクリックします。
4. 次のようにして、レポートを変更します。
    a. [レポート デザイナ] が自動的に開かない場合、[ソリューション エクスプローラ] 内でそれをダブルクリックして、変更用のレポートを開きます。
    b. [**データセット**] ボックスの **<新しいデータセット...>** をクリックします。
    c. データ セットに、たとえば TestDataSet などの名前を付けます。
    d. [**TfsOlapReportDS (共用)**] を選択します。
    e. [**OK**] をクリックします。
    f. **[ビルド]** (**[データセット]** ボックスのすぐ下) の横の省略記号 （…） ボタンをクリックしてから、**[Team System]** を選択します。 。

次に、メジャーおよびディメンションを [**データセット**] ツリーからクエリ ペインおよびフィルタ ペインにドラッグして、レポートを変更することができます。[**レイアウト**] タブをクリックして、レポートのレイアウトを変更することができます。また、[**プレビュー**] タブをクリックして、レポートをプレビューすることもできます。

注意: チーム レポート サイトに用意されているレポート ビルダを使用できますが、このツールは、Visual Studio シナリオの場合は十分にサポートされていないので、推奨されていません。

参考資料

- 詳細については、このガイドの「Visual Studio Team Foundation Server のカスタム レポートを作成する方法」を参照してください。

Excel での新しいレポートの作成方法

Microsoft Office Excel® を TFS レポート OLAP キューブに直接接続して、随時レポートを作成することができます。Excel を使用すれば、ピボット テーブルまたはピボット グラフの形式でレポート データを表示することができます。

Excel ピボット テーブル レポートを作成するには

1. Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB プロバイダがインストールされていることを確認します。これは、http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc からインストールできます。
2. Excel を開始します。
3. ピボット テーブル レポートを追加するワークシートを選択します。
4. [**データ**] メニューで [**ピボットテーブルとピボットグラフ レポート**] を選択します。
5. [**外部データ ソース**] を選択します。
6. [**次へ**] をクリックします。
7. [**データの取り出し**] をクリックします。
8. [**OLAP キューブ**] タブをクリックします。
9. **<新しいデータ ソース>** を選択してから、[**OK**] をクリックします。
10. データ ソースの名前を入力します。
11. Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB プロバイダを選択します。
12. [**接続**] をクリックします。
13. [**分析サーバー**] を選択します。
14. たとえば TFSRTM などの、レポート サーバーの名前を入力します。
15. [**次へ**] をクリックします。
16. [**TFSWarehouse**] を選択してから、[**完了**] をクリックします。
17. レポートを作成したいキューブ (たとえば、コード チャーン、作業項目、およびテスト結果) を選択してから、[**OK**] をクリックします。
18. もう一度 [**OK**] をクリックし、[ピボット テーブル/ピボット グラフ ウィザード] に戻ります。

19. [**完了**] をクリックして、ピボット テーブルをワークシートに追加します。

[**ピボットテーブルのフィールド リスト**] を使用して、列およびメジャーをドラッグしてピボット テーブルにドロップします。たとえば、サーバー上の各チーム プロジェクトごとの行カウントを表示するには、次のようにします。

1. 上記のステップ 17 の**コード チャーン** キューブを選択します。
2. ピボット テーブルの [**列フィールド**] セクションに [**TeamProject.TeamProject**] をドラッグします。
3. ピボット テーブルの [**データ項目**] セクションに [**Total Lines**] をドラッグします。

参考資料

- Excel を使用して随時レポートを作成する方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms244713(VS.80).aspx の「Team Foundation Server のレポート作成での Microsoft Excel の使用」を参照してください。
- Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB プロバイダは、http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc からインストールしてください。

スケジュールされたレポート スナップショットの作成方法

スケジュールされたレポート スナップショットを使用して、時間の経過に伴う傾向の理解を深めることができ、さらに、プロジェクトの存続期間全体を通して重要なデータ要素を記憶しておくことができます。

スケジュールされたレポート スナップショットを作成するには

1. [チーム エクスプローラ] で、チーム プロジェクトの [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。

2. レポート サイトからレポートを開きます。
3. [**プロパティ**] タブをクリックします。
4. [**履歴**] リンクをクリックします。
5. いつスナップショットを実行すればよいかのスケジュールをセットアップします。

スケジュールをセットアップした後、そのレポートの [**履歴**] タブ上にスナップショットが示されます。また、[**履歴**] タブ上でスナップショットを手動で作成することもできます。

レポート サブスクリプションの作成方法

レポート サブスクリプションを使用して、レポートを生成し、今後の使用に備えてファイルの共有にエクスポートすることができます。古いレポートを上書きするようにサブスクリプションを設定できますが、時間の経過とともに一連のレポートをビルドして、プロジェクト データのスナップショットを表示することもできます。

レポート サブスクリプションを作成するには

1. [チーム エクスプローラ] で、チーム プロジェクトの [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
2. レポート サイトからレポートを開きます。
3. [**サブスクリプション**] タブをクリックします。
4. [**新しいサブスクリプション**] をクリックして、レポート サブスクリプションを作成します。

既存のプロセス テンプレートへの新しいレポートの追加方法

最新バージョンの Team Foundation Server Power Tool で利用できる Process Editor ツールを使用して、既存のプロセス テンプレートに新しいレポートを追加することができます。Team Foundation Server Power Tool は、http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspx からダウンロードできます。

新しいレポートを追加するには

1. 次のようにして、自身の要件を最も的確に満たすプロセス テンプレートをダウンロードします。
    a. Visual Studio で、[チーム] メニューをクリックしてから、[Team Foundation Server の設定] を選択します。
    b. [プロセス テンプレート マネージャ] をクリックします。
    c. [プロセス テンプレート マネージャ] ダイアログ ボックスで、変更するプロセス テンプレートを選択してから、[ダウンロード] をクリックします。
    d. [プロセス テンプレートのダウンロード]ボックスで、ローカル ドライブのフォルダ場所を選択し、[**保存**] をクリックします。
2. 次のようにして、Process Editor でプロセス テンプレートを開きます。
    a. Visual Studio で、[チーム] メニューをクリックします。
    b. [Process Editor] をクリックしてから、[Open Process Template] をクリックします。
    c. [Open Process Template fileset] ダイアログ ボックスで、ダウンロードしたプロセス テンプレートに移動し、[Open] をクリックします。
    d. これで、ProcessTemplate.xml ファイルが Visual Studio 内で開かれます。
    e. カスタマイズの対象の methodology の [**Name**] を設定します。
3. プロセス テンプレート エクスプローラで、[**Reports**] をクリックします。
4. ツールバーで [**Add**] をクリックします。
5. [**Report**] ダイアログ ボックスの [**Report Detail**] タブにレポートの名前を入力します。
6. [**File Name**] フィールドで、追加したい .rdl ファイルを見つけます。
   他のフィールドは未変更のままにし、[**Properties**] および [**Parameters**] タブ内のデータも一切変更しません。
7. [DataSources] タブから、該当するデータ ソースを入力します。
   TFS に付属しているプロセス テンプレート用の既定のデータ ソースは、/TfsOlapReportDS と /TfsReportDS です。
8. [**OK**] をクリックします。

参考資料

- Team Foundation Server Power Tool は、

http://msdn2.microsoft.com/en-us/vstudio/aa718351.aspxからダウンロードできます。

- Process Editor ツールを使用して作業項目の種類をカスタマイズする方法の詳細については、このガイドの「Visual Studio Team Foundation Server でプロセス テンプレートをカスタマイズする方法」を参照してください。

表示

- **プロジェクトの状況の分析方法**
- **アプリケーション品質の分析方法**
- **残存作業の検討方法**
- **ビルド状態の検討方法**
- **バグおよびテスト結果の検討方法**
- **イテレーションでの実際の作業との比較によるスケジュール済み作業の検討方法**
- **ファイルの最終編集者の判別方法**
- **開発者が行ったすべてのコード変更の発見方法**
- **ファイルに加えられたすべてのコード変更の発見方法**
- **特定の作業項目に関連したすべてのコード変更の発見方法**
- **コード チャーン メトリックの生成方法**
- **ファイル数、コードの行数、およびプロジェクト数などのワークスペース メトリックの生成方法**

プロジェクトの状況の分析方法

[プロジェクト速度] レポートを使用して、プロジェクトの状況を分析することができます。

アプリケーション状態を検討するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開し、[**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
2. レポート サイトで、[**プロジェクト速度**] レポートを選択します。

このレポートは、チームがどのくらい速やかに作業を完了したかを示すとともに、日ごとの変化率を示します。

アプリケーション品質の分析方法

[品質指標] レポートを使用して、アプリケーション品質を分析することができます。

アプリケーション品質を分析するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開し、[**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
2. レポート サイトで、[**品質指標**] レポートを選択します。

このレポートでは、テスト結果、バグ、コード カバレッジ、およびコード チャーンが集められて、プロジェクトの稼働状態を追跡するのに使用できる 1 つのレポートにまとめられています。

残存作業の検討方法

[残存作業] レポートを使用して、残存作業を検討することができます。

残存作業を検討するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開し、[**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
2. レポート サイトで、[**残存作業**] レポートを選択します。

このレポートには時間ごとの、残っている作業、解決済みの作業、終了した作業が示されます。残存作業の傾向から将来を予測することで、コードが完了する時点を予測することができます。

ビルド状態の検討方法

Microsoft Solution Framework (MSF) for CMMI® (MS CMMI) プロセス テンプレートを使用してい

る場合、ビルド レポートを表示して、BVT 結果を見ることができます。

ビルド状態を検討するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. このレポート サイトで、[**ビルド**] レポートを選択します。

このレポートには、利用できるビルドの一覧が示され、ビルド品質やその他の詳細情報が記載されています。

バグおよびテスト結果の検討方法

[バグ (優先度順)] レポートを使用して、バグを検討することができます。このレポートには、低い優先順位のバグと比較した場合の、検出済みの高い優先順位のバグ率が示されます。

[品質指標] レポートを使用して、1 つのレポート中のテスト結果、バグ、コード カバレッジ、およびコード チャーンを表示することができます。

[バグ (優先度順)] レポートまたは [品質指標] レポートを表示するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. [**バグ (優先度順)**] レポートを選択してバグを検討するか、または [**品質指標**] レポートを選択してテスト結果を表示します。

イテレーションでの実際の作業との比較によるスケジュール済み作業の検討方法

[計画していなかった作業] レポートを使用して、実際の作業と比較しながらスケジュール済み作業を検討することができます。このレポートでは、作業全体と残存作業との比較がグラフで示され、計画済みのタスクと未計画のタスクが区別されます。

[計画していなかった作業] レポートを表示するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. レポート サイトで、[**計画していなかった作業**] レポートを選択します。

ファイルの最終編集者の判別方法

ソース管理エクスプローラのソース ファイル履歴を使用して、ファイルの最終編集者を判別することができます。

だれが最後にファイルを編集したかを判別するには

1. [ソース管理エクスプローラ] で、目的のファイルを選択します。
2. そのファイルを右クリックしてから、[**履歴の表示**] をクリックします。
3. [履歴] ペインで、所有者も含め、変更履歴を参照します。

参考資料

- ソース管理エクスプローラの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181370(VS.80).aspx の「ソース管理エクスプローラの使用」を参照してください。

開発者が行ったすべてのコード変更の発見方法

TF History コマンドを使用して、特定の開発者によって行われたプロジェクト中の変更をすべて見つけ出すことができます。

たとえば以下のコマンドは、ユーザー Mario によって行われたすべての変更点を表示します。

tf history $/ /r /user:Mario

上記の例では、リポジトリ全体を検索するために $/ が使用されています。これを $/TeamProjectName に置き換えれば、検索を特定のチーム プロジェクトに限定することができます。

参考資料

- **TF History** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/yxtbh4yh(VS.80).aspx の「History コマンド」を参照してください。

ファイルに加えられたすべてのコード変更の発見方法

ソース管理エクスプローラのソース ファイル履歴を使用して、ファイルに加えられた変更を判別することができます。

ファイルに加えられたすべての変更を判別するには

1. [ソース管理エクスプローラ] で、目的のファイルを選択します。
2. そのファイルを右クリックしてから、[**履歴の表示**] をクリックします。
3. [履歴] ペインで変更履歴を検討して、そのファイルに加えられたすべての変更点を確認します。

参考資料

- ソース管理エクスプローラの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181370(VS.80).aspx の「ソース管理エクスプローラの使用」を参照してください。

特定の作業項目に関連したすべてのコード変更の発見方法

作業項目が、チェックイン時に一連のコード変更に関連付けられていた場合、作業項目の [**リンク**] タ

ブからその変更を表示することができます。

作業項目に関連付けられているコード変更を表示するには

1. 該当する作業項目を開きます。
2. [**リンク**] タブをクリックします。
   その作業項目に変更セットが関連付けられている場合、作業項目の [リンク] ペインにそのセットが一覧で示されます。
3. リンクされている変更セットをダブルクリックして、チェックイン済みのファイルとコメントを表示します。

参考資料

- 変更セットの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181408(VS.80).aspx の「ソース管理変更セットの操作」を参照してください。

コード チャーン メトリックの生成方法

[品質指標] レポートを使用して、コード チャーンの詳細を表示することができます。

[品質指標] レポートを表示するには

1. [チーム エクスプローラ] で、チーム プロジェクト ノードを展開します。
2. [**レポート**] を右クリックしてから、[**レポート サイトの表示**] をクリックします。
3. [**品質指標**] レポートを選択します。

このレポートでは、テスト結果、バグ、コード カバレッジ、およびコード チャーンが集められて、プロジェクトの稼働状態を追跡するのに使用できる 1 つのレポートにまとめられています。
あるいは、Excel を使用して、コード チャーンに関するレポートを生成します。詳細については、こ

の資料の「How to Create a New Report in Excel」を参照してください。

ファイル数、コードの行数、およびプロジェクト数などのワークスペース メトリックの生成方法

さまざまなワークスペース メトリックに関するレポートを作成するには、TFS レポート OLAP キューブに Excel を直接接続します。Excel を使用すれば、ピボット テーブルまたはピボット グラフの形式でレポート データを表示することができます。

Excel ピボット テーブル レポートを作成するには

1. Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB プロバイダがインストールされていることを確認します。これは、http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc からインストールできます。
2. Excel を開始します。
3. ピボット テーブル レポートを追加したいワークシートを選択します。
4. [**データ**] メニューで [**ピボットテーブルとピボットグラフ レポート**] を選択します。
5. [**外部データ ソース**] を選択します。
6. [**次へ**] をクリックします。
7. [**データの取り出し**] をクリックします。
8. [**OLAP キューブ**] タブをクリックします。
9. **<新しいデータ ソース>** を選択してから、[**OK**] をクリックします。
10. データ ソースの名前を入力します。
11. Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB プロバイダを選択します。
12. [**接続**] をクリックします。
13. [**分析サーバー**] を選択します。
14. たとえば TFSRTM などの、レポート サーバーの名前を入力します。
15. [**次へ**] をクリックします。
16. [**TFSWarehouse**] を選択してから、[**完了**] をクリックします。

17. レポートの作成元の [**Code Churn**] キューブを選択してから、[**OK**] をクリックします。
18. もう一度 [**OK**] をクリックし、[ピボット テーブル/ピボット グラフ ウィザード] に戻ります。
19. [**完了**] をクリックして、ピボット テーブルをワークシートに追加します。

[**ピボットテーブルのフィールド リスト**] を使用して、列およびメジャーをドラッグしてピボット テーブルにドロップすることができます。

サーバー上のチーム プロジェクトごとのファイル数を表示するには

1. ピボット テーブルの [**ページ フィールド**] セクションに [**TeamProject.TeamProject**] をドラッグします。
2. ピボット テーブルの [**行フィールド**] セクションに [**File Name.FilePath**] をドラッグします。
3. ピボット テーブルの [**ページ フィールド**] セクションの [**Team Project**] ドロップダウン リストを使用して、各チーム プロジェクトをフィルタします。
   表示される行数に注意してください。これにより、ファイル数を確認できます。

サーバー上の各チーム プロジェクトごとの行数を表示するには

1. ピボット テーブルの [**列フィールド**] セクションに [**TeamProject.TeamProject**] をドラッグします。
2. ピボット テーブルの [**データ項目**] セクションに [**Total Lines**] をドラッグします。

サーバーのチーム プロジェクト数を表示するには

- ピボット テーブルの [**行フィールド**] セクションに [**TeamProject.TeamProject**] をドラッグします。
  表示される行数に注意してください。これにより、プロジェクト数を確認できます。

参考資料

- Excel を使用して一時的なレポートを作成する方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms244713(VS.80).aspx の「Team Foundation Server のレポート作成での Microsoft Excel の使用」を参照してください。
- Microsoft SQL Server 2005 Analysis Services 9.0 OLE DB プロバイダは、http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc からインストールしてください。

Team Foundation レポートの参照資料

- レポートの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194922(VS.80).aspx の「Team Foundation Server のレポート」を参照してください。

プラクティス: ソース管理

索引

バージョン管理へのアクセス

- 非 *Visual Studio* クライアントからのバージョン管理の使用方法
- 一般的なバージョン管理タスクの自動化方法
- オフラインでの作業方法

管理

- プロジェクトに新規開発者を追加する方法
- プロジェクトから離脱した開発者の削除方法
- ソース ツリー内でのアクセス許可の付与方法
- 別のサーバーへの *Team Foundation Server* バージョン管理の移動方法

分岐/ラベル/マージ

- ラベルの使用方法
- 分岐方法
- 分岐構造の計画方法
- リリースのサポートのための分岐の使用方法
- 前のリリースの保守のための分岐の使用方法
- 開発およびビルドのプロセスの安定化のための分岐の使用方法
- 機能の開発の安定化のための分岐の使用方法
- 複数のチームにまたがる開発の安定化のための分岐の使用方法
- 外部依存関係の分離のための分岐の使用方法
- 古いリリースの撤去方法
- マージの実行方法
- 基点のないマージの実行方法
- マージの競合の解決方法

ビルド

- *TFS* を使用した継続的な統合ビルドの実行方法

チェックインおよびチェックイン ポリシー

- ソース管理変更セットの処理方法
- チェックイン前のコーディング基準の実施方法
- チェックイン ポリシーのオーバーライド方法
- チェックインを元に戻す方法
- 競合の解決方法
- 競合の回避方法
- カスタム チェックイン ポリシーの作成方法

チェックアウト、取得、およびロック

- *TFS* に対するコンピュータの同期方法
- 編集用のファイルの準備方法

コードの共有

- コードの共有方法
- 共有バイナリの管理方法

依存関係

- *Web* サービスの依存関係の管理方法
- データベースの依存関係の管理方法

分散/リモート開発

- *TFS* へのインターネット経由のアクセス方法
- *TFS* バージョン管理プロキシのパフォーマンスの最適化方法

移行

- *Visual SourceSafe* からのソースの移行方法
- 他のバージョン管理システムからのソースの移行方法

プロジェクト/ワークスペースの管理

- 複数のチーム プロジェクトとの比較における単一チーム プロジェクトの選択方法
- ソース ツリーの編成方法
- ワークスペース マッピングの定義方法
- コンピュータ上でのワークスペースを使用したコード変更の分離方法

セキュリティ

- 開発者のワークステーションと *TFS* の間のチャネルを保護する方法

シェルビング

- シェルビングを使用した、保留中の作業のバックアップ方法
- シェルビングを使用した、チーム メンバとのコードの共有方法

バージョン管理へのアクセス

- **非 Visual Studio クライアントからのバージョン管理の使用方法**
- **一般的なバージョン管理タスクの自動化方法**
- **オフラインでの作業方法**

非 *Visual Studio* クライアントからのバージョン管理の使用方法

以下のアプローチを使用して、Microsoft® Visual Studio® 2005 Team System (VSTS) Team Foundation Server (TFS) のバージョン管理に他のクライアントからアクセスすることができます。

- Microsoft Source Code Control Interface (MSSCCI) 統合
- サード パーティの統合
- カスタム統合

MSSCCI 統合

以下のクライアントは、MSSCCI プロバイダを使用して TFS バージョン管理を直接操作することができます。

- Microsoft Visual Studio .NET 2003
- Microsoft Visual C++® 6 SP6
- Microsoft Visual Basic® 6 SP6

- Microsoft Visual FoxPro® 9 SP1
- Microsoft Access436 2003 SP2
- Microsoft SQL Server436 Management Studio
- Sparx Systems Enterprise Architect 61
- Sybase PowerBuilder 105
- Toad for SQL Server 2.0

MSSCCI プロバイダは、Visual Studio 2005 では、次のような点で TFS バージョン管理におけるものとは異なった振る舞い方をします。

- チェックアウトでは、**最新バージョンの取得**操作も実行されます。
- チェックアウト時には、排他的なチェックイン ロックが適用されます。
- [**ソース管理で開く**] と [**ソース管理に保存**] は、Microsoft Visual SourceSafe® (VSS) の場合と同じ振る舞い方をします。

MSSCCI プロバイダは、http://www.microsoft.com/downloads/details.aspx?FamilyId=87E1FFBD-A484-4C3A-8776-D560AB1E6198&displaylang=en の Microsoft MSDN® からダウンロードできます。

MSSCCI プロバイダは、Microsoft ではサポートされていません。不明な点があれば、http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=22&SiteID=1 の MSDN フォーラムに問い合わせてみてください。

サード パーティ統合

以下のクライアントは、他のベンダから提供される統合ソリューションを備えています。

- Eclipse
- Linux クライアント
- Apple Macintosh クライアント
- HTML Web クライアント

Eclipse IDE、Linux、または Macintosh のクライアントから Team Foundation のバージョン管理

にアクセスしたい場合は、クライアント アプリケーションの Teamprise スイートをインストールすることを検討してください。これは http://www.teamprise.com/ で入手できます。

Web から Team Foundation のバージョン管理に読み取り専用でアクセスしたい場合は、Team System Web Access をインストールすることを検討してください。これは http://msdn2.microsoft.com/en-us/teamsystem/bb676728.aspx で入手できます。

カスタム統合

その他のクライアントには、現在利用可能な統合ソリューションはありません。TFS バージョン管理にコマンド ラインからアクセスするか、または独自の統合ソリューションをビルドすることができます。

TFS バージョン管理の詳細については、http://msdn2.microsoft.com/ja-jp/library/zthc5x3f(VS.80).aspx の MSDN Web サイトの「チュートリアル : コマンドラインからの Team Foundation ソース管理の操作」を参照してください。

制御スクリプトとコマンド ファイルを使用して、コマンド ラインの使用法を自動化することができます。制御スクリプトとコマンド ファイルを使った作業の詳細については、http://msdn2.microsoft.com/ja-jp/library/1az5ay5c(VS80).aspx の MSDN Web サイトの「Team Foundation ソース管理のスクリプトおよびコマンド ファイル」を参照してください。

参考資料

- MSDN から MSSCCI プロバイダをダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyId=87E1FFBD-A484-4C3A-8776-D560AB1E6198&displaylang=en へアクセスしてください。
- Eclipse IDE、Linux、または Macintosh クライアントから TFS バージョン管理にアクセスする場合、http://www.teamprise.com/ の Teamprise のインストールを検討してみてください。
- インターネットから TFS バージョン管理にアクセスする場合、http://msdn2.microsoft.com/en-us/teamsystem/bb676728.aspx にある Team System

Web Access のインストールを検討してみてください。

- TFS バージョン管理を使った作業の詳細については、http://msdn2.microsoft.com/ja-jp/library/zthc5x3f(VS.80).aspx の「チュートリアル : コマンドラインからの Team Foundation ソース管理の操作」を参照してください。
- 制御スクリプトとコマンド ファイルを使った作業の詳細については、http://msdn2.microsoft.com/ja-jp/library/1az5ay5c(VS.80).aspx の「Team Foundation ソース管理のスクリプトおよびコマンド ファイル」を参照してください。
- TFS バージョン管理の拡張性の詳細については、http://msdn2.microsoft.com/en-us/library/bb187335(VS.80).aspx の「Walkthru: The Version Control Object Model」を参照してください。

一般的なバージョン管理タスクの自動化方法

一般的なバージョン管理タスクを自動化するには、Team Foundation コマンド ライン ツール (tf.exe) を使用します。このツールを使用すれば、ソース管理エクスプローラで実行できるものをすべて実行することができます。それにはたとえば、ソース管理操作 (**追加**、**チェックイン**、**チェックアウト**、**取得**、**ロック**、**ラベル**など)、分岐、シェルビング、ワークスペースの操作、および一般的な管理タスクなどがあげられます。

コマンド ライン ツールを使用する主な理由は、Microsoft Windows® のタスク スケジューラを使用することによって、特定の時刻または特定のイベントを契機として実行される反復操作やスケジューリング操作などが自動化されるからです。以下のコマンドも、コマンド ラインからのみ使用することができます。

- ユーザーのワークスペースの削除
- 別のユーザーのチェックアウトを元に戻す
- 別のユーザーのロックの解除
- ラベルの有効範囲の定義
- 基点のないマージの実行

該当するパスとその他の環境変数を確実にセットアップするには、Visual Studio 2005 のコマンド プ

ロンプト ウィンドウからコマンド ライン ツールを実行するか、または、通常は *DriveLetter:*¥Program Files¥Microsoft Visual Studio 8¥Common7¥Tools に置かれている Vsvars32 バッチ ファイルを実行します。

Tf.exe は、TFS クライアントの一部としてインストールされて、既定では、次のようなフォルダ内に置かれます。
C:¥Program Files¥Microsoft Visual Studio 8¥Common 7¥IDE.

コマンド ライン ツールを実行するには、**/s** スイッチを付けてサーバー名を指定する必要があります。以下のコマンドは、YourTFSServer という名前のサーバー上のソース管理でファイルを表示する方法を示しています。

```
tf.exe dir /s:YourTFSServer
```

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/zthc5x3f(VS.80).aspx の MSDN Web サイトの「チュートリアル : コマンドラインからの Team Foundation ソース管理の操作」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/cc31bk2e(VS.80).aspx の「Team Foundation ソース管理のコマンド ライン リファレンス」を参照してください。

オフラインでの作業方法

オフライン作業は、TFS バージョン管理ではもともとサポートされていません。

オフラインで作業するには、次のような厳密なワークフローをたどる必要があります。

1. **読み取り専用フラグを手動で除去します。**既定では、まだチェックアウトされていないワークスペース内のすべてのファイルは、読み取り専用のマークを付けられています。サーバー接続なしで作業を行う場合、読み取り専用フラグをファイルから手動で除去してから、そのファイルの編集または削除を行う必要があります。そのためには、Windows エクスプローラでそのファイルを右クリ

ックし、[**プロパティ**] をクリックし、[**読み取り専用**] チェック ボックスをクリアしてから、[**OK**] をクリックします。別の方法として、DOS コマンド **attrib -r** を使用することもできます。

2. **ファイルを編集します。**読み取り専用フラグを除去したファイルをすべて編集できます。
3. **ファイルを追加または削除します。**読み取り専用フラグを除去したファイルを追加または削除することができます。ファイルの名前を変更しないでください。なぜなら、**TFPT オンライン** ツールは、**削除**と**追加**がペアになった操作と、**名前変更**を区別できないからです。

   注意: 削除対象が探索されるようにするには、**TFPT online** コマンドに対するオプションを指定する必要があります。なぜなら、これは、時間のかかる操作だからです。
4. **TFPT オンライン コマンドを実行します。**オンラインに戻ってから、コマンド ラインに **TFPT online** と入力して、**TFPT オンライン** コマンドを実行します。このコマンドは、ワークスペースをスキャンして書き込み可能なファイルを見つけ出し、サーバー上でどのような変更を保留中にすればよいかを判別します。いずれかのファイルが削除済みの場合、**/delete** スイッチを使用します。これにより、ローカル ワークスペースでも削除済みのファイルをスキャンするようツールに指示が与えられます。その後ツールは、オンライン ウィンドウを表示するので、そこで、どの変更をワークスペース内で保留中にするかを選択できます。

重要事項: オフライン中はどのファイルも名前を変更しないでください。

参考資料

- MSDN から TFPT ツールをダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyID=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。
- Visual Studio Team Foundation Power Tool の詳細については、http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx の「Power Toy: tfptexe」を参照してください。

管理

- **プロジェクトに新規開発者を追加する方法**

- **プロジェクトから離脱した開発者の削除方法**
- **ソース ツリー内でのアクセス許可の付与方法**
- **別のサーバーへの Team Foundation バージョン管理の移動方法**

プロジェクトへの新しい開発者の追加方法

プロジェクトに新しい開発者を追加するには、チーム プロジェクトおよび関連 Microsoft Office SharePoint® プロジェクト サイトにアクセスするのに適したアクセス許可をその開発者に認可します。開発者がレポートを表示できるようにするには、SQL Server Reporting Services へのその開発者のアカウント アクセス許可を認可します。

チーム プロジェクトにアクセス権を認可するには

1. Team Foundation Administrators アプリケーション グループのメンバであるアカウントを使って、Visual Studio にログオンします。

2. 必要なプロジェクトをチーム エクスプローラに追加します (まだリスト表示されていない場合)。
3. チーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] をポイントしてから、[**グループ メンバシップ**] をクリックします。
4. [**Project¥Contributors**] を選択し、[**プロパティ**] をクリックしてから、新規開発者のアカウントをこのグループに追加します。

注意: 貢献者グループのメンバシップには、開発者にとって必要な一連の標準的なアクセス許可が備わっていることに注意してください。そのようなアクセス許可によって、たとえばチーム プロジェクト内のアイテムの追加、変更、および削除を行うとともに、ビルドを実行することができます。

SharePoint プロジェクト サイトへのアクセス権を認可するには

1. SharePoint 管理者サイト グループのメンバであるアカウントを使用して、チーム プロジェクト サイトにアクセスします。既定では、プロジェクト用のYourProject というプロジェクト サイト

は、http://server/sites/YourProject/default.aspx に置かれています。

2. [**サイトの設定**] をクリックします。
3. **管理**タイトルの下の [**ユーザーの管理**] をクリックします。
4. [**ユーザーの追加**] をクリックします。
5. 新規開発者のアカウント名を domain¥useraccount の形式で入力し、[**投稿者**] を選択してから [**次へ**] をクリックします。
6. 開発者の電子メール アドレスを入力してから、任意選択で、サイトへのアクセスに備えて歓迎メッセージを入力します。
7. [**完了**] をクリックします。

注意: 貢献者グループのメンバシップには、開発者にとって必要な一連の標準的なアクセス許可が備わっていることに注意してください。そのようなアクセス許可によって、たとえばチーム プロジェクト内のアイテムの追加、変更、および削除を行うとともに、ビルドを実行することができます。特定の Visual Studio ソリューションや、チーム プロジェクト内の特定のフォルダへのアクセスを制限する必要がある場合、フォルダ レベルまたはファイル レベルでアクセス許可を割り当てることもできます。

SQL Server Reporting Services にアクセス権を認可するには

1. 管理者アカウントを使用して、SQL Server Reporting Services の管理 Web サイトにログオンします。このサイトは、http://server/reports に置かれています。
2. チーム プロジェクト名をクリックします。
3. [**プロパティ**] タブをクリックします。
4. [**セキュリティ**] タブをクリックします。
5. [**新しいロールの割り当て**] をクリックします。
6. 開発者の Windows アカウント名を入力してから、[**閲覧者**] を選択し、次に [**OK**] をクリックします。

注意: 開発者は、閲覧者グループのメンバシップを使用して、レポートの表示やサブスクライブを行えることに注意してください。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253077(VS.80).aspx の「Team Foundation Server の既定のグループ、アクセス許可、およびロール」を参照してください。
- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181761(VS.80).aspx の「ソース管理のセキュリティ権限およびアクセス許可」を参照してください。

プロジェクトから離脱した開発者の削除方法

開発者がプロジェクトから離れた場合、必ずその開発者のワークスペースを削除してください。この操作によって、プロジェクトのセキュリティが確保されるだけでなく、その開発者の保留中の変更がすべて除去され、その開発者が設定していたすべてのロックが元に戻されます。

注意: チーム プロジェクトで、排他ロックがオンになっていた場合、変更を元に戻さない限り、ロックを元に戻すことはできません。

その開発者がどのファイルをロックしていたかを突き止めるには、次のようなコマンドを実行します。

tf workspaces /owner:domain¥devuser /computer:* /server:servername

ワークスペースを削除してロックを除去するには、次のようなコマンドを実行します。

tf workspace /delete workspacename;domain¥devuser /s:servername

次に、その開発者のアカウントをセキュリティ グループから削除します。そのためには、次の 3 つの場所で変更を行います。

- **TFS チーム プロジェクト–** Team Foundation Administrators アプリケーション グループのメンバであるアカウントを使って、Visual Studio にログオンします。[チーム エクスプローラ] でプロジェクトを右クリックし、[**チーム プロジェクトの設定**] をポイントしてから、[**グループ メ**

ンバシップ] をクリックして、開発者のアカウントを関連グループ (通常は Contributor) から除去します。

- **SharePoint チーム プロジェクト サイト– 管理者アカウントを使用して、** http://server/sites/*yourprojectname*/default.aspx のチーム サイトにログオンします。[**サイトの設定**]、[**ユーザーの管理**] をクリックしてから、開発者のアカウントを削除します。
- **SQL Server Reporting Services –** 管理者アカウントを使用して、SQL Server Reporting Services の管理 Web サイトにログオンします。このサイトは、http://server/reports に置かれています。チーム プロジェクト名をクリックし、[**プロパティ**] タブをクリックし、[**セキュリティ**] タブをクリックしてから、開発者のアカウントを削除します。

参考資料

- 開発者がプロジェクトを離れた後のクリーンアップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194958(VS.80).aspx の「方法 : チームを離れるメンバのファイルをクリーンアップする」を参照してください。
- **Workspace** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/y901w7se(VS.80).aspx の「Workspace コマンド」を参照してください。
- 保留中の変更の検出方法の詳細については、http://blogs.vertigosoftware.com/teamsystem/archive/2006/07/24/3125.aspx の「How do I tell who has files checked out or locked?」を参照してください。

ソース ツリー内でのアクセス許可の付与方法

ソース ツリー内でファイルまたはフォルダに対するアクセス許可を設定することができます。それには、ソース管理エクスプローラでそのファイルまたはフォルダを右クリックし、[**プロパティ**] をクリックし、[**セキュリティ**] タブをクリックし、アクセス許可を変更したいユーザーまたはグループを選択してから、アクセス許可を編集します。また、ソース管理の tf.exe コマンド ライン ユーティリティで **Permission** スイッチを使用して、アクセス許可を設定することもできます。

特定のフォルダおよびファイルに対してソース管理のアクセス許可を適用できますが、既定では、ソース ツリー内のアクセス許可は、プロジェクト フォルダに対して適用されるアクセス許可から継承されます。開発者が Project¥Contributors アプリケーション グループのメンバである場合、その開発者は、ソース ファイルの読み取り、チェックアウト、チェックイン、ラベル付け、およびロックを行うことができます。たとえば、チーム プロジェクト内の特定のソース ファイルのみを開発者が操作できるようにしたい場合のように、チーム プロジェクト内のフォルダまたはソース ファイルのサブセットへの制限付きアクセス権を設ける必要がある場合、フォルダ レベルまたはファイル レベルでアクセス許可を設定することができます。

参考資料

- アクセス許可の認可の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253077(VS.80).aspx の MSDN Web サイトの「Team Foundation Server の既定のグループ、アクセス許可、およびロール」を参照してください。
- アクセス許可の付与の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181761(VS.80).aspx の「ソース管理のセキュリティ権限およびアクセス許可」を参照してください。
- **tf permission** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/0dsd05ft(VS.80).aspx の「Permission コマンド」を参照してください。

別のサーバーへの *Team Foundation* バージョン管理の移動方法

Team Foundation Server は、ある場所からある場所へのサーバーのコピーも、ミラーリングもサポートしません。サーバー全体のバックアップと復元、別のドメインへの既存のハードウェアの移動、デュアル サーバー配置へのアップグレードを行うことができます。たとえば、一部のプロジェクトをサーバーから移動する一方で、その他のプロジェクトはそのまま残すといった、部分的な移動は実行できません。

Team Foundation Server は、以下の 3 種類の移動方式をサポートします。

- **バックアップ復元** – Team Foundation Server を新しいマシンに移動する必要がある場合、この移動方式をとります。具体的なステップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms404869(VS.80).aspx の「方法 : あるハードウェア構成から別のハードウェア構成へ Team Foundation Server を移動する」を参照してください。
- **環境** – Team Foundation Server を新しいドメインまたはワークグループに移動する必要がある場合、この移動方式をとります。具体的なステップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms404883(VS.80).aspx の「方法 : ある環境から別の環境へ Team Foundation Server を移動する」を参照してください。
- **シングル サーバーからデュアル サーバーへ** – シングル サーバー配置からデュアル サーバー配置に移動する必要がある場合、この移動方式をとります。具体的なステップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms404854(VS.80).aspx の「方法 : シングルサーバー配置からデュアルサーバー配置に移行する」を参照してください。

Team Foundation Server を移動するときは、以下の考慮事項に配慮してください。

- TFS アプリケーション層サーバー名を変更すると、すべての TFS クライアントが新しいサーバー名に接続しなければならなくなります。
- サーバー名を変更した場合、クエリ バインドの Microsoft Office ドキュメントはすべて機能しなくなります。ドキュメントは、その作成の対象であったサーバーにバインドされます。それには、プロジェクトの作成時に**ドキュメント**ノードで自動的に作成されるクエリ バインドのすべての Microsoft Office ドキュメントも含まれます。
- サーバー名を変更した場合、ドキュメントへの組み込みリンクは、不明のサーバー名を指し示します。
- オリジナルの TFS 上にはローカル アカウントが存在します。そのようなアカウントを、移動後の TFS 上でローカル アカウントとして再作成するか、または移動後の Team Foundation Server の新しいドメイン内のドメイン アカウントとして再作成するかを決定する必要があります。
- オリジナルの TFS 上で存在するドメイン アカウントを、オリジナル ドメインを信頼しないドメインに TFS を移動する場合。そのようなアカウントを、移動後の TFS 上でローカル アカウントとして再作成するか、または移動後の Team Foundation Server の新しいドメイン内のドメイン

アカウントとして再作成するかを決定する必要があります。

移動後にサーバーをテストして、移行の際に何も重大な損壊がなかったことを確認してください。テストでは、次のような点をチェックする必要があります。

- すべての資産が正しく移動されたか。ソース ツリー、作業項目、およびチーム ページをよく調べて、すべて揃っていることを確認します。
- ユーザー アカウントはこれまでどおりの効力があるか。数種類のアカウントを使ってログインを試みて、有効であることを確認します。

参考資料

- TFS の移動の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms404879(VS.80).aspx の「Team Foundation Server の移動」を参照してください。

分岐/ラベル/マージ

- **ラベルの使用方法**
- **分岐方法**
- **分岐構造の計画方法**
- **リリースのサポートのための分岐の使用方法**
- **前のリリースの保守のための分岐の使用方法**
- **開発およびビルドのプロセスの安定化のための分岐の使用方法**
- **機能の開発の安定化のための分岐の使用方法**
- **複数のチームにまたがる開発の安定化のための分岐の使用方法**
- **外部依存関係の分離のための分岐の使用方法**
- **古いリリースの撤去方法**
- **マージの実行方法**
- **基点のないマージの実行方法**
- **マージの競合の解決方法**

ラベルの使用方法

将来の操作に備えて、一連のファイルとフォルダを 1 つのグループにまとめるときに、ラベルを使用します。ラベルを使用して、ファイルの分岐、マージ、差分の確認、または取得を行うことができます。上記の操作の 1 つを後日実行するときに戻るための目印のマーカーが、ラベルによって提供されます。

ファイルまたはフォルダにラベルを適用するには

1. [ソース管理エクスプローラ] のファイルまたはフォルダを右クリックしてから、[**ラベルの適用**] をクリックします。
2. [**項目のバージョンの選択**] ボックスで、ファイルまたはフォルダの選択内容を変更し、ラベルを付けるファイルまたはフォルダのバージョンを選択し、次に [**OK**] をクリックしてラベルを適用します。

ラベルを適用するときは、以下の点に留意してください。

- ラベルは、一度にファイルまたはフォルダの 1 つのバージョンにのみ適用できるマーカーです。
- ファイルまたはフォルダの 1 つのバージョンに対して、複数のラベルを適用することができます。
- ソース管理エクスプローラを使用してラベルを適用すると、その有効範囲は、その作成場所であるチーム プロジェクトのルート フォルダにまで自動的に達します。1 つの有効範囲内で同じ名前の付いたラベルを 2 つ作成することはできません。
- ラベルは、バージョンのないオブジェクトであるため、関連した履歴はありません。
- ラベルは瞬時に適用されるので、チェックインの必要はありません。
- チーム ビルドは、作成する各ビルドに関連付けられた一連のファイルにラベルを自動的に割り当てます。
- 削除済みの項目にはラベルは適用されません。つまり、削除済みのファイルはラベル別のマージでは検出されないということです。

既存のラベルを見つけ出すには

1. **[ファイル]** メニューの **[ソース管理]** 、**[ラベル]** をポイントし、次に **[ラベルの検索]** をクリックしてラベルへ移動します。
2. ラベルを検索後、[ラベルの検索]　ボックス内で、そのラベルを変更または削除できます。

参考資料

- ラベルの使用の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181439(VS.80).aspx の「ラベルの操作」を参照してください。
- ラベルの適用の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181440(VS.80).aspx の「方法 ： ラベルを適用する」を参照してください。

分岐方法

分岐を作成するには、[ソース管理エクスプローラ] を使用するか、またはコマンド ラインから **tf branch** コマンドを使用します。

[ソース管理エクスプローラ] から分岐するには、プロジェクト ソースを格納している最上位フォルダを右クリックし、[**分岐**] をクリックしてから、たとえば **MyProject_Release1.0_Branch** といった、その分岐の目的を示すフォルダ名を使ってターゲットのフォルダ場所を指定します。

コマンド ラインから分岐するには、Visual Studio 2005 コマンド プロンプトから **tf branch** コマンドを使用します。以下に例を示します。

tf branch C:¥MyProject $/MyProject_Release1.0_Branch

分岐を使用するのは、並列開発を行っているときに、分離する必要がある場合のみです。分岐を使用すると、最終的に変更内容をメイン分岐にマージする必要があり、しかもそのマージによってオーバーヘッドが生じるとともに、競合を管理しなければならなくなるので、分岐によって実現されるファイルの

分離が必要でない限り、分岐を行わないでください。ビルドにラベルを付けておいて、必要があれば後で分岐することができます。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

分岐構造の計画方法

分岐とは、分離を可能にするためのメカニズムのことです。これを使用して、複数の開発者が互いに分離した状態で同じファイルを作業することができます。分岐のアプローチを計画するには、一般的な分岐のシナリオを評価したうえで、チームの規模、構成、リリースのスケジュール、およびビルドの安定性の要件に基づいて、戦略を選びます。

一般的な分岐のシナリオをいくつかを以下に示してあります。

- **リリース** – リリースしようとしているコードの安定化のための分岐。リリースの前に分岐することによって、メイン分岐をロックダウンしなくて済むようになります。
- **メンテナンス** – これまでにリリースされたビルドを保守するための分岐。
- **機能** – プロジェクトの残りの部分の不安定化につながる可能性のある機能開発を分離するための分岐。
- **チーム** – 複数のサブチームを分離するための分岐。これにより、変更点に煩わされずに作業する

ことができ、ただ 1 つの目標を目指して作業することができます。これは、機能の分岐によく似ています。

開発チームにとって必要にならない限り、分岐を行わないでください。分岐を行うと、余計なソース ツリーのメンテナンスやマージ タスクが生じます。短期サイクルに携わっている (たとえば基幹業務 [LOB] アプリケーション) たいていの開発チームにとって、分岐が必要になることは決してありません。長期サイクルで作業している (たとえば、独立系ソフトウェア ベンダ [ISV] のパッケージ アプリケーション) に携わっている開発チームは、開発プロセスの一環として分岐を用いる場合があると考えられます。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

リリースのサポートのための分岐の使用方法

リリース前のビルドを安定化する準備ができたら、リリース分岐を使用します。

リリース分岐を作成した後、その分岐構造がどのようになるかの例を以下に示します。

- **Main** – 統合分岐

    - **Source**
    - **Other Asset Folders**
- **Releases** – リリース分岐のコンテナ
    - **Release 1** – リリース分岐
        - **Source**

リリース分岐を処理する際は、以下の推奨事項に留意してください。

- **分岐が必要な場合** – リリースの準備ができたら、すべてをメイン分岐に統合してから、リリース前のアプリケーションの安定化に使用できるリリース分岐を作成します。
- **分岐が不要な場合** – リリースごとに 1 つの TFS プロジェクトを使用している場合、現行プロジェクトで分岐するのではなく、新しいプロジェクトを使用して開発を続行することができます。
- **分岐に対するアクセス許可**:
    - **リリース前** – すべての開発者に読み取り/書き込みのアクセス許可を割り当てます。
    - **リリース後** – ホットフィックスを処理する開発者に読み取り/書き込みアクセス許可を割り当てて、他の開発者全員に読み取り専用アクセス許可を割り当てます。
- **分岐におけるビルドの頻度** –要求に基づいたビルド。
- **分岐におけるテストの焦点** – リリースの承認。

リリース前のビルドの安定化に必要なターゲットのフィックスおよび変更では、リリース分岐を使用する必要があります。アプリケーションの次のバージョンの開発は、開発分岐またはメイン (統合) 分岐内で並列して実行することができます。これにより、リリース前に行われた安定化のための変更を有効に利用することができます。最終リリース ビルドの作成が完了したら、リリース分岐の変更を開発分岐およびメイン (統合) 分岐にマージして戻すことができます。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx

の「方法 ： ファイルとフォルダを分岐する」を参照してください。

- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

前のリリースの保守のための分岐の使用方法

メンテナンス分岐を使用して、これまでにリリースしたビルドをサポートします。これは、リリースの場合の分岐によく似ていますが、リリースをサポートするために分岐は長期間保持されます。

アプリケーションをリリースした後、そのリリースをサポートするために分岐を保持すると、その分岐構造がどのようになるかの例を以下に示します。

- **Main** – 統合分岐
  - **Source**
  - **Other Asset Folders**
- **Releases** – リリース分岐のコンテナ
  - **Release 1** – メンテナンス分岐
    - **Source**
    - **Other Asset Folders**

メンテナンス分岐を処理する際は、以下の推奨事項に留意してください。

- **分岐が必要な場合** – リリースを行った後、リリース フォルダ内の分岐でリリースをサポートします。
- **分岐が不要な場合** – リリースを保守する必要がない場合、ラベルを使用して古いリリース済みビルドにマークをつけ、メイン分岐で作業を続行することができます。
- **分岐に対するアクセス許可** – ホットフィックスを処理する開発者に読み取り/書き込みアクセス許可を割り当てて、他の開発者全員に読み取り専用アクセス許可を割り当てます。

- **分岐におけるビルドの頻度** – 要求に基づいたビルド。
- **分岐におけるテストの焦点** – リリースの承認。

古いバージョンのアプリケーションをサポートするには、メンテナンス分岐を使用する必要があります。選択に応じて、そのような変更をメイン (統合) 分岐にマージするか、またはメンテナンス分岐に置いたままにすることができます。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

開発およびビルドのプロセスの安定化のための分岐の使用方法

アクティブな開発では開発分岐を使用し、統合ビルドではメイン分岐を使用して、ビルドが壊れないようにします。

開発分岐を作成した後、その分岐構造がどのようになるかの例を以下に示します。

- **Development** – 開発分岐
  - **Source**
- **Main** – 統合分岐

- **Source**
- **Other Asset Folders**

開発分岐を処理する際は、以下の推奨事項に留意してください。

- **分岐が必要な場合** – デイリービルドを作成していて、ビルドの安定化と統合に問題がある場合、メインと開発の両方の分岐を作成し、デイリービルドをより明確に予測できるようにします。また、チェックイン品質の向上のために、さらに厳格なチェックイン ポリシーを検討してみることもできます。
- **分岐が不要な場合** – 単に継続的な統合 (CI) ビルドを作成しているだけの場合や、デイリービルドが予想どおりの安定性を既に確保している場合、わざわざ統合分岐を作成して余分なオーバーヘッドを生じる必要はありません。
- **分岐に対するアクセス許可**:
  - メイン分岐は、マージおよび統合を担当する開発者にとっては読み取り/書き込み可能でなければならず、他のすべての開発者にとっては読み取り専用でなければなりません。
  - 開発分岐は、全員にとって読み取り/書き込み可能でなければなりません。
- **分岐におけるビルドの頻度**:
  - メイン分岐上のデイリービルド
  - 開発分岐上の継続的な統合ビルド
- **分岐に対するテストの焦点**
  - メイン分岐に対する統合、パフォーマンス、およびセキュリティのテストを実行します。
  - 開発分岐に対する機能および高速フィードバックのテストを実行します。

開発分岐にチェックインされた統合変更用のステージング基盤として、メイン分岐を使用します。開発分岐内で、すべてのアクティブな開発を実行し、非変更点をメイン分岐に統合します。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。

- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

機能の開発の安定化のための分岐の使用方法

複数の機能を対象に、変更点の統合および安定化を改善するには、機能分岐を使用します。

機能分岐を作成した後、その分岐構造がどのようになるかの例を以下に示します。

- **Development –** 機能分岐のコンテナ
  - **Feature A –** 機能分岐
    - **Source**
  - **Feature B –** 機能分岐
    - **Source**
  - **Feature C –** 機能分岐
    - **Source**
- **Main –** 統合分岐
  - **Source**
  - **Other Asset Folders**

機能分岐を処理する際は、以下の推奨事項に留意してください。

- **分岐が必要な場合** – 機能チームでソース ファイルの重複が生じることはよくあり、それによって、ビルドの破壊やチェックインの競合の可能性が高まります。このような問題が発生した場合、機能を互いに分離するために、各機能ごとに機能分岐の作成を検討してみてください。これは、小規模のチームの場合はメイン分岐から作成することができ、大規模のチームの場合はチーム分岐から作

成することができます。

- **分岐が不要な場合** – 単に CI ビルドを作成しているだけの場合や、デイリービルドが予想どおりの安定性を既に確保している場合、わざわざ機能分岐を作成して余分なオーバーヘッドを生じる必要はありません。
- **分岐でのアクセス許可** – 機能分岐上の開発者に読み取り/書き込みアクセス許可を割り当てて、他の開発者全員に読み取り専用アクセス許可を割り当てます。

- **分岐に対するビルドの頻度** – 分岐に対して、継続的な統合ビルドを実行します。
- **分岐におけるテストの焦点** – 機能テストおよび高速フィードバックのテストを実行します。

どの機能も並列して開発できるようにするには、機能分岐を使用します。機能分岐内で、すべてのアクティブな開発を実行し、次にコードをメイン分岐に統合します。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

複数のチームにまたがる開発の安定化のための分岐の使用方法

複数のチームを対象に、変更点の統合および安定化を改善するには、チーム分岐を使用します。

チーム分岐を作成した後、その分岐構造がどのようになるかの例を以下に示します。

- **Development –** チーム分岐のコンテナ
  - **Team 1 -** チーム分岐
    - **Source**
  - **Team 2 -** チーム分岐
    - **Source**
- **Main –** 統合分岐
  - **Source**
  - **Other Asset Folders**

チーム分岐を処理する際は、以下の推奨事項に留意してください。

- **分岐が必要な場合** – 開発チームにおいてファイルやコンポーネントの重複が生じた場合、チーム分岐を使用して、各チームの作業を互いに分離します。また、選択しだいで、チーム分岐の下に機能分岐を作成することもできます。
- **分岐が不要な場合** – ソース ツリーがコンポーネントによって編成されていて、インターフェイスの変更点や、複数のチームにまたがるチェックイン競合は生じないことが確実と思われる場合、チーム分岐は必要ありません。
- **分岐でのアクセス許可** – チームの開発者に読み取り/書き込みアクセス許可を割り当てて、他の開発者全員に読み取り専用アクセス許可を割り当てます。
- **分岐に対するビルドの頻度** – 分岐に対して、継続的な統合ビルドを実行します。
- **分岐におけるテストの焦点** – 機能および高速フィードバックのテストを実行します。

複数のチームが開発タスクを並列して実行できるようにするには、チーム分岐を使用する必要があります。これが便利なのは、複数のチームを、他のチームで行われた変更点から分離する場合や、複数のチームが単一の目標に向けて作業できるようにしたい場合です。アクティブな開発はすべて、チーム分岐内で行われてから、メイン分岐に統合されなければなりません。また、各チーム分岐の下に機能分岐を作成することもできます。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

外部依存関係の分離のための分岐の使用方法

外部依存関係が原因の変更点の統合および安定化を改善するには、外部分岐を使用します。

外部分岐を作成した後、その分岐構造がどのようになるかの例を以下に示します。

- **External –** 外部分岐
  - **Source**
- **Main –** 統合分岐
  - **Source**
  - **Other Asset Folders**

外部分岐を処理する際は、以下の推奨事項に留意してください。

- **分岐が必要な場合** – プロジェクト内の変更点の原因となる可能性のある外部依存関係が存在する場合。依存関係に起因して、コードに影響与えることになるインターフェイスの変更やロジックのかなりの変更を行う場合、そのような変更を分離するための外部分岐を作成します。
- **分岐が不要な場合** – 外部依存関係が安定している場合や、変更点を伴わないと確信できる場合。
- **分岐におけるアクセス許可** – 外部依存関係の統合に携わる開発者には読み取り/書き込みの許可

を与え、他の全員に読み取り専用許可を与えます。

- **分岐に対するビルドの頻度** – オンデマンドでビルドを実行します。
- **分岐におけるテストの焦点** – 統合テスト。

外部分岐は、隔絶した場所で使用して、外部依存関係における変更をテストしてから、その変更をメイン分岐に統合します。このようにすれば、ヘッダーやライブラリなどの外部依存関係に起因した変更点から開発チームを分離するのに便利です。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

古いリリースの撤去方法

リリース フォルダ内のリリース分岐が、古くなってもうサポートする必要がなくなったら、古いリリースを保管する Safe Keeping フォルダを作成します。Safe Keeping フォルダを作成した後、分岐構造がどのようになるかの例を以下に示します。

- **Main** – 統合分岐
  - **Source**
  - **Other Asset Folders**

- **Releases** – リリース分岐のコンテナ
  - **Release 2** – メンテナンス分岐
    - **Source**
    - **Other Asset Folders**
- **Safe Keeping -** Safe Keeping 分岐のコンテナ。
  - **Release 1 -** Safe keeping 分岐
    - **Source**
    - **Other Asset Folders**

リリース フォルダから Safe Keeping フォルダに分岐を移動する目的は、リリース フォルダ内の煩雑さを軽減し、古いリリースを削除しなくて済むようにすることにあります。新しい分岐を設けるのではなく、古い分岐を新しいフォルダに移動するということです。

参考資料

- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 での分岐およびマージの実行方法の追加情報は、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

マージの実行方法

分岐から分岐に変更を統合するには、**マージ** コマンドを使用します。マージを実行する場合、ソース管理エクスプローラのマージ機能を使用できます。あるいは、**tf merge** コマンド ラインを使用して

もかまいません。変更セット、ラベル、日付、またはバージョン別のマージを行うことができます。マージを開始するには、ソース管理エクスプローラ内の分岐を右クリックしてから、[**マージ**] をクリックします。[ソース管理マージ ウィザード] では、マージの対象のターゲット分岐を選択することができます。

分岐構造によっては、分岐階層を上下に移動したり、階層をまたがったりしながら変更をマージする必要があるかもしれません。階層をまたがったマージを行う場合、基点のないマージを実行することになるので、**tf merge** コマンド ラインを使用する必要があります。なぜなら、これを Visual Studio で実行することはできないからです。基点のないマージでは、分岐/マージの関係を持たないファイルおよびフォルダをマージすることができます。基点のないマージを実行した後は、マージ関係が確立されるので、その後は基点のないマージではなくなります。その場合でも、やはりコマンド ラインからマージを実行する必要があります。ただし、マージ競合の数は削減されます。

マージを実行する場合は、以下の点に留意してください。

- 階層に沿ったマージ (親から子または子から親への) を実行すれば、階層をまたがるマージよりも競合数は少なくなります。
- 分岐階層では、親分岐と子分岐がベースになります。これは、ソース管理エクスプローラで見られる物理構造とは異なっているかもしれません。以下に例を示します。
    - **物理構造**:
        - **Development –** 開発分岐
        - **Main –** 統合分岐
        - **Releases** – リリース分岐のコンテナ
            - **Release 1** – リリース分岐
    - **論理構造**:
        - **Main**
            - **Development**
            - **Release 1**

参考資料

- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181427(VS.80).aspx の「マージについての情報」を参照してください。
- マージの実行方法のチュートリアルは、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。

基点のないマージの実行方法

基点のないマージを実行するには、Visual Studio 2005 コマンド プロンプトから **tf merge /baseless** コマンドを使用します。

たとえば、以下のコマンド ラインは、ソース分岐からターゲット分岐への基点のないマージを実行します。指定した分岐パス内のすべてのファイルとフォルダの再帰的マージを実行するために、**/recursive** スイッチを使用しています。

tf merge /baseless <<ソース パス>> <<ターゲット パス>> /recursive

例

tf merge /baseless c:¥data¥proj1 c:¥data proj2 /recursive

それぞれ相手からは直接分岐していない 2 つのアイテムをマージするプロセスを、基点のないマージと呼びます。たとえば、互いに兄弟である 2 つのリリース分岐の間で変更内容をマージしたいけれども、親分岐にまでさかのぼってマージしたくない場合があります。基点のないマージは、コマンド ラインからのみ作動します。これを Visual Studio から実行することはできません。

基点のないマージの実行時には、分岐内のファイルの関係に関する情報を TFS はまったく持っていません。たとえば、ファイルの名前を変更した場合、それは分岐内の削除済みのファイルおよび新規のファイルと見なされます。こうした理由で、通常のマージを実行する場合よりも、手動で競合を解決する回数が増えます。ただし、そのような競合の解決は 1 回実行するだけで済みます。基点のないマージ

の実行が完了すれば、TFS には履歴が記録され、フォルダとファイルの間の関係が確立されます。

基点のないマージを作成できるのは、コマンド ラインからのみです。最初の基点のないマージの後は、分岐の間に関係が確立されますが、その後のマージはやはりコマンド ラインから作成する必要があります。

参考資料

- **マージ** コマンドの構文の詳細については、http://msdn2.microsoft.com/ja-jp/library/bd6dxhfy(VS.80).aspx の「Merge コマンド」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181427(VS.80).aspx の「マージについての情報」を参照してください。
- マージの実行方法のチュートリアルは、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。

マージの競合の解決方法

マージの競合を解決するには、Visual Studio のマージ ツールを使用します。マージ中に競合が検出された場合、自動または手動で競合を解決することができます。手動での競合の解決を選択した場合、マージ ツールで、マージ元の変更の保持、マージ先の変更の保持、または競合の解決を行うことができます。

それぞれの分岐間の変更のマージ、ワークスペースへのファイルの取り込み、新しいバージョンのファイルのチェックインのときに、競合を解決する必要が生じる可能性があります。競合には、以下の 3 種類があります。

- **バージョン** – 何本にも分かれたパスに沿ってファイルが進展してきた場合。これは、ファイルの編集、名前変更、削除、または削除の取り消しの操作に起因すると考えられます。
- **ファイル名の衝突** – 複数のアイテムが同じパスの占有を試みた場合。

- **ローカル上書き** – ローカルの編集可能ファイルの上書きを試みた場合に、**get** 操作でのみ発生します。

競合のほとんどは、自動的に解決することができます。バージョン競合は、手動のマージ操作へとつながる唯一の種類の競合です。手動マージが最もよく使用されるのは、次のようなシナリオの場合です。

- 1 つのファイルが両方の分岐で編集されて、同じコード行に変更が加えられた場合。
- 基点のないマージが実行された場合に、分岐ファイルどうしの関係が TFS によって認識されていないとき。

マージ ツールでは、各競合の詳細が示されるので、マージ後の最終ファイルにどの変更を保持するかを選択することができます。マージ元の変更の保持、マージ先の変更の保持、またはこの両方の変更の統合を選択できますが、ファイルに直接入力することによって、最終バージョンを手動で変更することもできます。

ファイルにおけるすべての競合を解決した後、ターゲット分岐にチェックインする保留中の変更として、最終バージョンを保存します。

マージは慎重に行ってください。なぜなら、間違いが起きることはよくあり、それが原因でビルドが不安定になることがあるからです。マージを完了したら、その結果のソースをコンパイルし、重大な破壊がないことを確かめる単体テストを実行します。

参考資料

- 競合の解決の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181433(VS.80).aspx の「方法 ： 競合を解決する」を参照してください。

ビルド

- **TFS を使用した継続的な統合ビルドの実行方法**

TFS を使用した継続的な統合ビルドの実行方法

ファイルのチェックインのたびに継続的な統合を実行して、プロジェクトを再ビルドするためには、Web サービスを使用してビルド プロセスをトリガする必要があります。誰がファイルをチェックインしても、そのつどビルドがトリガされるようにするには、Web サービスでチェックイン イベントをサブスクライブします。ビルドをトリガするには、どのような構成をビルドするか、どのようなビルド後のステップおよびテストを実行するか、どのようなドロップ場所を使用するか、などを決定するのに適したビルドの種類を使用する必要もあります。

参考資料

- ビルド プロセスのトリガ用として Microsoft から提供されている Web サービスをダウンロードするには、http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi にアクセスしてください。

チェックインおよびチェックイン ポリシー

- **ソース管理変更セットの処理方法**
- **チェックイン前にコーディング基準を強制する方法**
- **チェックイン ポリシーのオーバーライド方法**
- **チェックインを元に戻す方法**
- **競合の解決方法**
- **競合の回避方法**
- **カスタム チェックイン ポリシーの作成方法**

ソース管理変更セットの処理方法

変更セット は、特定のチェックインに関連した一連の変更をグループにまとめたものです。変更セッ

トに対して実行する必要のある一般的な操作は次のとおりです。

- 作業項目に関連した変更セットのチェックイン
- 変更セットへのラベル付け
- 変更セットの詳細事項の表示
- 変更セットの詳細事項の変更
- 変更セットのロール バック

作業項目に関連した変更セットをチェックインするには

1. Visual Studio の [**表示**] メニューで [**その他のウィンドウ**] をポイントしてから、[**保留中の変更**] をクリックします。
2. 変更の特性を反映するのにふさわしいコメントを入力します。
3. [**作業項目**] アイコンをクリックして、関連した作業項目のリストを表示します。
4. 関連した作業項目を選択してから、チェックイン アクション [**関連付け**] または [**解決**] (チェックインが作業項目の解決を意味する場合) を指定して、その作業項目を更新します。
5. [**チェックイン**] をクリックして、変更セットをソース管理サーバーにチェックインします。

変更セットにラベルを付けるには

1. Visual Studio で、[ソース管理エクスプローラ] の [チーム プロジェクト] フォルダを右クリックしてから、[**ラベルの適用**] をクリックします。
2. [**バージョン**] ボックスで、[**変更セット**] を選択し、[**変更セット番号**] を入力してから、[**OK**] をクリックします。
3. [**ラベルの適用**] ダイアログ ボックスで、ラベル名とコメントを入力してから、[**OK**] をクリックします。

変更セットの詳細を表示するには

- **tf changeset** コマンドを使用します。

  以下のコマンドは、[**変更セットの詳細**] ダイアログ ボックスを表示して、変更セット番号 1234 の詳細を表示します。

tf changeset 1234

このダイアログ ボックスを使用して、変更セットに格納されているソース ファイルの表示、チェックインのコメントおよびメモの表示、関連作業項目の表示、変更セットをサーバーにチェックインしたときに生成されたすべてのポリシー警告の表示を行うことができます。

変更セットの詳細を変更するには

- **tf changeset** コマンドを使用して、変更セットに関連付けられているコメントまたはメモのどちらか一方または両方を更新します。
  以下のコマンドは、変更セット番号 1234 の [**変更セットの詳細**] ダイアログ ボックスを表示して、表示されたコメント フィールドを更新します。ここで、コメントを変更することができます。

  tf changeset /comment:"こちらのほうが、前のものよりはるかに良いコメントです。" 1234

  以下のコマンドは、変更セット 1234 に関連付けられたコード レビュー担当者およびセキュリティ レビュー担当者のメモを更新します。

  tf changeset /notes:" Code Reviewer"="C Davis";"Security Reviewer"="F Smith" 1234

変更セットをロールバックして、ソース管理サーバーからその変更を削除するには、Team Foundation Power Tool を使用します。

Team Foundation Power Tool を使用して変更セットをロールバックするには

- **Tfpt rollback** コマンドを使用します。
  以下のコマンドは、変更セット番号 1234 をロールバックします。

TFPT rollback /changeset:1234

このコマンドを使用すると、[Roll Back Changeset] ウィンドウが表示されます。変更セットに入っているファイルを選択して、ロールバックすることができます。

参考資料

- **tf changeset** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/w51xa47k(VS.80).aspx の「Changeset コマンド」を参照してください。
- 変更セットの検索の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181416(VS.80).aspx の「方法 : 変更セットからファイルの以前のバージョンを取得する」を参照してください。
- MSDN から Team Foundation Power Tool (TFPT) をダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyID=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。
- Team Foundation Power Tool の詳細を知りたい方は、http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx の「Power Toy: tfptexe」を参照してください。

チェックイン前にコーディング基準を強制する方法

サーバーにチェックインしたすべてのコードが、規定のコーディング基準を必ず満たすようにするには、TFS チェックイン ポリシーを使用します。

既定では、次のようなチェックイン ポリシーを利用できます。

- **コード分析 –** チェックインの前のコード分析の実行を義務付けます。
- **テスト ポリシー –** チェックインの前のチェックイン テストの実行を義務付けます。
- **作業項目 –** 1 つ以上の作業項目をチェックインに関連付けることを義務付けます。

既定のコード分析チェックイン ポリシーには、マネージ コードとアンマネージ コードの両方を対象

としたコード分析チェックが装備されています。マネージ コードは、コードを静的に分析して、標準の .NET 設計ルール、グローバリゼーション ルール、相互運用性ルール、命名ルール、パフォーマンス ルール、セキュリティ ルールなどにそのコードが準拠していることを確認します。コード分析を拡張する必要が生じた場合、独自のコード分析ルールを開発することができます。

コード分析チェックイン ポリシーを構成するには

1. [チーム エクスプローラ] でチーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] をポイントしてから、[**ソース管理**] をクリックします。
2. [**チェックイン ポリシー**] をクリックしてから、[**追加**] をクリックします。
3. [**チェックイン ポリシー**] リストで、[**コード分析**] を選択してから、[**OK**] をクリックします。
4. 該当するチェック ボックスを選択して、実行したいコード分析の種類を指定します。[**マネージ コードのコード分析を強制**] を選択した場合、[**マネージ コード分析の規則設定**] リスト中の規定のルールを選択します。
5. [**OK**] をダブルクリックします。

重要事項: 上記の手順では、開発者がソース ファイルをチェックインするたびに、構成済みのポリシーが必ず実行されますが、開発者はチェックイン時にいつでもポリシーをオーバーライドすることができます。このシナリオを管理するために、状況を監視したい場合は、ポリシーのオーバーライド イベントをフックすることができます。

プロジェクトごとの品質基準を有効化する独自のカスタム チェックイン ポリシーを作成できます。たとえば、次のようなポリシーを有効化することができます。

- ユーザーが、チェックイン時にコメントを追加する必要がある。
- ユーザーが、コードをチェックインする前に、追加の受け入れテストを実行する必要がある。
- ユーザーが、XML ドキュメント警告を抑止するための特定のプラグマ ディレクティブを C# では使用しない。
- コンパイル中に XML ドキュメントを必ず生成するようにプロジェクトを構成する。

[**チェックイン ポリシーの追加**] ダイアログ ボックスに表示されるカスタム ポリシー プラグインを

作成するには、Visual Studio Team Foundation Server の Software Development Kit (SDK) に付属している拡張性機能を使用します。

参考資料

- カスタム チェックイン ポリシーの作成と使用方法の作成の詳細については、このガイドの「Visual Studio Team Foundation Server でチェックインを作成する方法」を参照してください。
- チェックイン ポリシーをカスタマイズする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインでの選択済みパターンを禁止するサンプル コードの詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインでコメントを実施するサンプル コードは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn Empty」を参照してください。
- 新しいチェックイン ポリシーを登録する方法の詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。
- コード分析ツールの使用法詳細については、http://msdn2.microsoft.com/ja-jp/library/ms182023(VS.80).aspx の「コード分析ツールを使用するためのガイドライン」を参照してください。

チェックイン ポリシーのオーバーライド方法

チェックイン ポリシーをオーバーライドするには、[**ポリシー エラー**] ダイアログ ボックスの [**ポリシー エラーをオーバーライドして、チェックインを続行**] チェック ボックスを選択します。ファイルをチェックインするアクセス許可を受けたすべてのユーザーが、チェックイン ポリシーをオーバーライドすることができます。

チームのメンバによってチェックイン ポリシーがオーバーライドされたときには、それを検出したい場合、Team Foundation Eventing Service を使用してチェックイン イベントをフックすることができます。

参考資料

- チェックイン ポリシーのオーバーライドの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245460(VS.80).aspx の「方法 : チェックイン ポリシーをオーバーライドする」を参照してください。
- Team Foundation Eventing Service の詳細については、http://msdn2.microsoft.com/en-us/library/bb130154(vs.80).aspx の「Eventing Service」を参照してください。
- イベンティングではなく、TFS オブジェクト モデルを使用してポリシーのオーバーライドを検出する方法の詳細については、http://blogs.msdn.com/jmanning/archive/2005/11/04/489193.aspx の MSDN の James Manning 氏のブログ ポストを参照してください。

チェックインを元に戻す方法

ファイルのチェックインを元に戻すには、Team Foundation Power Tools (TFPT) の **ロールバック** コマンドを使用して、そのファイルを前のバージョンに戻します。**ロールバック** コマンドを使用すると、変更セット全体を一度にロールバックできますが、変更セットからファイルを 1 つずつ選択してロールバックしてもかまいません。それが便利なのは、誤ってチェックインしたファイルに加えた変更を元に戻す必要がある場合や、変更が原因で統合ビルドで重大な問題が起きたためにチェックインを元に戻したい場合などです。

ファイルのチェックインを元に戻して、前のバージョンに復帰するには

1. パス内に ¥program files¥Microsoft Team Foundation Server パワー ツールのフォルダが入っているコマンド ウィンドウから以下のコマンドを実行します。

TFPT rollback filename.cs

注意: 元に戻そうとしているチェックインを格納した変更セットの変更セット番号がわかっている場合、次のように、コマンド ラインにその変更セット番号を入力することができます。

TFPT rollback filename.cs /changeset:54

2. **ロールバック** コマンドでは、ローカル ワークスペースが最新のものであることを確認する必要があります。
   [**Roll Back Changeset**] のメッセージに対する応答として、[**はい**] をクリックします。これで、サーバーのファイルを使ってワークスペースが更新されます。
3. コマンド ラインに変更セット番号を指定しなかった場合、[**Find Changeset**] ダイアログ ボックスが表示されます。検索条件を入力するか、または単に [**Find**] をクリックしてから、元に戻したいチェックインを格納している変更セットを見つけ出して選択し、次に [**Roll Back**] をクリックします。
4. [**Roll Back Changeset**] ダイアログ ボックスが表示されます。変更セットには複数のチェックインが格納されていることがあるので、チェックインを元に戻したいファイルを選択してから [**Roll Back**] をクリックする必要があります。
   注意: ファイル名をコマンド ラインに指定した場合、変更セット内のそのファイルだけが選択されます。

TFPT のロールバック コマンドを使用してチェックインを元に戻す場合、以下の点に注意する必要があります。

- **ワークスペースの場所** – TFPTがワークスペースを検索する方法は以下のとおりです。ロールバックをフィルタする引数としてファイル パスを指定した場合、そのファイル パスを使ってワークスペースを検索します。ファイル パスを指定しないと、ローカル ディレクトリを使用して、ワークスペースを識別します (そのローカル ディレクトリがマップされている場合)。このツールが正しいワークスペースを確実に検索できるようにする最もシンプルなアプローチは、ローカル側でマップされたディレクトリからコマンドを実行することです。

- **保留中の変更 –** 保留中の変更を格納している変更セットはロールバックできません。ロールバックを試みると、ツールからエラーがレポートされます。その場合、保留中の変更をシェルブセットに移動してから、すべての保留中の変更を元に戻すかまたはチェックインしたうえで、**ロールバック** コマンドを実行する必要があります。
- **マージ シナリオ –** たった今チェックインしたばかりのファイルのチェックインを元に戻す場合、他のだれかがその間に項目を更新した可能性は低いので、おそらく変更をマージする必要はないと考えられます。ただし、チェックインを元に戻したい場合に、元に戻すチェックインがファイルの最新の変更でなければ、3 方向のマージが必要です。サーバーの現行バージョン、ワークスペース内にあるバージョン、およびロールバック先のバージョンをまとめてマージする必要があります。競合がない場合、ロールバックするバージョンの変更はファイルから抽出され、そのバージョン以降に発生した変更はすべて保持されます。
- **競合の解決 –** マージを実行する必要がある場合、マージ ウィンドウが表示されます。マージを解決するには、項目を選択してから、[**マージ**] をクリックします。まず自動マージが試みられます。それに失敗した場合、マージ ツールが呼び出されて、マージが解決されます。[**すべてを自動マージ**] ボタンを使用すると、マージ リスト中の各項目の自動マージが試みられますが、マージ ツールの呼び出しは試みられません。

参考資料

- MSDN から TFPT ツールをダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyID=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en にアクセスしてください。
- TFPT の詳細について知りたい方は、http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx の「Power Toy: tfptexe」を参照してください。

マージの競合の解決方法

マージの競合を解決するには、Visual Studio のマージ ツールを使用します。マージ中に競合が検出された場合、自動または手動で競合を解決することができます。手動での競合の解決を選択した場合、マ

ージ ツールで、マージ元の変更の保持、マージ先の変更の保持、または競合の解決を行うことができます。

それぞれの分岐の変更のマージ、ワークスペースへのファイルの取り込み、新しいバージョンのファイルのチェックインのときには、競合を解決する必要が生じる可能性があります。競合には、以下の 3 種類があります。

- **バージョン** – 何本にも分かれたパスに沿ってファイルが進展してきた場合。これは、ファイルの編集、名前変更、削除、または削除の取り消しの操作に起因すると考えられます。
- **ファイル名の衝突** – 複数のアイテムが同じパスの占有を試みた場合。
- **ローカル上書き** – ローカルの編集可能ファイルの上書きを試みた場合に、**get** 操作でのみ発生します。

競合のほとんどは、自動的に解決することができます。バージョン競合は、手動のマージ操作へとつながる唯一の種類の競合です。手動マージが最もよく使用されるのは、次のようなシナリオの場合です。

- 1 つのファイルが両方の分岐で編集されて、同じコード行に変更が加えられた場合。
- 基点のないマージが実行された場合に、分岐ファイルどうしの関係が TFS によって認識されていないとき。

マージ ツールでは、各競合の詳細が示されるので、マージ後の最終ファイルにどの変更を保管するかを選択することができます。マージ元の変更の保持、マージ先の変更の保持、またはこの両方の変更の統合を選択できますが、ファイルに直接入力することによって、最終バージョンを手動で変更することもできます。

ファイルにおけるすべての競合を解決した後、ターゲット分岐にチェックインする保留中の変更として、最終バージョンを保存します。

マージは慎重に行ってください。なぜなら、間違いが起きることはよくあり、それが原因でビルドが不安定になることがあるからです。マージを完了したら、その結果のソースをコンパイルし、重大な破壊がないことを確かめる単体テストを実行します。

参考資料

- 競合の解決の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181433(VS.80).aspx の「方法 : 競合を解決する」を参照してください。

競合の回避方法

競合を回避するには

- **チームが効率よくコミュニケーションをとっていることを確認します。**ソース ファイルを処理する場合、そのファイルが処理中であることと、どの局面を更新中であるかを他のチーム メンバに知らせます。自動的な競合の解決によって競合のほとんどは解決できますが、複数の開発者が同じソース ファイル内の同じ機能領域を処理したために、両者とも同じコード行に対して変更を加える可能性がきわめて高くなるといった状況が起きないようにする必要があります。同じコード行上で競合が起きると、競合を手動で解決する必要が生じるので、マージ操作が複雑になってしまいます。
- **他に誰がファイルをチェックアウトしたかを判別します。**ファイルの更新の前に、別のチーム メンバが既にファイルをチェックアウトしたかどうかを調べて確認します。既にチェックアウトしていた場合、何を処理したかをその同僚に問い合わせてから、その同僚の変更が完了するまで待ったほうがよいかどうかを判断します。ただし、自身が、そのファイル内の別のソース コード領域の別の機能を処理していたのであれば、同じファイルの並列処理を続行しても安全であると判断しても差し支えありません。

保留中の変更を表示するには

1. [ソース管理エクスプローラ] で、保留中の変更を表示したいソリューション、プロジェクト、フォルダ、またはファイルを右クリックします。
2. [**保留中の変更を表示**] を選択します。

この方法をとると、選択した範囲内のすべての保留中の変更が表示されます。また、次のようにコマンド ラインのチェックアウト ツールを使用して、保留中の変更を確認することもできます。

tf status /format:detailed /user:*

このコマンドは、すべてのユーザーによって行われたすべての保留中の変更の詳細な状況情報を表示します。その結果のリストでは、誰がどのファイルをチェックアウトしたかが、保留中の変更とともに表示されます。特定のファイルの保留中の変更を表示するには、そのファイル名を tf.exe の最初の引数に指定します。

参考資料

- ワークスペース内での保留中の変更の表示の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181400(VS.80).aspx の「方法 : ワークスペース内のすべての保留中の変更を表示および管理する」を参照してください。
- その他のワークスペースでの保留中の変更の表示の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181401(VS.80).aspx の「方法 : 他のワークスペース内の保留中の変更を表示する」を参照してください。
- **tf status** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/9s5ae285(VS.80).aspx の「Status コマンド」を参照してください。

カスタム チェックイン ポリシーの作成方法

カスタム チェックイン ポリシーを作成するには、クライアント フレームワークから公開されるプラグイン モデルを使用します。

カスタム チェックイン ポリシーを作成して、独自のポリシー ロジックを実装することができます。たとえば、チェックイン時に、ユーザーがコメントを追加したかどうかや、正規表現を正しく使用しているかどうかをチェックすることができます。

プラグインは、ポリシー定義プロセスとポリシー評価プロセスの両方で使用します。ポリシー プラグインは、独立型ユーティリティとしてインストールするか、または別個のアプリケーションの一部とし

てインストールします。すると、ポリシー フレームワークに登録されて、必要時の読み込みが可能になります。

ポリシー プラグインは、以下のインターフェイスを公開する必要があります。

- **IPolicyDefinition – IPolicyDefinition** インターフェイスは、チーム プロジェクトのポリシー要件の定義プロセスで使用されるメソッドを公開します。このインターフェイスは、ポリシー プラグインごとのユーザー インターフェイスを呼び出すためのメソッドなどで構成されます。これにより、ユーザーは簡単にチェックイン ポリシーを定義することができます。
- **IPolicyEvaluation – IPolicyEvaluation** インターフェイスは、チェックイン プロセス中にポリシーが遵守されているかどうかの評価プロセスで使用されるメソッドを公開します。このインターフェイスは、チェックイン操作の内容を受け入れてから、定義されているポリシーが遵守されているかどうかを判断する分析を行うメソッドなどで構成されます。

複数のポリシー プラグインを、1 つのアセンブリにまとめてパッケージ化することができます。唯一の要件として、プラグインを別個のクラスとして実装する必要があります。

注意: 上記のインターフェイスは、**PolicyBase** クラス内で公開されます。**IPolicyDefinition** インターフェイスや **IPolicyEvaluation** インターフェイスを実装する代わりの方法として、**PolicyBase** からクラスを派生することができます。

参考資料

- カスタム チェックイン ポリシーの作成と使用方法の作成の詳細については、このガイドの「Visual Studio Team Foundation Server でチェックインを作成する方法」を参照してください。
- チェックイン ポリシーをカスタマイズする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。
- チェックインでの選択済みパターンを禁止するサンプル コードの詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy

to Disallow Certain Patterns」を参照してください。

- チェックイン時のコメントを実施するサンプル コードは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーを登録する方法の詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。
- コード分析ツールの使用法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms182023(VS.80).aspx の「コード分析ツールを使用するためのガイドライン」を参照してください。
- 新しいポリシーの作成方法の詳細については、http://msdn2.microsoft.com/en-us/library/bb130343(VS.80).aspx の「Policy Plug-ins」を参照してください。

チェックアウト、取得、およびロック

- **TFS に対するコンピュータの同期方法**
- **編集用のファイルの準備方法**

TFS に対するコンピュータの同期方法

コンピュータをバージョン管理サーバーに同期するには、**tf get** コマンドを使用します。これは、共有作業の最新コピーが必ず各自の手元に揃っているようにするために、チームの中でコンピュータを確実に互いに同期させるための最速の方法です。古くなったファイルだけでなく、すべてのファイルをダウンロードするには、Visual Studio 2005 のコマンド プロンプト ウィンドウから以下のコマンドを実行します。

```
tf get /all
```

このコマンドを使用すると、自身のコンピュータ上に既にある書き込み可能などのローカル ファイル

は上書きされません。書き込み可能なローカル ファイルを上書きして、コンピュータをサーバーに完全に同期したい場合、次のように、**/force** スイッチを使用します。

tf get /force

このコマンドは、書き込み可能なローカル ファイルを上書きしますが、保留中の編集の対象でもある書き込み可能なファイルは上書きしません。保存する必要のあるファイルに対する保留中の編集がある場合、サーバーとの再同期の前に、その編集をチェックインまたはシェルビングします。

Visual Studio から同じ操作を実行するには

1. チーム エクスプローラで、[ソース管理] フォルダをダブルクリックし、サーバーまたはチーム プロジェクトを右クリックしてから、[**特定バージョンの取得**] をクリックします。
2. [**チェックアウトされていない書き込み可能ファイルを上書きする**] および [**ワークスペースに既に存在するファイル バージョンの取得を強制**] を選択します。
3. [**種類**] ボックスで、[**最新バージョン**] が選択されていることを確認してから、[**取得**] をクリックします。

コンピュータをバージョン管理サーバーに完全に同期したい場合、Visual Studio から利用できる [**最新バージョンの取得**] オプションを使用しないでください。このコマンドは、ワークスペース内にまだないファイルをダウンロードするだけで、ローカル ディレクトリにチェックアウトした書き込み可能なファイルを上書きしません。よって、コンピュータは、サーバーと同期しないままになります。

参考資料

- **get** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/fx7sdeyf(VS.80).aspx の「Get コマンド」を参照してください。

編集用のファイルの準備方法

編集目的でファイルを準備するには、まず Team Foundation Server ソース管理から最新バージョンを入手してから、編集用にチェックアウトする必要があります。

編集用のファイルを準備するには

1. ソース管理エクスプローラでファイルを選択して右クリックし、[**最新バージョンの取得**] をクリックします。
   すると、ファイルの最新バージョンの読み取り専用コピーが、コンピュータ上のワークスペースにダウンロードされます。
2. そのファイルを右クリックしてから、[**編集用にチェックアウト**] をクリックします。
3. 必要なロックの種類を選択します。自身が作業をしている間、他のユーザーがファイルのチェックインとチェックアウトを行えるようにするには、[**なし**] を選択します。
   一般的に、この種の共有チェックアウトが推奨されます。なぜなら、競合が発生しても、ほとんどは自動的に解決できるからです。

注意: [**最新バージョンの取得**] 操作によってファイルがチェックアウトされることはありません。また、[**編集用にチェックアウト**] 操作によって [**取得**] が実行されることもありません。この 2 つのステップは、1 つずつ実行する必要があります。この動作は、Microsoft Visual SourceSafe の動作とは異なります。

ロックの種類を選択するときは、以下の点に気をつけてください。

- 共有チェックアウト ([**なし**]) を使えば、他のだれかが同じファイルを処理することはなくなるので、開発プロセスでボトルネックが生じることもなくなります。
- ファイルをロックしても差し支えないのは、そのファイルの編集中に、複雑な手動のマージ操作につながる競合が発生するおそれがある場合のみです。
- [**チェックアウト**] のロックの種類を選択して、他のユーザーがファイルのチェックアウトおよびチェックインを行わないようにします。このオプションを使用すると、他の誰もファイルを編集できなくなります。それは、開発プロセス中にボトルネックが生じる可能性を示唆します。このオプションでは、他のユーザーがファイルにその他の変更を加える可能性を排除したうえで、ソース管理データベースに変更を確実に適用することができます。

- 他のユーザーが、ファイルのチェックアウトはできても、チェックインはできないようにするには、[**チェックイン**] のロックの種類を選択します。この場合も、このオプションによって、競合を生じないで必ず編集をチェックインできるようになります。

参考資料

- **チェックアウト** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/1yft8zkw(VS.80).aspx の「チェックアウトと編集のコマンド」を参照してください。

コードの共有

- **コードの共有方法**
- **共有バイナリの管理方法**

コードの共有方法

複数のチーム プロジェクトで使用されているソース コードがある場合、その所有者のチームのチーム プロジェクト内でそのソース コードを管理するか、またはその共有ソース コード専用のチーム プロジェクトを作成することができます。

共有ソース コードを実行するチームには、以下の 2 つの選択肢があります。

1. 共有場所からのコードの参照
2. 共有コードの分岐

1. 共有場所からのコードの参照

この場合、共有コードを実行するプロジェクトは、単に共有場所からクライアント マシン上のワークスペースにソースをマップするだけで済みます。すると、共有場所のソースとクライアント サイドのプロジェクトを統一する構成が作成されます。

このアプローチの長所は、最新のソースをワークスペースに取り込むたびに、共有ソースに加えられた変更も取り込まれる点にあります。たとえば、Client および Shared Code という 2 つのチーム プロジェクトがあると仮定します。ただし、Shared Code は、共有ソースの場所であるとします。共有場所からコードを参照するときは、この 2 つのプロジェクトは、以下に示されているとおり、クライアントの共有ドライブ上の共通パスを共有することになります。

- c:¥TestProject¥Client
- c:¥TestProject¥Shared Code

どちらのプロジェクトも、これらのハードディスク パスに対するワークスペース マッピングを持つことになります。

ソース管理フォルダ	**ローカル フォルダ**
$/Client	c:¥TestProject¥Client
$/Shared Code	c:¥TestProject¥Shared Code

詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の「Working with multiple team projects in Team Build」を参照してください。

2. 共有コードの分岐

この場合、共有コードを実行するプロジェクトは、共有場所からそれぞれのチーム プロジェクトにソースを分岐することができます。すると、共有場所のソースとサーバー サイドのプロジェクトを統一する構成が作成されます。

共有ソースに加えた変更は、2 つの分岐のマージ プロセスの一環として取り込まれる点が異なります。それに起因して、共有ソースにおける変更を取り込む決定をもっと明示的に下せるようになります。

たとえば、Client および Shared Code という 2 つのチーム プロジェクトがあると仮定します。ただし、Shared Code は、共有ソースの場所であるとします。以下のステップを行って、共有場所のコードを分岐します。

a. [ソース管理エクスプローラ] で、Shared Code のルート フォルダを右クリックします。
b. [**分岐**] オプションを選択します。
c. [**分岐**] ボックスの [**ターゲット**] を Client チーム プロジェクトのルート フォルダに設定してから、[**OK**] をクリックします。
d. 分岐操作が完了した後、分岐後のソース コードをチェックインするのを忘れないでください。

参考資料

- 詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の「Working with multiple team projects in Team Build」を参照してください。
- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- プロジェクト参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。

共有バイナリの管理方法

共有バイナリの管理は、共有ソースの管理に似ています。つまり、バイナリの保管場所と、チームからのそのバイナリへのアクセス方法を決定する必要があります。

バイナリの保管の際は、以下のオプションを利用できます。

- 共有バイナリを所有するのがどのチームか明白な場合は、そのバイナリを該当するチーム プロジェクトに保管します。
- 共有バイナリの所有者が明白でない場合、その共有バイナリ専用のチーム プロジェクトを作成します。

別のプロジェクトのバイナリを使用する際は、以下のオプションを利用できます。

- 通常、共有バイナリは、定期的にしか更新されません。それに該当する場合、共有場所から実行者のチーム プロジェクトに分岐します。バイナリが変更された場合、マージを実行して、最新バージョンを取得することができます。
- 共有バイナリに常時同期している必要がある場合、共有場所からクライアント マシン上のローカル ワークスペースにソースをマップします。

共有バイナリをプロジェクトに分岐するには

1. [ソース管理エクスプローラ] で、共有バイナリのルート フォルダを右クリックします。
2. [**分岐**] オプションを選択します。
3. [**分岐**] ボックスの [**ターゲット**] を Client チーム プロジェクトのルート フォルダに設定してから、[**OK**] をクリックします。
4. 分岐操作が完了した後、分岐後のソース コードをチェックインするのを忘れないでください。

ワークスペースのマッピングまたは分岐のどちらを使用する場合も、プロジェクト内の共有バイナリの場所がどこかを明白にする命名規則を使用する必要があります。以下に例を示します。

- **Main**
  - **Source** – このプロジェクトのコードを格納します。
  - **Lib** – 共有バイナリを格納します。

参考資料

- 詳細については、http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx の「Working with multiple team projects in Team Build」を参照してください。
- 分岐およびマージの入門書については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- プロジェクト参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。

依存関係

- **Web サービスの依存関係の管理方法**
- **データベースの依存関係の管理方法**

Web サービスの依存関係の管理方法

一般的に、実動環境の Web サービスの URL (Uniform Resource Locator) 参照は、開発およびテストの環境のものとは異なります。各 Web サービス URL 参照を簡単に管理できるようにするには、URL 値をユーザー構成ファイルに指定する必要があります。これは、メインの App.config ファイルに影響を与えずに、個々の開発者およびテスト担当者が変更できるファイルです。指定するためには、Web サービス参照の [**URL の動作**] プロパティを [**ダイナミック**] に設定します。Web サービス URL は、ユーザー構成ファイルから保管して参照します。

Web 参照が追加されると、既定では、Visual Studio はこのプロパティの値を [**ダイナミック**] に設定します。

この値がこれまでどおり [ダイナミック] に設定されていることを検証するには

1. [ソリューション エクスプローラ] で、Web 参照のリストを展開します。
2. リスト中の各 Web 参照を選択します。
3. 各 Web 参照ごとに、[**URL の動作**] プロパティの値が [**ダイナミック**] に設定されているかどうかをチェックします。

Web サービス URL をユーザー構成ファイルに指定します。

Web 参照を初めて追加した場合、App.config ファイルは次のようになります。

```
<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="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>
```

このファイルには、このプロキシの生成時に Visual Studio が検出した Web サービスのアドレスを記載した新しい構成セクションがあります。

User.config ファイルを作成するには

1. [ソリューション エクスプローラ] で、Web サービス参照を格納しているプロジェクトを右クリックし、[**追加**] をポイントしてから、[**新しい項目**] をクリックします。
2. [**アプリケーション構成ファイル**] を選択し、名前を **User.config** に変更してから、[**追加**] をクリックします。
3. *<YourProject.Properties.Settings>* 要素設定を、アプリケーション構成ファイル (App.config) から User.config ファイルにコピーします。このファイルには、ランタイムをリダイレクトされる要素のみが入っていなければなりません。以下の例に示されているとおり、<?xml> ディレクティブと <configuration> 要素 (ある場合) を削除します。

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

4. [ソリューション エクスプローラ] で、User.config ファイルを右クリックし、[**プロパティ**] をク

リックしてから、[**出力ディレクトリにコピー**] プロパティを [**新しい場合はコピーする**] に設定します。

個々の開発者は、該当する Web サービス URL 参照を参照するように各自の User.config ファイルを設定する必要があります。

Web サービス URL へのアクセス時に User.config ファイルを参照するように App.config ファイルを更新するには

1. メインのアプリケーション構成ファイルの *<YourProject.Properties.Settings>* 要素に **configSource="user.config"** 属性を追加します。
   これにより、このセクションから情報にアクセスすると、指定のユーザー構成ファイルにランタイムが自動的にリダイレクトされます。
2. *<YourProject.Properties.Settings>* 要素から内容を削除します。

これで、App.config は次のようになるはずです。

```
<?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="false" />
</sectionGroup>
</configSections>
<applicationSettings>
<YourProject.Properties.Settings configSource="user.config">
</YourProject.Properties.Settings>
</applicationSettings>
</configuration>
```

上記の例では、*YourProject* は、Web サービス参照を格納しているプロジェクトの名前です。App.config ファイル内では、<*SomeService.Properties.Service*> 要素は空であることを確認します。

重要事項:

- User.config ファイルをソース管理に追加しないでください。追加しないことによって、各開発者 (およびテスト チーム) は、各自の User.config ファイルを使って、該当する URL に明示的にバインドすることができます。これが行われないようにするには、初めてファイルをチェックインするときに、User.config ファイルのチェック ボックスをクリアします。次に、ソリューション エクスプローラ内のファイルを右クリックし、[**保留中の変更を元に戻す**] オプションを選択して、ファイルがソース管理の配下に入らないようにします。
- ソース管理には、たとえばテスト用や実動用の、その他の User.config ファイルが入っていることがあります。そのようなファイルは、テストおよび実動環境の管理に携わるユーザーによって管理される必要があります。それらのテスト用および実動用の User.config ファイルを、Web サービス プロジェクトの一部として保管してはなりません。つまり、ソース管理システムのどこか別の領域に配置する必要があります。
- グローバル User.config ファイルをソース管理に保管します。これには、ルート要素 (<設定> 要素ではない) のみを入れるか、または、Web サービスの既定の場所をこれに指定することができます。構成システムが作動するには、User.config ファイルが存在している必要があります。

このメカニズムを活用するには、User.config ファイルが存在している必要があることを理解することが大切です。いずれかのチーム メンバの責任のもとに、実動リリース用およびテスト環境用のビルドの作成が、必ず適正な環境で行われるようにする必要があります。このビルド ステップの一環として、それに適した User.confg ファイルをソース管理システムから取り出して正しい場所にコピーすることによって、MSBuild がそれを見つけられるようにしなければなりません。

参考資料

- 詳細については、http://msdn2.microsoft.com/en-US/library/aa290068(VS.71).aspx の「Web Projects and Source Control Integration in Visual Studio .NET」を参照してください。
- **ApplicationSettingsGroup** クラスの詳細については、

http://msdn2.microsoft.com/ja-jp/library/system.configuration.applicationsettingsgroup.aspx の「ApplicationSettingsGroup クラス」を参照してください。

データベースの依存関係の管理方法

以下に示す手順は、ユーザー構成ファイル内でデータベース接続文字列の保管と参照を行う方法を説明しています。

ユーザー構成ファイルを使用してデータベース接続文字列を保管するには

1. 次のようにして、メイン アプリケーション構成ファイルの <connectionStrings> 要素に **configSource="user.config"** 属性を追加します。

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

2. メイン アプリケーション構成ファイルをオーバーライドするには、User.config ファイル (アプリケーション構成ファイルと同じフォルダ内に配置されています) を作成してから、それに似た <connectionStrings> 項目をこのファイルに追加します。
   以下の接続文字列は、ローカル データベースを参照することに注意してください。

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

3. プロジェクト内で以下のコードを使用して、ユーザー構成ファイルから接続文字列を取り出します。このコードは、**System.Configuration.ConfigurationManager** クラスの **ConnectionStrings** プロパティを使用します。Win フォーム アプリケーションで、**System.Configuration.dll** への参照を明示的に追加する必要があります。

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

4. アプリケーション コードと一緒に、必ず User.config ファイルが配置されるようにします。[ソリューション エクスプローラ] で、User.config ファイルを右クリックし、[プロパティ] をクリックしてから、[プロパティ] ペインで [出力ディレクトリにコピー]プロパティを [新しい場合はコピーする] に設定します。

   User.config ファイルをソース管理に追加しないでください。追加しないことによって、各開発者 (およびテスト チーム) は、各自の User.config ファイルを通して、接続文字列を明示的に指定することができます。ソース管理には、たとえばテスト用や実動用の、その他の User.config ファイルが入っていることがあります。そのようなファイルは、テストおよび実動環境の管理に携わるユーザーによって管理される必要があります。それらのテスト用および実動用の User.config ファイルを、データベース プロジェクトの一部として保管してはなりません。つまり、ソース管理システムのどこか別の領域に配置する必要があります。

   ソース管理では、実動やテストなどの、使用する環境ごとに User.config ファイルがなければなりません。そのような構成ファイルは、データベース用の接続文字列を指定していなければなりません。構成システムが作動するには、User.config ファイルが存在している必要があります。

ヒント: 既定では、ソリューションを追加すると、ユーザー構成ファイルが自動的にソース管理に追加されます。これが行われないようにするには、初めてファイルをチェックインするときに、User.config ファイルのチェック ボックスをクリアします。次に、ソリューション エクスプローラ内のファイルを右クリックし、[保留中の変更を元に戻す] を選択して、ファイルがソース管理の配下に入らないようにします。

このメカニズムを活用するには、User.config ファイルが存在している必要があることを理解することが大切です。いずれかのチーム メンバの責任のもとに、実動リリース用およびテスト環境用のビルド

の作成が、必ず適正な環境で行われるようにする必要があります。このビルド ステップの一環として、それに適した User.confg ファイルをソース管理システムから取り出して正しい場所にコピーすることによって、MSBuild がそれを見つけられるようにする必要があります。

参考資料

- 構成ファイルを使用してデータ ソースを定義する方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms243192(VS.80).aspx の「チュートリアル : データ ソースを定義するための構成ファイルの使用」を参照してください。
- **configSource** 属性の詳細については、http://msdn2.microsoft.com/ja-jp/library/system.configuration.sectioninformation.configsource.aspx の「SectionInformation.ConfigSource プロパティ」を参照してください。

分散/リモート開発

- **TFS へのインターネット経由のアクセス方法**
- **TFS バージョン管理プロキシのパフォーマンスの最適化方法**

TFS へのインターネット経由のアクセス方法

インターネットを介して TFS にアクセスできるようにするには、以下の 3 つのソリューションのいずれかを選択します。

- **VPN 接続を使用します。**仮想プライベート ネットワーク (VPN) を通して TFS にアクセスできるようにすることができます。
- **リバース プロキシを介して、TFS を発行します。**Microsoft の ISA (Internet Security and Acceleration) サーバーなどのリバース プロキシを通して、TFS にアクセスできるようにすることができます。
- **エクストラネット内で TFS を見つけ出します (「ホストされるシナリオ」)。**エクストラネット上で TFS サーバーをホストすることができます。

VPN アクセスを使用するリモート ユーザーをサポートしている場合、VPN ソリューションを使用します。これは、最も簡単に実用化できるソリューションです。これにより、周知のセキュリティが提供され、すべての TFS 機能へのリモート アクセスが可能になり、TFS プロキシを使用したパフォーマンスの改善が可能になります。このソリューションでは、TFS は内部ネットワーク内に設置されるので、外部ユーザーは VPN 経由でそこにアクセスします。内部ユーザーは、TFS に直接アクセスします。

VPN アクセス権がないか、またはドメインへのアクセス権のないリモート ユーザーをサポートしている場合、リバース プロキシのシナリオを使用します。このソリューションをセットアップするほうが難しいですが、リモート ユーザーは、VPN を必要としないで、内部に配置された TFS にアクセスできるようになります。このソリューションでは、TFS は内部ネットワーク内に設置され、ISA サーバーなどの 1 つ以上のリバース プロキシ コンピュータが、インターネットから TFS にクライアント要求を配送します。

コミュニティ開発サイトなどの、TFS インストール環境を専用で使用する一群のリモート ユーザーをサポートしている場合、エクストラネット シナリオを使用します。このソリューションの場合に、リモート ユーザーと内部ネットワーク リソースとの間の隔絶が最も大きくなります。このソリューションでは、外部クライアントのみが TFS にアクセスし、TFS は、エクストラネット上のファイアウォールの外部に設置されます。

リモートの Team Foundation Server に接続している複数のクライアントから成るオフィスをサポートしている場合、リモート オフィス内に Team Foundation Server プロキシをインストールして構成する必要があります。この場合、リモート オフィスのプロキシ サーバー上でソース管理ファイルをキャッシュすることによって、パフォーマンスが向上します。

リモート TFS に接続する単一のクライアントをサポートしている場合、TFS に直接接続するようにクライアントを構成します。

参考資料

- TFS リモート アクセスのシナリオについて詳しく知りたい場合は、このガイドの「第 17 章 - Team Foundation Server に対するインターネット アクセスの提供」を参照してください。
- Team Foundation Server プロキシを詳しく知りたい場合は、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx の「Team Foundation Server のプロキシとソース管理」を参照してください。
- TFS プロキシのパフォーマンスの検査の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252455(VS.80).aspx の「方法 ：Team Foundation Server Proxy のキャッシュのパフォーマンスをチェックする」を参照してください。

TFS バージョン 管理プロキシのパフォーマンスの最適化方法

リモート場所で、Team Foundation Server プロキシをインストールして構成します。この場合、リモート オフィスのプロキシでソース管理ファイルをキャッシュすることによって、パフォーマンスが向上します。

プロキシを構成してパフォーマンス最適化するには

1. キャッシュが使用可能になっていることを確認してから、キャッシュのパフォーマンスを監視します。パフォーマンス カウンタ (既定でインストールされます) とイベント ログ (エラー/警告に関する) を定期的に監視し、プロキシがどのように稼働しているかを確認します。

   注意: TFS プロキシは、ProxyStatistics.xml という名前の XML ファイルにキャッシュ パフォーマンス統計を保存します。この統計の保存サイクル間隔は、変更可能です。ProxyStatistics.xml ファイルは、プロキシのインストール ディレクトリの App_Data フォルダ内に置かれます。
2. スケジュールしたタスクを定期的に実行して、最新のファイルをプロキシ サーバーに取り出します。これにより、最新バージョンのファイルがプロキシ キャッシュに必ず揃っているようになり、ファイルを求めるその後のクライアント要求は、必ずキャッシュ ヒットすることになります。
3. 帯域幅の狭いネットワーク (3 Mbps 未満) 経由で大きなファイルをダウンロードすることになることがわかっている場合は、Web.config の executionTimeout を適切な値に設定してください。デフォルトの値は 1 時間、つまり <httpRuntime executionTimeout="3600"/> です。

参考資料

- TFS リモート アクセスのシナリオについて詳しく知りたい場合は、このガイドの「第 17 章 - Team Foundation Server に対するインターネット アクセスの提供」を参照してください。
- Team Foundation Server プロキシを詳しく知りたい場合は、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx の「Team Foundation Server のプロキシとソース管理」を参照してください。
- TFS プロキシのパフォーマンスの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252455(VS.80).aspx の「方法 ：Team Foundation Server Proxy のキャッシュのパフォーマンスをチェックする」を参照してください。
- TFS プロキシのファイル キャッシュ サーバーについて詳しく知りたい場合は、http://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?culture=ja-jp&EventID=1032291120&CountryCode=JPの MSDN Webcast の「イベントの概要」を参照してください。

移行

- **Visual SourceSafe からのソースの移行方法**
- **他のバージョン管理システムからのソースの移行方法**

Visual SourceSafe からのソースの移行方法

VSS からソースを移行するには、次のようなステップを行います。

注意: この手順を実行するには、Team Foundation Administrators グループのメンバでなければなりません。

1. VSS を準備します。移行の準備をするには、VSS データベースをバックアップし、ファイルがチェックイン済みであることを確認し、Visual SourceSafe の分析ツールを使用して、既存のデータベース内のデータ保全性に関する問題を確認して解決します。

2. プロジェクトを分析します。次のように、コンバータ コマンド ライン ツール (VSSConverter.exe) を使用して、設定 XML ファイルの名前と一緒に分析コマンド スイッチを渡し

ます。

VSSConverter analyze conversionsettings.xml

サンプルの XML 設定ファイルを以下に示します。

```
<?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>
```

設定ファイルには、VSSDatabase の名前が入っています。Source Safe の .ini ファイルを格納しているフォルダを指すように、**name** 属性を設定します。Project 要素は、VSS データベース内で変換しようとしているプロジェクトのパスを定義します。VSS データベース全体を移行するには、<Project Source="$/"></Project> を使用します。

VssConverter.exe **analyze** コマンドは、usermap.xml ファイルも生成します。このファイルにマッピングを追加することによって、バージョン履歴などに関連した名前を、VSS ログイン名から TFS Windows ログイン名に変更することができます。

3. プロジェクトを移行します。移行したいフォルダを選択してから、次のように **migrate** 引数を含む VSSConverter.exe を使用します。

VSSConverter migrate conversionsettings.xml

この場合も、構成設定 XML ファイルを渡しますが、次のように 2 つの重要な追加があります。

```
<?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>
```

<Project> 要素に **Destination** 属性が追加されていることに注意してください。この属性は、TFS 内のチーム プロジェクト (これは、あらかじめ作成しておく必要があります) を指します。また、TFS アプリケーション層の接続の詳細を示す <Settings> 要素が追加されていることにも注意してください。

参考資料

- 移行の準備の詳細については、

http://msdn2.microsoft.com/ja-jp/library/ms181246(vs.80).aspx の「チュートリアル : Visual SourceSafe から Team Foundation への移行の準備」を参照してください。

- 移行の実行方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181247(VS.80).aspx の「チュートリアル : Visual SourceSafe から Team Foundation への移行」を参照してください。
- Visual SourceSafe コンバータの制限事項の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252491(VS.80).aspx の「Visual SourceSafe コンバータの制限事項」を参照してください。

他のバージョン管理システムからのソースの移行方法

移行元のバージョン管理システムからファイルを手動でエクスポートしてから、Team Foundation Server バージョン管理にそのファイルをインポートすることができます。移行元のバージョン管理システムのファイル履歴やその他の属性を保存したい場合、Team Foundation Server バージョン管理オブジェクト モデルを使用して、独自の移行ツールを作成することができます。

現在、マイクロソフトは、ClearCase コンバータへの取り組みを進めています。このコンバータが提供可能になったときは、http://blogs.msdn.com/tfs_migration の TFS Migration ブログで発表されます。

ComponentSoftware によって、GNU RCS、CS-RCS、GNU CVS、Subversion (SVN)、および Visual SourceSafe (VSS) 対応のコンバータが作成されています。

参考資料

- Visual Studio Team Foundation Server SDK をダウンロードするには、http://go.microsoft.com/fwlink/?linkid=68586 にアクセスしてください。
- Team Foundation バージョン管理の拡張性の詳細については、http://msdn2.microsoft.com/en-us/library/bb187335(VS.80).aspx の「Walkthru: The Version Control Object Model」を参照してください。

- ComponentSoftware コンバータの詳細については、http://www.componentsoftware.com/Products/Converter/ の ComponentSoftware の Web サイトを参照してください。

プロジェクト/ワークスペースの管理

- **複数のチーム プロジェクトとの比較における単一チーム プロジェクトの選択方法**
- **ソース ツリーの編成方法**
- **ワークスペース マッピングの定義方法**
- **コンピュータ上でのワークスペースを使用したコード変更の分離方法**

複数のチーム プロジェクトとの比較における単一チーム プロジェクトの選択方法

プロジェクト構造の計画を立てるときは、一般的なプロジェクト編成戦略を評価し、各自の組織の規模、サーバーの制約事項、およびプロセス ワークフローに最も合う構造を選択します。プロジェクトは、組織で発生しうる最大の作業単位を表すための語です。望ましくは、1 つのプロジェクトと複数のプロジェクトの作成を比較したうえで、選択を行う必要があります。複数のプロジェクトは、やむをえない理由がある場合に使用します。たとえば、チームが変更になるか、またはリリース間で重大な変更があったために、不要な作業項目 (バグなど) を持ち越したくない場合や、プロセス テンプレートにおいて変更があった場合などです。

複数のプロジェクトと比較して、1 つのプロジェクトを作成する主な利点としては、要求、機能、シナリオ、次のリリースに持ち越されるバグの移動が簡単になります。

決断での最も重要な点は次のとおりです。

- 作業項目およびその他の資産を次のリリースに引き継ぐのが望ましいか。引き継ぎたい場合、同一プロジェクト内のすべてのリリースを保持するように選択を行います。
- 新しいリリースに移ったときに、新しいプロセスと作業項目の構造をゼロから作成するのが望ましいか。作成したい場合、リリースごとに新しいプロジェクトを作成するように選択を行います。

一般的なプロジェクトの構造をいくつかを以下に示してあります。

- **アプリケーションごとに 1 つのプロジェクト。**アプリケーションのすべてのバージョンを格納するプロジェクトを 1 つ使用します。そのプロジェクトでは、リリースごとに分岐を 1 つずつ作成します。
  - **この構造を使用する理由:** コード、作業項目、およびその他の資産を簡単に持ち越すため。
  - **この構造を使用しない理由:**
    - 並行しているリリースでは、作業項目のスキーマ、チェックイン ポリシー、およびプロセス ガイダンスを共有することが強制されます。
    - レポーティングはさらに複雑です。これは、デフォルトでプロジェクト全体を対象とするため、リリースごとにフィルタリングを追加する必要があるためです。
    - 数百のアプリケーションがあり、それぞれが独自のプロジェクト内にある場合は、TFS のパフォーマンスおよび拡張性の制約の問題に直面します。
    - 複数のリリースにわたって「荷物」が累積されます。これを取り除くための最も簡単な方法は、新しいチーム プロジェクトを作成し、そのプロジェクトで継続して使用したいコードを分岐することです。
- **リリースごとに 1 つのプロジェクト。**アプリケーションのバージョンごとに新しいプロジェクトを作成します。
  - **この構造を使用する理由:**
    - 各リリースごとにまったく新しいプロジェクトから開始したい場合は、この構造が有効であるため。
    - 古いプロジェクトは、メンテナンスで使用することができ、別の支援エンジニアリング チームへハンドオフすることができます。
    - プロジェクトからプロジェクトにソースを移動するのは簡単です。
  - **この構造を使用しない理由:**
    - ソース コードをプロジェクトから別のプロジェクトへ移動するのは非常に簡単ですが、作業項目やその他の TFS 資産をプロジェクトから別のプロジェクトへ移動するのは難しいです。作業項目は、一度に 1 つずつしか別のプロジェクトにコピーできないため、いくつかまとめて作業項目をコピーしたい場合は、ユーティリティを作成する必要があります。
    - アプリケーションとリリースが何百もあり、それぞれが別々のプロジェクトに含まれている場合は、TFS のパフォーマンスと拡張性が限界に達します。

戦略を選択する場合は、以下の考慮事項に留意してください。

- 既存のチーム プロジェクトの構造を変えるのは難しいため、長い期間利用できる構造を選んでください。
- ソースは、次のようにしてチーム プロジェクトどうしで簡単に共有できます。
  - あるプロジェクトから別のプロジェクトへソースを分岐します。
  - 別のプロジェクトから自身のワークスペースへソースをマップします。
- Team Foundation Server で対応できるプロジェクトの個数は、MSF Agile プロセス テンプレートを使用した場合は 500 個、MSF CMMI プロセス テンプレートを使用した場合は 250 個まで拡張できます。MSF for CMMIR (MSF CMMI) プロセス テンプレートを使用した場合は 250 個まで拡張できます。自分だけのプロセス テンプレートを作成する場合、または既存のプロセス テンプレートをカスタマイズする場合は、サーバーの拡張性に一番大きく影響するのが作業項目スキーマであることを忘れないでください。スキーマが複雑だと、サーバーで対応できるプロジェクトの個数が減ります。

参考資料

- プロジェクト戦略の詳細については、http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx の Eric Lee 氏のブログ ポスト「When to use Team Projects」を参照してください。

ソース ツリーの編成方法

ソース ツリー構造は、フォルダ構造、ファイル構造、および分岐構造を組み合わせて構成されます。メイン分岐では、以下のフォルダ構造およびファイル構造が、さまざまな規模のチームにとって有効であることが確証されています。

- **Main** – 製品の出荷に必要なすべての資産のコンテナ。
  - **Source** – ビルドに必要なあらゆるもののコンテナ。
    - **Code - ソース コードのコンテナ**
    - **Shared Code** – 他のプロジェクトから取り込んで共用されるソース コードのコンテナ。

        - **Unit Tests - 単体テストのコンテナ**。
        - **Lib** - バイナリの依存関係用のコンテナ。
    - **Docs** – 製品に付属する資料のコンテナ。
    - **Installer** – インストーラのソース コードおよびバイナリ用のコンテナ。
    - **Tests** – テスト チームのテスト用のコンテナ。

メイン分岐から作成するどの分岐も、このフォルダ構造とファイル構造を新しい分岐にコピーします。以下に例を示します。

- **Development** – 開発分岐
    - **Source** – ビルドに必要なあらゆるもののコンテナ。
        - **Code - ソース コードのコンテナ**
        - **Shared Code** – 他のプロジェクトから取り込んで共用されるソース コードのコンテナ。
        - **Unit Tests - 単体テストのコンテナ**。
        - **Lib** - バイナリの依存関係用のコンテナ。
- **Main** – 統合分岐
    - **Source** – ビルドに必要なあらゆるもののコンテナ。
        - **Code - ソース コードのコンテナ**
        - **Shared Code** – 他のプロジェクトから取り込んで共用されるソース コードのコンテナ。
        - **Unit Tests - 単体テストのコンテナ**。
        - **Lib** - バイナリの依存関係用のコンテナ。
    - **Docs** – 製品に付属する資料のコンテナ。
    - **Installer** – インストーラのソース コードおよびバイナリ用のコンテナ。
    - **Tests** – テスト チームのテスト用のコンテナ。

参考資料

-

ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 ： ワークスペースを作成する」を参照してください。

ワークスペース マッピングの定義方法

サーバー上のソース管理ファイルおよびフォルダをローカル ドライブにマップするには、ワークスペース マッピングを定義します。

ハードディスク上にまだないプロジェクト用のワークスペース マッピングを作成するには

1. [ソース管理エクスプローラ] で、ルート ソース フォルダを選択します。
2. [**最新バージョンの取得**] を右クリックして選択します。
3. ワークスペースのマップ先のローカル フォルダを選択します。

ハードディスク上に既にあるプロジェクト用のワークスペース マッピングを変更するには

1. [**ファイル**] → [**ソース管理**] → [**ワークスペース**] の順にクリックします。
2. [**ワークスペースの管理**] ボックスを使用して、既存のワークスペースの追加、削除、または編集を行います。

既存のワークスペース マッピングを表示するには

1. [ソース管理エクスプローラ] で、ソース フォルダを選択します。
2. 右クリックしてから、[**プロパティ**] をクリックします。
   ローカル ドライブに対するワークスペースのマッピングが [**ローカル名**] フィールドに表示されます。

次のような、ワークスペース マッピングの作成のベスト プラクティスを考察してみましょう。

- **チーム プロジェクトのルート レベルでマッピングを作成します。**新しいチーム プロジェクトの場合、チーム プロジェクトのルート ($/MyTeamProject) を、たとえば C:¥TeamProjects などの、ローカル ドライブ上の同じ名前の付いたフォルダにマッピングします。マッピングは再帰的なため、ソース管理内の構造とまったく同じローカル フォルダ構造全体を自動的に作成します。

- **共有コンピュータ上で、固有のローカル フォルダ パスを使用します。**1 台のコンピュータ上で 2 人のユーザーが、同じワークスペース マッピングを共有することはできません。たとえば、同僚 2 人が、同一のチーム プロジェクト ($/MyTeamProject) を、ローカル コンピュータ上の同じフォルダにマップすることはできません。この場合は、[マイ ドキュメント] の下にマッピングを作成する (この場合、場所パスが長くなってしまいます) か、またはローカル コンピュータ上でチーム フォルダの命名規則を設定します (たとえば C:¥TeamProjects¥User1 や C:¥TeampProjects¥User2 など)。
- **ツリー全体が必要とは限りません。**パフォーマンスを向上してディスク サイズの必要量を削減するには、開発プロジェクトで必要なファイルのみをマップしてください。一般的には、作業対象のソリューションに関係するものファイルおよびプロジェクトのみが必要です。
- **プロジェクト間の依存関係をサポートするワークスペース マッピングに対し、マッピングの使用を避けます。**一般的に、複数のチーム プロジェクトにまたがる依存関係を確立しないようにします。つまり、すべての関連/依存関係にあるソリューション/プロジェクトは、すべて 1 つの同じチーム プロジェクトの下に置いてください。これにより、ビルド スクリプトをカスタマイズする必要性が削減されます。依存関係が存在する場合、プロジェクト参照を使用してその依存関係を定義するか、または共有プロジェクトから各自のプロジェクトに依存関係を分岐します。ファイル参照は、管理しづらいので、避ける必要があります。例外として、依存する側のプロジェクトを並列開発していて、リアルタイムの変更を必要とする場合があります。そのような場合、**ワークスペース マッピング**の使用を検討してみることができます。ただし、依存する側のコードが原因で、変更点の数が多くなりすぎた場合、分離のバッファを作成するための分岐を検討してみてもかまいません。

参考資料

- ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ワークスペースの編集の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

コンピュータ上でのワークスペースを使用したコード変更の分離方法

開発者の場合、2 つのワークスペースを作成することができます。一方には、チームの他のだれかが扱うファイルおよびフォルダへの参照を格納し、もう一方には、分離が必要なファイルおよびフォルダを格納します。このようにファイルを分離すれば、他のどこかで実行されている作業と並列して特定のファイルを開発することができます。たとえばこれを使用して、リスクを伴う保留中の変更の処理やコード レビューを行うことができます。

2 つ目のワークスペースを作成するには

1. [ソース管理エクスプローラ] で、[**ワークスペース**] ボックスをクリックし、[**ワークスペース**] をクリックします。
2. [**ワークスペースの管理**] ダイアログ ボックスで、[**追加**] をクリックします。
3. [**ワークスペースの追加**] ダイアログ ボックスで、**MyIsolatedWork** といった新しいワークスペース名を入力し、今後の使用に備えてワークスペースの用途に関するコメントを記入します。
4. [**作業フォルダ**] リストで、ワークプレースの状態を [**アクティブ**] に設定し、ワークプレースに組み入れるソース管理フォルダ (これは、チーム プロジェクトのルート フォルダまたは他の任意のサブフォルダでかまいません) を指定し、次に、ワークスペースから取り込んだファイルを収容するコンピュータ上のローカル フォルダ パスを指定します。
5. [**OK**] をクリックしてから [**閉じる**] をクリックして、分離ワークスペースを作成します。

分離ワークスペースでの作業の開始のための一連の最新ソースを取り出すには

1. [ソース管理エクスプローラ] で、分離ワークスペース名が [**ワークスペース**] ボックスで選択されていることを確認します。
2. チーム プロジェクトのルート フォルダ (ただし、ソース ツリーの一部しか必要ない場合はサブフォルダ) を選択して右クリックしてから、[**最新バージョンの取得**] をクリックします。これにより、ソース管理サーバーから、新しいワークスペースにマップしたコンピュータ上のローカル ディレクトリに、フォルダ構造と最新のファイル セットがコピーされます。
3.

参考資料

- ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ワークスペースの編集の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

セキュリティ

- **開発者のワークステーションと TFS の間のチャネルを保護する方法**

開発者のワークステーションと *TFS* の間のチャネルを保護する方法

開発者のワークステーションと TFS の間のチャネルを保護するために、HTTPS と SSL (Secure Sockets Layer: セキュア ソケット レイヤー) 暗号化を使用するように TFS を構成することができます。HTTP 接続を使用不可にして、HTTPS と SSL のみを使用するように TFS を構成することもできます。そのためには、まず、HTTPS と SSL が使用可能になるように TFS を構成してから、HTTPS と SSL に必要な追加のステップを実行する必要があります。

HTTPS と SSL を使用すると、チーム プロジェクトのポータル、レポート、および作業項目を含め、Team Foundation Server の Web リソースへのアクセスを要求する Team Foundation Client と TFS の間のネットワーク トラフィックが暗号化されます。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/aa395265(VS.80).aspx の「HTTPS と Secure Sockets Layer (SSL) を使用した Team Foundation Server のセキュリティ保護」を参照してください。
- SSL (Secure Socket Layer) を使って TFS をセットアップする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms242875(VS.80).aspx の「チュートリアル :

SSL (Secure Sockets Layer) を使用する Team Foundation Server のセットアップ」を参照してください。

- HTTPS および SSL のみに対応するように TFS を構成する方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/aa395285(VS.80).aspx の「方法 ：HTTPS と SSL だけを使用するように Team Foundation Server を構成する」を参照してください。

シェルビング

- **シェルビングを使用した、保留中の作業のバックアップ方法**
- **シェルビングを使用した、チーム メンバとのコードの共有方法**

シェルビングを使用した、保留中の作業のバックアップ方法

保留中の変更をサーバーにバックアップするには、チェックインの準備がまだできていないファイルをシェルビングします。これにより、ビルドの不安定性を招く可能性のある部分的に完了した作業をチェックインしなくても、ソースは確実にサーバーにアップロードされます。

一連の保留中の変更をシェルビングするには

1. [ソリューション エクスプローラ] でソリューションを右クリックしてから、[**保留中の変更を表示**] をクリックして、保留中の変更を表示します。
2. シェルビングしたいファイルを選択してから、[**シェルブ**] をクリックします。
3. シェルブセット名と、シェルブセットの用途を示すコメントを入力してから、[**シェルブ**] をクリックします。

その次の日に作業を復元するには

1. [**ファイル**] メニューで、[**ソース管理**] を指して、[アンシェルブ] をクリックします。
2. 目的のシェルブセットを選択し、[**アンシェルブ**] をクリックします。
   シェルビングされたリビジョンが、ワークスペース内で既に保留中になっている変更と競合しない限り、シェルビングされた各リビジョンは、Team Foundation Server によって保留中の変更と

して宛先ワークスペースに復元されます。

参考資料

- 保留中の変更のシェルビングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

シェルビングを使用した、チーム メンバとのコードの共有方法

チーム メンバと共有するためにソース コードをシェルビングするには、[**最新バージョンの取得**] 操作を実行し、ワークスペースを最新のサーバー バージョンに同期してから、確実にコンパイルが行われるように、アプリケーションをビルドします。[ソース管理エクスプローラ] を使用してソースをシェルビングします。コードのハンドオフ先のチーム メンバは、その後、コードをアンシェルブする必要があります。

シェルビングが便利なのは、別のチーム メンバによって完了される予定の進行中の作業に携わっている場合です。そのようなケースでは、簡単にハンドオフできるように、変更をシェルビングすることができます。最新のコードを同期することによって、ワークスペース外で行われたソース ファイルの変更を組み入れる機会が与えられます。

ソース管理エクスプローラからフォルダおよびファイルをシェルビングするには

1. ソース管理エクスプローラで、右クリックしてから、[**保留中の変更をシェルブ**] をクリックします。
2. [**シェルブ - ソース ファイル**] ダイアログ ボックスの [**シェルブセット名**] ボックスに、たとえば **shelvetest** といったシェルブセット名を入力します。
3. [**コメント**] ボックスに **Testing my shelveset** と入力し、[**シェルブ**] をクリックします。

ファイルとフォルダがソース管理サーバーにコピーされて、他のチーム メンバがアンシェルブで使用できるようになります。

他のチーム メンバがシェルブセットをアンシェルブした場合、シェルビングされたリビジョンが、ワークスペース内で既に保留中になっている変更と競合しない限り、シェルビングされた各リビジョンは、TFS によって保留中の変更として宛先ワークスペースに復元されます。

一連の保留中の変更をアンシェルビングするには

1. Visual Studio 2005 Team System で、[**ファイル**] メニューをクリックし、[**ソース管理**] をポイントして、 [**アンシェルブ**] をクリックします。
2. [**所有者の名前**] ボックスにシェルブセットの作成者の名前 (たとえば ADVENTUREWORKS¥JuanGo または単純に juango) を入力してから、[**検索**] をクリックします。
3. [結果] ペインで、ワークスペースにアンシェルブするシェルブセットを選択してから、[**詳細**] をクリックします。
4. シェルブセットをソース管理サーバーから削除したい場合、[**シェルブセットをサーバーに保持する**] オプションをクリアします。
5. (オプション) 復元されるシェルブセットに作業項目とチェックイン メモを関連付けたくない場合は、[**作業項目およびチェックイン メモの復元**] オプションをクリアします。
6. [**詳細**] ダイアログ ボックスが表示されたら、ワークスペースにアンシェルブするシェルブセットまたはシェルブセット項目を選択してから、[**アンシェルブ**] をクリックします。

参考資料

- 詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

ソース管理の資料

- Team Foundation Server のソース管理の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181237(VS.80).aspx の「Team Foundation ソース管理」を参照してください。

質問と回答: Team Foundation Server のソース管理とバージョニング

索引

バージョン管理へのアクセス

- *MSSCCI Provider* とは何ですか、いつ使用するのですか?
- *TFS* をサポートしている 他の *IDE* にはどのようなものがありますか?
- *Team Foundation Server Power Tool* はいつ使用すればよいですか?
- バージョン管理の拡張性に関する最も一般的なシナリオはどのようなものですか?
- コマンドラインからバージョン管理をどのように操作すればよいですか?

管理

- 継承しているアクセス許可を持つフォルダ内のファイルには、どのようにしてアクセス許可を付与すればよいですか?
- 開発者がプロジェクトを辞めてしまった場合はどうすればよいですか?
- チェックインを実行することについて信頼していないインターンまたは他の開発者は、どのように管理すればよいですか?
- アプリケーションを出荷した後でアクセス許可を修正するには、どのようにすればよいですか?

分岐/ラベル/マージ

- ラベルはいつ使用すればよいですか?
- *TFS* のラベルは *VSS* のラベルとどのように違いますか?
- 分岐とは何ですか?
- 分岐はいつ検討すればよいですか?
- 分岐しない理由は何ですか?
- アプリケーションをリリースするために分岐をどのように使用すればよいですか?

- アプリケーションを保守するために分岐をどのように使用すればよいですか?
- チーム間の競合を減らすために分岐をどのように使用すればよいですか?
- 機能間の競合を減らすために分岐をどのように使用すればよいですか?
- 分岐およびマージに対して実証されているプラクティスには、どのようなものがありますか?
- 分岐とラベルの違いは何ですか?
- *"path space"* の分岐モデルとは何ですか?
- *TFS* のプロモーション モデルはどのように動作しますか?
- *2* つの分岐をどのようにマージしますか?
- 複数のチーム プロジェクトをまたがってマージできますか?
- 基点のないマージとは何ですか?
- コード プロモーション モデルとは何ですか?
- 分岐の論理ビューと物理ビューの違いは何ですか?
- コード プロモーション モデルを使用している場合、どのくらいの頻度でマージすればよいですか?

チェックインおよびチェックイン ポリシー

- 変更セットとは何ですか?
- チェックイン ポリシーとは何ですか?
- チェックイン ポリシーはいつ、どのようにオーバーライドできますか?
- どのようにしてポリシーを強制するのですか?
- チェックインの検証システムをどのように使用すればよいですか?
- ディスク上のファイル名の修正、またはファイルの削除を行った場合、バージョン管理では同期をとれませんか?
- 自動的な競合解決はどのように動作しますか?
- 競合を手動で解決するにはどのようにすればよいですか?
- どのようにして競合を回避すればよいですか?

チェックアウト、取得、およびロック

- ファイルを最後に修正した開発者を見つけるにはどのようにすればよいですか?
- *get* コマンドはどのように機能しますか?
- 共有チェックアウトと排他チェックアウトの違いは何ですか?
- *lock* コマンドはいつ使用すればよいですか?
- *TFS* はどのようなタイプのロックをサポートしていますか?

分散/リモート開発

- オフラインでどのように作業すればよいですか?
- 分散したチーム開発を最適化するにはどのようにすればよいですか?
- *TFS* のバージョン管理プロキシとは何ですか?
- *TFS* のバージョン管理プロキシのパフォーマンスを最適化するにはどのようにすればよいですか?

移行

- *TFS* のバージョン管理は *VSS* とどのように違いますか?
- チェックアウトのモデルは *VSS* とどのように違いますか?
- *VSS* から *TFS* へソースをどのように移行すればよいですか?
- 他のバージョン管理システムからソースをどのように移行すればよいですか?

プロジェクト/ワークスペースの管理

- チーム プロジェクトはどのように編成すればよいですか?
- プロジェクト間の依存関係はどのように管理すればよいですか?
- ワークスペースとは何ですか?
- 開発者を分離するためにワークスペースをどのように使用すればよいですか?
- ワークスペースをマッピングするために実証されているプラクティスにはどのようなものがありますか?
- 共有コンポーネントおよびコードを管理するために実証されているプラクティスにはどのようなものがありますか?

- 新しいチーム プロジェクトまたは新しい分岐はいつ作成すればよいですか?
- 複数のプロジェクト間で共有しているソース コードはどのように管理すればよいですか?
- 複数のプロジェクト間で共有しているバイナリはどのように管理すればよいですか?
- ソース ツリーはどのように編成すればよいですか?

シェルビング

- シェルビングとは何ですか?
- シェルブセットとは何ですか?
- 一般的にシェルビングはいつ使用すればよいですか?
- シェルビングをどのように使用して作業をバックアップすればよいですか?
- シェルブセットをアンシェルブしたくなる理由は何ですか?

バージョン管理へのアクセス

- **MSSCCI Provider とは何ですか、いつ使用するのですか?**
- **TFS をサポートしている 他の IDE にはどのようなものがありますか?**
- **Team Foundation Server Power Tool はいつ使用すればよいですか?**
- **バージョン管理の拡張性に関する最も一般的なシナリオはどのようなものですか?**
- **コマンドラインからバージョン管理をどのように操作すればよいですか?**

MSSCCI Provider とは何ですか、いつ使用するのですか?

Microsoft® Source Code Control Interface (MSSCCI) プロバイダを使用して、Microsoft Visual Studio® チーム エクスプローラをサポートしていない製品に対して、統合されたバージョン管理のユーザー エクスペリエンスを提供します。たとえば、Visual Studio 6.0 を使用している場合は、MSSCCI クライアントまたはコマンドラインを使用して Microsoft Visual Studio Team System (VSTS) Team Foundation バージョン管理とやりとりできます。

次のクライアントは、MSSCCI プロバイダを使用して、Team Foundation バージョン管理を直接使用することができます。

- 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

MSSCCI プロバイダは、Visual Studio 2005 の Team Foundation のバージョン管理とは次の点で動作が異なります。

- チェックアウトにより**最新バージョンの取得**の操作も実行する。
- チェックアウト時に排他チェックイン ロックが適用される。
- **[ソース管理から開く]** および **[ソース管理へ保存]** のメニュー オプションは、Microsoft Visual SourceSafe® と同様の動作をする。

参考資料

- Microsoft MSDN® の MSSCCI の詳細については、「http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcug98/html/_asug_the_microsoft_source_code_control_interface.asp の「The Microsoft Source-Code Control Interface」を参照してください。
- MSSCCI Provider の詳細については、http://blogs.msdn.com/bharry/archive/2006/03/24/559876.aspx の「Update on the TFS MSSCCI Provider」を参照してください。
- MSSCCI のアドインは、TFS Power Tool で開発されていますが、Microsoft では正式にサポートされていません。このツールを MSDN からダウンロードするには、http://www.microsoft.com/downloads/details.aspx?FamilyId=87E1FFBD-A484-4C3A-8776-D560AB1E6198&displaylang=en へアクセスしてください。

TFS をサポートしている 他の *IDE* にはどのようなものがありますか*?*

Team Foundation Server は、チーム エクスプローラがインストールされている Visual Studio 2005 のすべてのエディションから使用できます。チーム エクスプローラを、Visual Studio 2005 以外の統合開発環境 (IDE) と共に実行して、チーム プロジェクトで作業し、作業項目を管理することもできます。

次のクライアントは、他のベンダーから提供されている統合ソリューションを備えています。

- Eclipse
- Linux クライアント
- Apple Macintosh クライアント
- Hypertext Markup Language (HTML) Web クライアント

Eclipse IDE、Linux、または Macintosh のクライアントから Team Foundation のバージョン管理にアクセスしたい場合は、クライアント アプリケーションの Teamprise スイートをインストールすることを検討してください。これは http://www.teamprise.com/ で入手できます。

Web から Team Foundation のバージョン管理に読み取り専用でアクセスしたい場合は、Team System Web Access をインストールすることを検討してください。これは http://msdn2.microsoft.com/en-us/teamsystem/bb676728.aspx で入手できます。

参考資料

- チーム エクスプローラの使用の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms242912(vs.80).aspx の「以前のバージョンの Visual Studio プロジェクトまたはその他のコード プロジェクトの操作」を参照してください。
- Teamprise の詳細については、http://www.teamprise.com を参照してください。
- Team System Web Access の詳細については、

http://msdn2.microsoft.com/en-us/teamsystem/bb676728.aspx を参照してください。

Team Foundation Server Power Tool はいつ使用すればよいですか*?*

Team Foundation Power Tool (TFPT) は、Visual Studio 2005 のユーザー インターフェイス (UI) から利用できないバージョン管理機能を提供します。TFPT は、たとえばオフラインの作業に便利です。また、変更セットのチェックインを元に戻す手段として、TFPT を使用してロールバック処理を実行することもできます。次のいずれかの操作を実行しなければならない場合は、TFPT の使用を検討してください。

- **アンシェルブ – TFS** でサポートされている**アンシェルブ**の操作では、シェルブされた変更とローカルな変更を一緒にマージすることはできません。保留中の変更 (編集) がローカル ワークスペース内の項目に含まれていて、かつシェルブされた変更も編集である場合は、TFPT を使用して変更セットから変更をアンシェルブすることにより、3 方向マージを用いて TFPT で各変更をマージすることができます。
- **ロールバック** – TFS は、変更セットのチェックインを元に戻す機能には直接は対応していません。TFPT の **rollback** コマンドを使用すると、特定の変更セットで行われた変更の大半は元に戻すことができます。すべての変更をロールバックできるわけではありませんが、ほとんどのシナリオでロールバックが動作します。
- **オフライン作業** – TFPT のオンライン ツールでは、ある一定期間、サーバーに接続しなくても作業することができます。これは、ローカル ワークスペースに対して行った変更をサーバーに通知する機能によって実現されます。
- **変更セットの取得 – TFPT** の **GetCS** コマンドを実行すると、変更セットの当該バージョンの時点で変更セットに列挙されていた項目がすべて取得することができます。これは、自分のワークスペースに必要な変更が既に同僚によってチェックインされていた場合は便利ですが、ワークスペース全体を最新バージョンに更新することはできません。
- **保留中の変更の解除** – TFPT の **UU** (Undo Unchanged) コマンドを実行すると、実際にはまだ編集されていない保留中の編集がファイルから削除されます。編集用にファイルを大量にチェックアウトしておいて、実際には少数のファイルしか変更しないような場合は、この方法が便利です。TFPT の **UU** コマンドを実行することにより、まだ変更されていないファイル上での編集状態を取り消すことができます。このコマンドは、ローカル ワークスペースに含まれているファイルの

ハッシュと、サーバー上にあるハッシュとを比較して、ファイルが実際に編集されたのかどうかを判定するコマンドです。

これらの各コマンドは、Tfpt.exe を使用してコマンド ラインから実行します。

参考資料

- TFPT をダウンロードするときは、http://www.microsoft.com/downloads/details.aspx?FamilyID=7324c3db-658d-441b-8522-689c557d0a79&DisplayLang=en へアクセスしてください。
- TFPT について意見を交わしているフォーラムを見たい場合は、http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1 にアクセスしてください。

バージョン管理の拡張性に関する最も一般的なシナリオはどのようなものですか?

バージョン管理の拡張性における最も一般的なシナリオは、チェックイン時の基準を強化するためにチェックイン ポリシーをカスタマイズすることです。**[チェックイン ポリシーの追加]** ダイアログ ボックスに示されるカスタム ポリシーのプラグインを作成するには、Visual Studio Team Foundation Server の Software Development Kit (SDK) で提供される拡張性機能を使用します。TFS の SDK は、http://go.microsoft.com/fwlink/?linkid=68586 からダウンロードできます。

一般的ではありませんが、Visual Studio 2005 以外のクライアントが Team Foundation のバージョン管理と作業できるようにするための統合レイヤーを記述することもできます。

参考資料

- チェックイン ポリシーのカスタマイズ方法を学習するには、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。

- 特定のパターンのチェックインを禁止するサンプル コードの詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインでのコメントを義務付けるサンプル コードの詳細については、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーを登録する方法の詳細については、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。
- TFS の SDK をダウンロードするには、http://go.microsoft.com/fwlink/?linkid=68586 へアクセスしてください。
- Team Foundation のバージョン管理の拡張性について詳しく学習するには、http://msdn2.microsoft.com/en-us/library/bb187335(VS.80).aspx の「Walkthru: The Version Control Object Model」を参照してください。

コマンドラインからバージョン管理をどのように操作すればよいですか?

Team Foundation Server には、ソース管理の操作を実行するためのコマンドライン ツール (Tf.exe) が用意されています。たとえば、このコマンドラインを使用して Microsoft Windows® Task Scheduler を使用し、操作をスケジュールすることができます。

適切なパスおよび他の環境変数を確実にセットアップするには、Visual Studio 2005 の [コマンド プロンプト] ウィンドウからツールを実行するか、または Vsvars32 バッチ ファイルを実行します。このバッチ ファイルは通常、ドライブ文字:¥Program Files¥Microsoft Visual Studio 8¥Common7¥Tools にあります。このコマンドラインは、**Checkin**、**Checkout**、**Get**、**History**、**Shelve**、**Branch**、**Merge**、**Label**、**Status**、**Undelete**、**Undo** などソース管理のほとんどのコマンドをサポートしています。

次に、コマンドラインから実行する一般的な操作を示します。

- サーバーからローカル マシンにファイルを同期する – **tf get**

- ファイルをサーバーへ追加する – **tf add**
- 編集するためのファイルをチェックアウトする – **tf checkout**
- 保留中の変更をチェックインする – **tf checkin**
- 特定の変更セットをサーバーから取得する – **tf get /version**

次の操作は、コマンドラインでのみ実行できます。

- 他のユーザーのワークスペースを削除する – **tf workspace /delete**
- 他のユーザーのチェックインを元に戻す – **tf undo**
- 他のユーザーのロックを解除する – **tf lock**
- ラベルの範囲を定義する – **tf label**
- 基点のないマージを実行する – **tf merge**

参考資料

- Tf.exe コマンドの操作の詳細については、http://msdn2.microsoft.com/ja-jp/library/zthc5x3f(VS.80).aspx の「チュートリアル : コマンドラインからの Team Foundation ソース管理の操作」を参照してください。
- コマンドラインのみで使用できるコマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194957(VS.80).aspx の「コマンド ラインからのみ実行できる操作 (Team Foundation ソース管理)」を参照してください。

管理

- **継承しているアクセス許可を持つフォルダ内のファイルには、どのようにしてアクセス許可を付与すればよいですか?**
- **開発者がプロジェクトを辞めてしまった場合はどうすればよいですか?**
- **チェックインを実行することについて信頼していないインターンまたは他の開発者は、どのように管理すればよいですか?**
- **アプリケーションを出荷した後でアクセス許可を修正するには、どのようにすればよいですか?**

継承しているアクセス許可を持つフォルダ内のファイルには、どのようにしてアクセス許可を付与すればよいですか?

アクセス許可を設定するには、ソース管理エクスプローラでフォルダまたはファイルを右クリックして **[プロパティ]** をクリックし、**[セキュリティ]** タブで、アクセス許可を変更したいユーザーまたはグループを選択して、**[アクセス許可]** にリストされているアクセス許可を編集します。これらのアクセス許可は、TF コマンドライン ユーティリティの **Permission** コマンドを使用して、コマンドラインで設定することもできます。

Team Foundation のソース管理では、Windows グループ、Windows ユーザー、および Team Foundation グループに対してアクセス許可を付与することができます。アクセス許可は、上位フォルダから継承することができます。また、アクセス許可を明示的に宣言することもできます。

アクセス許可の設定は、Allow または Deny のいずれかの形式になります。Deny が継承されており、Grant が明示的に定義されている場合でも、Deny は常に Grant を上書きします。継承されたアクセス許可と明示的なアクセス許可が組み合わされて、項目についてのユーザーまたはグループの有効なアクセス許可が決定されます。Deny は常に Allow を上書きするため、継承を常に有効にしてチェックインを拒否する、といったことができます。

継承をオフにする場合は注意が必要です。たとえば、継承をオフにする前に、必要な明示的なアクセス許可を最初に設定し、自分のアカウントに特有のアクセス許可を割り当てるようにします。対象のアクセス許可を設定せずに継承をオフにすると、自身がファイルまたはフォルダから締め出され、アプリケーション層のコンピュータの管理者でもある、TFS 管理者にアクセス許可を修正してもらうことが必要になります。アプリケーション層のローカル管理者は、設計上、自身を完全に締め出すことはありません。

参考資料

- アクセス許可の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms253077(VS.80).aspx の「Team Foundation Server の既定のグループ、アクセス許可、およびロール」を参照してください。

- アクセス許可について、さらに詳しい情報は、http://msdn2.microsoft.com/ja-jp/library/ms181761(VS.80).aspx の「ソース管理のセキュリティ権限およびアクセス許可」を参照してください。

開発者がプロジェクトを辞めてしまった場合はどうすればよいですか?

開発者がプロジェクトを辞めてしまった場合は、その開発者のワークスペースを必ず削除してください。この操作により、その開発者の保留中の変更もすべて削除され、開発者が保持していたすべてのロックも元に戻されます。

注意:チーム プロジェクトに対して排他ロックがオンになっている場合は、変更を元に戻さずにロックを元に戻すことはできません。

開発者がロックしていたファイルを見つけるには、次のコマンドを実行します。

tf workspaces /owner:domain¥devuser /computer:* /server:servername

ワークスペースを削除し、ロックを除去するには、次のコマンドを実行します。

Tf workspace /delete workspacename;domain¥devuser /s:servername

参考資料

- 開発者がプロジェクトを辞めた後のクリーンアップの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms194958(VS.80).aspx の「方法 : チームを離れるメンバのファイルをクリーンアップする」を参照してください。
- **Workspace** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/y901w7se(VS.80).aspx の「Workspace コマンド」を参照してください。

チェックインを実行することについて信頼していない研修生または他の開発者は、どのように管理すればよいですか?

新人や研修生のように、まだ確固とした信頼の置けない開発者には、ソース ツリーのチェックイン権限を利用させないことができます。引き継いだ権限をオフにする前に、自身のアカウントに関する権限をはじめとして必要な権限は必ず設定するようにしてください。対象となる開発者は、直接チェックインはできなくても、保留中の変更は可能ですし、そうした変更をシェルブすることもできます。もっと経験を積んだ開発者であれば、そうした変更をアンシェルブして、内容をよく調べて、チェックインすることができます。

参考資料

- 権限を削除の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400718(VS.80).aspx の「方法 : ソース管理対象のファイルへのアクセスを削除する」を参照してください。

アプリケーションを出荷した後でアクセス許可を修正するには、どのようにすればよいですか?

出荷の後など、分岐がメンテナンス モードになっている場合は、アクセス許可の継承をオフにして、ツリーをロックすることができます。アクセス許可をオフにした後、ホットフィックスの必要に応じて、個々のユーザーに**「更新しない」**および**「チェックイン」**のアクセス許可を定義することができます。

参考資料

- アクセス許可の削除の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms400718(VS.80).aspx の「方法 : ソース管理対象のファイルへのアクセスを削除する」を参照してください。

分岐/ラベル/マージ

- **ラベルはいつ使用すればよいですか?**
- **TFS のラベルは VSS のラベルとどのように違いますか?**
- **分岐とは何ですか?**
- **分岐はいつ検討すればよいですか?**
- **分岐しない理由は何ですか?**
- **アプリケーションをリリースするために分岐をどのように使用すればよいですか?**
- **アプリケーションを保守するために分岐をどのように使用すればよいですか?**
- **チーム間の競合を減らすために分岐をどのように使用すればよいですか?**
- **機能間の競合を減らすために分岐をどのように使用すればよいですか?**
- **分岐およびマージに対して実証されているプラクティスには、どのようなものがありますか?**
- **分岐とラベルの違いは何ですか?**
- **"path space" の分岐モデルとは何ですか?**
- **TFS のプロモーション モデルはどのように動作しますか?**
- **2 つの分岐をどのようにマージしますか?**
- **複数のチーム プロジェクトをまたがってマージできますか?**
- **基点のないマージとは何ですか?**
- **コード プロモーション モデルとは何ですか?**
- **分岐の論理ビューと物理ビューの違いは何ですか?**
- **コード プロモーション モデルを使用している場合、どのくらいの頻度でマージすればよいですか?**

ラベルはいつ使用すればよいですか?

後の操作のために、特定のファイルとフォルダをまとめてグループ化する場合にラベルを使用します。ラベルは、分岐、マージ、差分抽出、ファイルの取得に対して使用できます。ラベルは、これらのいずれかの操作を実行するときに、後で戻ることができる目印を付けます。

Team Foundation のビルドは、作成する各ビルドに関連付けられているファイル バージョンに自動的にラベルを付けます。

注意:分岐が必要かどうかわからない場合は、ファイル セットにラベルを付けて、後でそのラベルを基にして分岐を作成することができます。

参考資料

- ラベルの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181439(VS.80).aspx の「ラベルの操作」を参照してください。
- ラベルについて、さらに詳しい情報は、http://msdn2.microsoft.com/ja-jp/library/ms181440(VS.80).aspx の「方法 : ラベルを適用する」を参照してください。

TFS のラベルは VSS のラベルとどのように違いますか?

Team Foundation Server のラベルは VSS のラベルと大きく異なります。VSS のラベルは、通常は VSS のすべてまたは一部のツリーに対して割り当てる「一時点」を表すラベルなので、VSS は、時系列に従ってファイル履歴とともにラベルを表示します。ラベル付けされているファイルの前にリストに表示されているものはすべて、ラベルに含まれていますが、このリストの後のものはすべて含まれません。

TFS では、タイミングを使用する代わりに、ラベルによって、一連のソース ファイルの特定のバージョン同士を結びつけます。ラベルを使用する一般的なシナリオの 1 つは、デイリー ビルドの目印に使用することです。これによって、修正を適用する必要がある場合など、特定のビルドに対応するソース ファイル セットを簡単に取得することができます。

参考資料

- TFS と VSS のラベルの比較の詳細については、http://blogs.vertigosoftware.com/teamsystem/archive/2006/05/03/Comparing_SourceSafe_Labels_to_Team_Foundation_Server_Labels.aspx の「Comparing SourceSafe Labels to Team Foundation Server Labels」を参照してください。

分岐とは何ですか?

分岐 (フォークとも呼ばれる) は、ファイルの集まりを個々の開発パスに分けることです。Team Foundation Server は、分岐および高度なマージをサポートしており、これによって、個々の分岐のファイルを結合することができます。たとえば、分岐を使用して、アプリケーションのメジャー リリースを分離することができます。リリースされたアプリケーションのバージョンを分岐したら、後でそれをより簡単に保守することができます。マージによって、両方の分岐に対して選択的に修正を行うことができます。

分岐およびマージの目的は、並行する開発の流れを分離することです。たとえば、チームで将来的に維持したいビルドを作成したときに、分岐を作成します。また、非常に規模の大きい開発組織で機能チームを分離するために分岐を選択したり、アプリケーションで同時に複数のバージョンの開発をサポートしたりする場合にも、分岐を選択できます。

TFS のバージョン管理では、分岐のときにファイル コンテンツの別のコピーを作成しないため、ソース管理データベースで、多量の追加スペースを消費しません。分岐は、差分リストのソース コンテンツを指しているデータベース内のポインタから構成されます。この差分によって、ベースとなるバージョンからソース コンテンツを変更します。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

分岐はいつ検討すればよいですか?

分岐を使用すると、複数の開発者が同じファイル上で個別に作業することができます。マージにはオーバーヘッドが伴い、競合を管理しなければならないため、分岐によるファイルの分離が必要ない場合は分岐しないでください。ビルドにラベルを付けて、後で必要なときに分岐することができます。

分岐を作成するかどうかの判断は、競合をリアルタイムでマージするコストの方が高いか、または分岐間の競合をマージするための総コストより高いかどうか、という点に集約することができます。

分岐を行う一般的な理由には、次のものがあります。

- **リリース** – 保守したいビルドに対して分岐するか、または並行している複数のリリースに対して分岐します。
- **保守** – 以前リリースしたビルドを保守するために分岐します。
- **機能** –プロジェクトの他の部分を不安定にさせる可能性のある、試行的な機能またはリスクの高い機能における作業を分離します。プロジェクトの他の部分を不安定にさせるような、インターフェイスの変更作業を分離します。
- **チーム** – サブチームを分離し、他のチームの誤った変更の影響を受けずに作業できるようにします。サブチームを分離して、独自のマイルストーンを目標として作業できるようにします。チームの分岐は、機能の分岐に非常に似ています。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation

ソース管理の分岐およびマージ」を参照してください。

分岐しない理由は何ですか?

開発チームのメンバーが、同じファイルで並行して作業する必要がない場合は、分岐はしないでください。迷った場合は、ビルドにラベルを付けておいて、後でそのビルドから分岐を作成してもかまいません。分岐をマージするのはたいへんに手間のかかることがあります。分岐どうしの間の変更が大きい場合は特にそうです。

マージを行うためには、1 人以上の開発者がそのマージを実行して競合を解決する必要があります。ビルドを不安定にさせるような、不適切なマージを行ってしまうことも多いため、マージしたソースは完全にテストする必要があります。

分岐階層越しにマージするのは特に難しく、普通なら自動的に処理される競合の多くを手動で処理しなければなりません。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

アプリケーションをリリースするために分岐をどのように使用すればよいですか?

リリース前にビルドを安定させる準備ができている場合は、Release 分岐を使用します。

Release 分岐を作成した後、分岐構造がどのように見えるかについて、次に例を示します。

- **Main** – 統合分岐
    - **Source**
    - **Other asset folders**
- **Release 1** – リリース分岐
    - **Source**

Release 分岐で作業する場合には、次の推奨事項に留意してください。

- **分岐すべきとき:** リリースの準備ができたときに、すべてのものを Main 分岐に統合し、その後で Release 分岐を作成します。これを使用して、リリースの前にアプリケーションを安定させることができます。
- **分岐すべきでないとき:** リリースごとに 1 つの TFS プロジェクトを使用している場合は、現在のプロジェクト内の分岐の代わりに、新しいプロジェクトを使用して開発を継続することができます。
- **分岐におけるアクセス許可:**
    - リリース前 – すべての開発者に対して読み込み/書き込み
    - リリースの後 – ホットフィックスの作業をしている開発者に対しては読み込み/書き込み、その他の開発者に対しては読み込み専用
- **分岐におけるビルドの頻度:** 要求に基づいたビルド。
- **分岐におけるテストの焦点:** リリースの承認。

リリース前のビルドを安定させるために必要な修正および変更に対しては Release 分岐を使用します。アプリケーションの次のバージョンの開発は、メインの Development または Integration 分岐の中で並行して行うことができ、リリース前に行った安定化のための変更の恩恵も新しいバージョンが受けることができます。最終的なリリースのビルドが作成されたら、Release 分岐からメインの Development および Integration 分岐へ、変更をマージして戻すことができます。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

アプリケーションを保守するために分岐をどのように使用すればよいですか?

以前リリースした分岐をサポートするには、Maintenance 分岐を使用します。

リリースを Maintenance フォルダへ移動した後に、分岐構造がどのように見えるかについて、次に例を示します。

- **Main** – 統合分岐
    - **Source**
    - **Other asset folders**
- **Releases** – Release 分岐のコンテナ
    - **Release 1** – Maintenance 分岐
        - **Source**

Maintenance 分岐で作業する場合には、次の推奨事項に留意してください。

- **分岐すべきとき:** リリースの後で、Maintenance フォルダ内の分岐でリリースをサポートします。

- **分岐すべきでないとき:** 将来リリースを保守する必要がない場合は、ラベルを使用して、リリースされた古いビルドに印を付けて、Main 分岐で継続して作業することができます。
- **分岐におけるアクセス許可:**
    - ホットフィックスで作業している開発者に対して読み込み/書き込み。
    - その他の開発者に対しては読み込み専用。
- **分岐におけるビルドの頻度:** 要求に基づいたビルド。
- **分岐におけるテストの焦点:** リリースの承認。

アプリケーションの古いバージョンをサポートするには、Maintenance 分岐を使用します。これらの変更をメインの Integration 分岐にマージするか、または Maintenance 分岐に固有の変更のままにしておくか、選択できます。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

チーム間の競合を減らすために分岐をどのように使用すればよいですか?

Team 分岐を使用して、複数のチーム間で分断されている変更の統合および安定性を改善します。

Team 分岐を作成した後で、分岐構造がどのように見えるかについて、次に例を示します。

- **Development -** Team 分岐のコンテナ
    - **Team 1 -** Team 分岐
        - **Source**
    - **Team 2 -** Team 分岐
        - **Source**
- **Main** – 統合分岐
    - **Source**
    - **Other Asset Folders**

Release 分岐で作業する場合には、次の推奨事項に留意してください。

- **分岐すべきとき:** 開発チームの作業が、ファイルおよびコンポーネントで重複する場合は、Team 分岐を使用して、各チームの作業を分離します。Team 分岐の下で Feature 分岐を作成してもよいでしょう。
- **分岐すべきでないとき:** ソース ツリーがコンポーネントごとに整理されており、誤ったインターフェイスの変更がない、またはチーム間で大量のチェックインの競合がないと確信している場合は、おそらくチームの分岐は必要ないでしょう。
- **分岐におけるアクセス許可:**
    - チームの開発者に対して読み込み/書き込み。
    - その他の開発者に対しては読み込み専用。
- **分岐におけるビルドの頻度:** 継続的な統合 (CI) ビルド
- **分岐におけるテストの焦点:** 機能テストおよび迅速なフィードバックのテスト。

チームで完全な開発タスクを並行して行えるようにするには、Team 分岐を使用します。これは、他のチームに起因する誤った変更からチームを分離したり、チームが独自のマイルストーンに向けて作業したりする場合に有用です。アクティブなすべての開発は Team 分岐から発生し、Main 分岐へ統合されるようにします。各チームの分岐下で Feature 分岐を作成してもよいでしょう。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

機能間の競合を減らすために分岐をどのように使用すればよいですか?

Feature 分岐を使用して、機能間で分断されている変更の統合および安定性を改善します。

Feature 分岐を作成した後で、分岐構造がどのように見えるかについて、次に例を示します。

- **Development -** Feature 分岐のコンテナ
    - **Feature A –** Feature 分岐
        - **Source**
    - **Feature B –** Feature 分岐
        - **Source**
    - **Feature C –** Feature 分岐
        - **Source**
- **Main** – 統合分岐
    - **Source**
    - **Other Asset Folders**

Release 分岐で作業する場合には、次の推奨事項に留意してください。

- **分岐すべきとき:** 機能チームでは、ソース ファイルが重複することがよくあるため、不具合のあるビルドやビルドの競合が発生する可能性が高くなります。このような問題に直面している場合は、各機能に対して機能を分岐し、機能を分離させることを検討します。小規模なチームの場合は、機能の分岐を Main 分岐から分岐して作成したり、大規模なチームの場合は、機能の分岐を Team 分岐から分岐して作成したりできます。
- **分岐すべきでないとき:** CI ビルドのみを作成している場合、またはデイリー ビルドが既に安定していると予想される場合は、機能の分岐による余分なオーバーヘッドは必要ないでしょう。
- **分岐におけるアクセス許可:**
  - Feature 分岐の開発者に対して読み込み/書き込みのアクセス許可。
  - その他の開発者に対しては読み込み専用。
- **分岐におけるビルドの頻度:** CI ビルド。
- **分岐におけるテストの焦点:** 機能テストおよび迅速なフィードバックのテスト。

各機能が並行に開発されるようにするには、Feature 分岐を使用します。アクティブなすべての開発は Feature 分岐で実行し、コードは Main 分岐へ統合します。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

分岐およびマージに対して実証されているプラクティスには、どのようなものがありますか?

分岐を行う場合には、次のガイドラインを考慮してください。

- 分岐ツリーの構造は、階層越しにではなく階層に沿って (つまり分岐ツリーの上下方向に沿って) だけマージすればよい構造にしてください。階層越しに分岐する場合は、基点のないマージを使用せざるを得ませんが、これは、多くの競合を手動で解決する必要があります。
- 分岐階層が分岐親および分岐子のベースとなります。この分岐階層というのは、ディスク上のソース コードの物理的な構造とは異なる場合があります。マージを計画する場合は、ディスク上の物理構造ではなく、論理的な分岐の構造に留意してください。
- 分岐はあまり深くしないでください。分岐のマージには遅延が伴うため、分岐の構造を深くすると、子分岐の変更をメインの分岐に反映させるのに、非常に時間がかかることがあります。これはプロジェクトのスケジュールにマイナスの影響を及ぼし、バグを修正する時間が長くなります。
- 分岐は上位レベルで行い、その中に構成ファイルおよびソース ファイルを含めるようにします。
- 分岐構造を時間とともに発展させて、ニーズの変化に対応できるようにしてください。
- 開発チームが同じファイル セットで同時に作業する必要がある場合を除き、分岐はしないでください。迷った場合は、ビルドにラベルを付けておいて、後でそのビルドから分岐を作成してもかまいません。分岐をマージするのは非常に手間がかかることがあります。分岐どうしの間の変更が大きい場合は特にそうです。
- マージを行うためには、1 人以上の開発者がそのマージを実行して競合を解決する必要があります。マージの判断を誤ることによって、ビルドが不安定になることは珍しくないため、マージされたソースは徹底的にテストしなければなりません。
- 分岐階層越しにマージするのは特に難しく、普通なら自動的に処理される競合の多くを手動で処理しなければなりません。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx

の「方法 : ファイルおよびフォルダをマージする」を参照してください。

- Visual Studio 2005 における分岐とマージの方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181423(VS.80).aspx の「Team Foundation ソース管理の分岐およびマージ」を参照してください。

分岐とラベルの違いは何ですか?

分岐は、ファイルの集まりを個々の開発パスへ分割する目的で設計されています。分岐を使用して ファイルの 2 つのバージョン セットを分離し、2 つのパスを個別に展開することができます。たとえば、分岐を使用して、アプリケーションの以前リリースされたバージョンを管理および保守できます。また、分岐を使用して、複数の開発者に及ぶ、リスクの高いコード変更を分離し、チームへの影響を最小限にすることができます。ただし、1 人の開発者のみを分離する必要がある場合は、ワークスペースを使用して分岐およびマージの複雑さを軽減します。

ファイルまたはファイルの集まりに明確な表記のタグを付けて、後で簡単に取り出せるようにするには、ラベルを使用します。たとえば、該当するリリースに関する問題に対処したい場合など、ラベルを使用して、デイリー ビルドにマークし、後で取り出せるようにできます。

参考資料

- **branch** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/d73s8b27(VS.80).aspx の「Branch コマンド」を参照してください。
- **label** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/9ew32kd1(VS.80).aspx の「Label コマンド (Team Foundation ソース管理)」を参照してください。

"path space" の分岐モデルとは何ですか?

分岐は、ファイル コピーの操作と類似して、リポジトリのパス構造内に作成されます。これは、Visual

SourceSafe で使用される固定と共有のメカニズムとは異なります。このアプローチにより、ソフトウェアの古いバージョンの管理がより簡単になります。これは、ある分岐から別の分岐へ変更をマージしたり、2 つの分岐で同時に変更を行ったりすることが簡単になるためです。
Team Foundation Server のバージョン管理では、分岐のときにファイル コンテンツの別のコピーを作成しないため、ソース管理データベースで、多量の追加スペースを消費しません。分岐は、差分リストのソース コンテンツを指しているデータベース内のポインタから構成されます。この差分によって、ベースとなるバージョンからソース コンテンツを修正します。

参考資料

- 分岐およびマージの概要については、http://msdn2.microsoft.com/en-us/library/aa730834(VS.80).aspx の「Branching and Merging Primer」を参照してください。

TFS のプロモーション モデルはどのように動作しますか?

プロモーション モデルは、ある分岐から他の分岐へ変更を伝えるための方法です。Visual SourceSafe でも分岐およびマージは可能でしたが、実際には、不完全なプロモーション モデルとして固定と共有を使用していました。Team Foundation Server のバージョン管理では、分岐およびマージを使用してプロモーション モデリングをサポートします。複数の分岐を使用して、アプリケーションにおける作業中の特定のバージョンを分離させることができます。ある分岐のファイルから別の分岐のファイルへマージすることによって、変更を伝播することができます。

参考資料

- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。
- Visual SourceSafe と Team Foundation Server のバージョン管理の違いの詳細については、

http://msmvps.com/blogs/vstsblog/archive/2005/07/02/56401.aspx の「Visual Source Safe vs. Team Foundation Server」を参照してください。

2 つの分岐をどのようにマージしますか*?*

マージを実行するには、ソース管理エクスプローラのマージ機能を使用するか、または **merge** コマンドライン ツールを使用することができます。変更セット、ラベル、日付、またはバージョンをベースにしてマージすることができます。

分岐の構造が複数のレベルになっていることもあります。このような場合には、一度に 1 つのレベルしかマージできないことに留意してください。たとえば、次のソース ツリー構造を考えてください。

- **Development** – 分離した開発分岐のコンテナ
  - **Feature A**
    - **Source**
  - **Feature B**
    - **Source**
- **Main** – メインの統合ビルドの分岐
  - **Source**
  - **Other Asset Folders**
- **Releases** – Release 分岐のコンテナ
  - **Release 1**
    - **Source**
  - **Release 2 –** ロックダウンされている対象のリリース
    - **Source**

Release 2 の分岐に含まれているソース コードが変更されたときに、その変更を最終的に Feature 分岐へマージしたい場合があります。たとえばバグ修正を最新の機能コードに反映させる場合です。**Release 2** から **Feature B** へ直接マージするのは簡単ではありません。代わりに、**Release 2** から論理的な親である **Main** へマージし、次に **Main** から論理的な子である **Feature B** へマージしま

す。直接的な親子関係がない分岐間でマージが必要な場合は、基点のないマージを実行する必要があります。

参考資料

- ファイルとフォルダをマージする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- 基点のないマージの詳細については、http://msdn2.microsoft.com/ja-jp/library/bd6dxhfy(VS.80).aspx の「Merge コマンド」の **/baseless** オプションを参照してください。

複数のチーム プロジェクトをまたがってマージできますか?

はい。複数のチーム プロジェクト間のマージについては特別なことはありません。この場合でも通常のマージを行います。チーム プロジェクトの作成ウィザードで、別のプロジェクトの分岐としてチーム プロジェクトを作成することができます。

参考資料

- マージの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。

基点のないマージとは何ですか?

branch コマンドは、2 つのフォルダ間の関係を作成します。この関連を利用して、分岐間でコードをマージすることができます。マージの対象における直接の子または親ではない分岐には、この関係がありません。MSDN で定義されているように、基点のないマージは、基点となるバージョンなしでマージを実行します。つまり、ユーザーは、分岐/マージの関係を持たないファイルおよびフォルダをマージすることができます。基点のないマージを行うと、マージの関係が確立され、以降のマージは「基点のないマージ」ではなくなります。

基点のないマージでは、通常のマージよりも競合が多くなります。ただし、最初に基点のないマージを行った後は、それらの分岐における以降のマージは「基点のないマージ」ではなくなるため、競合は減ります。

基点のないマージは、コマンドラインでのみ作成できます。最初に基点のないマージを実行し、分岐間で関係が作成された後でも、コマンドラインから以降のマージを作成する必要があります。

参考資料

- ファイルとフォルダをマージする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 ： ファイルおよびフォルダをマージする」を参照してください。
- 基点のないマージ、およびその他のマージのオプションの詳細については、http://msdn2.microsoft.com/ja-jp/library/bd6dxhfy(VS.80).aspx の「Merge コマンド」の **/baseless** オプションを参照してください。

コード プロモーション モデルとは何ですか?

コード プロモーションは、分岐を使用して、いくつかの安定フェーズを通してコードを成長させる方法です。たとえば、チームがアクティブな開発を行う Development 分岐、チームが統合およびテストを行う Main 分岐、およびチームが最終的なリリースの安定のために使用する Production 分岐を持つことができます。

参考資料

- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。

分岐の論理ビューと物理ビューの違いは何ですか?

物理ビューは、次の例に示すように、ソース ツリー内に表示される階層です。

- $/MyTeamProject
  - **Development**
    - Source
  - **Main**
    - Source
  - **Releases**
    - **Release1**
      - Source
    - **Release2**
      - Source

Development、Main、Release1、および Release2 は、ソース コードが直接含まれている分岐です。Releases は複数の分岐が含まれているフォルダです。

論理ビューは、作成されたとおりの分岐親と分岐子の階層です。この階層はソース ツリーには表示されませんが、各分岐がどのように作成されたかに基づいて視覚化することができます。たとえば、次のようになります。

- **Main**
  - **Development**
  - **Release1**
  - **Release2**

論理階層をまたがってマージするよりも、上下方向にマージする方がマージが簡単であるため、論理ビューは重要です。階層をまたがってマージする場合は、基点のないマージを実行しなければなりません。このようなマージでは、コマンドラインの使用、およびより多くの競合の解決が必要になります。

参考資料

- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 ： ファイルとフォルダを分岐する」を参照してください。
- ファイルとフォルダをマージする方法の詳細については、

http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。

- 基点のないマージの詳細については、http://msdn2.microsoft.com/ja-jp/library/bd6dxhfy(VS.80).aspx の「Merge コマンド」の **/baseless** オプションを参照してください。

コード プロモーション モデルを使用している場合、どのくらいの頻度でマージすればよいですか?

マージの頻度は分岐構造によって決まります。たとえば、一般的な分岐構造は次のようになります。

- **Development** – 分離した開発の分岐のコンテナ
  - **Feature A**
    - **Source**
  - **Feature B**
    - **Source**
- **Main** – メインの統合ビルドの分岐
  - **Source**
  - **Other Asset Folders**
- **Releases** – Release 分岐のコンテナ
  - **Release 1**
    - **Source**
  - **Release 2 –** 現在ロックダウンされているリリース
    - **Source**

この場合には、アクティブな開発は Feature 分岐で行われ、2 つのマージの頻度を決定しなければなりません。

1. 機能チームは、Main 分岐から Feature 分岐へどのくらいの頻度で変更をマージすべきでしょうか?
2. 機能チームは、どのくらいの頻度で Main 分岐へ変更をマージすべきでしょうか?

次のマージ頻度は、変更が分断されることが少なくなり、移行時間を短縮するうえで実証されている方針です。

1. 機能チームは、機能が完全になったらすぐに Main 分岐へ変更をマージします。
2. 機能チームは、Main からの変更を定期的にマージして、各チームの機能が完全になったときに、統合互換性が確実に実現されるようにします。理想的には、1 日に 1 回マージして、マージの間隔は 2 日より多く空けないようにします。

参考資料

- ファイルとフォルダをマージする方法の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx の「方法 : ファイルおよびフォルダをマージする」を参照してください。

チェックインおよびチェックイン ポリシー

- **変更セットとは何ですか?**
- **チェックイン ポリシーとは何ですか?**
- **チェックイン ポリシーはいつ、どのようにオーバーライドできますか?**
- **どのようにしてポリシーを強制するのですか?**
- **チェックインの検証システムをどのように使用すればよいですか?**
- **ディスク上のファイル名の修正、またはファイルの削除を行った場合、バージョン管理では同期をとれませんか?**
- **自動的な競合解決はどのように動作しますか?**
- **競合を手動で解決するにはどのようにすればよいですか?**
- **どのようにして競合を回避すればよいですか?**

変更セットとは何ですか?

変更セットとは、チェックインに関連付けられている変更のセットを表します。変更セット内のすべての変更は、変更セットにチェックインしたときに最小単位の変更要素として TFS に適用されます。変更セットは一意の、連続して増加する数字によって識別されます。

どのファイルが変更されたか、チェックインのコメント、および関連している作業項目の詳細を参照するために、後で変更セットを取り出すことができます。古い変更をレビュー、テストまたはビルドしなければならない場合は、変更セットに関連付けられているファイル バージョンを取り出すこともできます。

参考資料

- 変更セットを取り出す場合の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181416(VS.80).aspx の「方法 : 変更セットからファイルの以前のバージョンを取得する」を参照してください。

チェックイン ポリシーとは何ですか?

チェックイン ポリシーを使用して、チェックイン時にコーディング基準を強制することができます。たとえば、コードにチェックインする前に、各開発者が静的なコード分析を必ず実行するようにできます。

VSTS では、デフォルトで次のチェックイン ポリシーを使用できます。

- **コード分析** – チェックインの前に必ずコード分析を実行します。
- **テスト ポリシー** – チェックインの前に必ずチェックイン テストが完了しているようにします。
- **作業項目** – チェックインに必ず 1 つ以上の作業項目を関連付けるようにします。

カスタム チェックイン ポリシーを作成して、既定では使用できないチェックを実行することができます。たとえば、禁止されている API コールなどのコード パターンを無効にしたり、チェックイン コメントの使用を強制したり、チームのスタイル ガイドラインについてチェックを強制したりできます。

参考資料

- チェックイン ポリシーのカスタマイズ方法を知りたいときは、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。

- チェックインの際に特定のパターンだけ禁止するサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx の「Checkin Policy to Disallow Certain Patterns」を参照してください。
- チェックインの際にコメントを義務付けるサンプル コードを見たいときは、http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx の「Sample Checkin Policy: Make Sure the Comment Isn't Empty」を参照してください。
- 新しいチェックイン ポリシーの登録方法を知りたいときは、http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx の「I've Made a New Check-In Policy! How Do I Add It?」を参照してください。

チェックイン ポリシーはいつ、どのようにオーバーライドできますか?

チェックイン ポリシーは、**[ポリシー エラーをオーバーライドして、チェックインを続行]** チェックボックスを選択してオーバーライド (無効化) できます。ファイルをチェックインする権限を持っているチームのすべてのメンバーは、チェックイン ポリシーをオーバーライドすることができます。

チームのメンバーがチェックイン ポリシーをオーバーライドしたことを検出したい場合は、Team Foundation Eventing Service を使用してチェックイン イベントを捕捉できます。

参考資料

- チェックイン ポリシーのオーバーライドについて詳しく学習するには、http://msdn2.microsoft.com/ja-jp/library/ms245460(VS.80).aspx の「方法 : チェックイン ポリシーをオーバーライドする」を参照してください。
- Team Foundation Eventing Service について詳しく学習するには、http://msdn2.microsoft.com/en-us/library/bb130154(vs.80).aspx の「Eventing Service」を参照してください。

どのようにしてポリシーを強制するのですか?

Team Foundation Version Control には、ポリシーの無視を防ぐ手段がありません。ただし、次以下

の手順を実行すれば、ポリシーが無視されたかどうかを検出することができます。

- チェックイン イベントに捕捉するために、(Team Foundation Core Services API から) Team Foundation Eventing Service を使用します。
- チェックイン ポリシーが無視された場合に変更セットの詳細を解析して、その後で変更セットに対する処理を行う Notify メソッドを記述します。

また、手動で変更セットの履歴を細かく調べて、ポリシーが無視されたかどうかを見つけることもできます。

参考資料

- ポリシーの違反があったときに電子メール メッセージを受信する方法については、http://blogs.infosupport.com/marcelv/archive/2005/10/18/1635.aspx を参照してください。

チェックインの検証システムをどのように使用すればよいですか?

Team Foundation のバージョン管理のチェックイン ポリシーを、チェックインの検証システムとして使用することができます。Team Foundation Server にはチェックイン ポリシーが付随しており、チェックインがコミットされる前に、次の内容が確実に実行されるようにできます。

- 作業項目が変更に関連付けられている
- 単体テストがすべて合格している
- 静的解析がきちんと実行されている

新しいポリシーのプラグインを作成して、独自のチェックイン要件を作成することができます。プラグインによって、コードがチームのコーディング基準に合っていること、ビルドの検証テストが実行されたこと、またチームのニーズにとって重要な他のすべての要件についても保証することができます。

参考資料

- チェックイン ポリシーの作成およびカスタマイズについて詳しく学習するには、http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx の「チュートリアル : チ

ェックイン ポリシーとチェックイン メモのカスタマイズ」を参照してください。

ディスク上のファイル名の修正、またはファイルの削除を行った場合、バージョン管理では同期をとれませんか?

はい。ファイルまたはフォルダを誤って削除した場合でも、TFS サーバーは、ローカル マシン上に最新バージョンがあるとみなします。これは、**get latest version** コマンドでは、削除されたファイルをディスク上に追加して戻さないことを意味しています。このような場合は、**force get** コマンドを使用してファイルまたはフォルダを復元します。

ファイル名を変更する、またはファイルやフォルダを削除する必要がある場合は、ソース エクスプローラを使用してこれらの処理を行い、サーバーがローカルな変更との同期を保持するようにします。

自動的な競合解決はどのように動作しますか?

保留中に **change**、**merge**、または **get** 操作を行うと、競合が発生することがあります。競合を解決する際には、Visual Studio に競合を自動的に解決させるよう選択することができます。自動競合解決は、重複した部分の変更 (同じコード行に適用されている変更など) を持たない非バイナリ ファイルでのみ機能します。この場合には、2 つのバージョンのファイルの変更が、新しいファイルのバージョンにマージされます。

自動競合解決が機能しない場合は、いずれか一方のファイルの変更を受け入れるか、または VSTS で提供されるグラフィカルな比較ツール、および差分抽出ツールを使用して手動でマージを実行できます。

参考資料

- 競合の解決の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181433(VS.80).aspx の「方法 : 競合を解決する」を参照してください。

競合を手動で解決するにはどのようにすればよいですか?

マージの競合を解決するには、Visual Studio のマージ ツールを使用します。マージ中に競合が検出された場合は、自動または手動で競合を解決することができます。競合を手動で解決するよう選択する場合には、マージ元の変更を保持する、マージ先の変更を保持する、またはマージ ツールで競合を解決する、といったことができます。

分岐間で変更をマージする、ファイルをワークスペースに取り込む、または新しいバージョンのファイルをチェックインする場合、競合を解決しなければならないことがあります。競合には次の 3 つのタイプがあります。

- **バージョン** – 分岐したパスに従ってファイルが変化しています。これは、ファイルの編集、名前変更、削除、または (削除したファイルの) 復元によって発生することがあります。
- **ファイル名の衝突** – 2 つ以上の項目が同じパスを占有しようとしています。
- **ローカルな上書き** – ローカルな編集可能ファイルを上書きしようとするときに、**get** 操作でのみ発生します。

大半の競合は自動的に解決できます。バージョンの競合だけに手動によるマージ操作が必要となることがあります。手動によるマージは、次のシナリオで最も一般的に行われます。

- ファイルが両方の分岐で編集されており、コードの同じ行が変更されている。
- 基点のないマージが行われており、分岐ファイルの関係が TFS で認識されていない。

マージ ツールは、各競合の詳細を提示し、最終的にマージされるファイルでどの変更を保持したいかを選択できるようにします。マージ元の変更を保持する、マージ先の変更を保持する、両方の変更を統合する、ファイルに直接入力して最終バージョンを手動で修正する、といった方法を選択できます。

ファイル内のすべての競合を解決したら、ターゲットの分岐にチェックインする予定の保留中の変更として、最終バージョンを保存します。

マージをする際は、ビルドが不安定になる間違いをしやすいため、注意が必要です。マージが完了したら、結果のソースをコンパイルし、単体テストを実行して大きな問題がないかテストします。

参考資料

- 競合の解決の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181433(VS.80).aspx の「方法 : 競合を解決する」を参照してください。

どのようにして競合を回避すればよいですか?

大半の競合は、Team Foundation のバージョン管理によって自動的に解決できます。コードの同じ行で変更が重複している、または複数の開発者が同時に 1 つのファイルで作業しているというシナリオでは、ほとんどの場合に手動によるマージが必要であるため、それが発生しないようにすることは重要です。

保留中の変更を参照する方法

1. ソース管理エクスプローラで、保留中の変更を参照したいソリューション、プロジェクト、フォルダ、またはファイルを右クリックします。
2. **[保留中の変更を表示]** を選択します。

この方法では、選択した範囲内で保留されているすべての変更が表示されます。
コマンドラインのチェックアウト ツールを使用すると、他のどのユーザーがこのファイルをチェックアウトしたか、およびユーザーの保留中の変更の性質に関する情報が提示されます。次の例では、Match.cs というファイルがチェックアウトされており、他に 2 人のユーザー (James と Sally) がそのファイルで作業中であることを TFS が通知しています。この通知では、各ユーザーが自分のワークスペース内に保持している保留中の変更の種類 (Edit および Rename) を示し、ファイルのローカル バージョン (現在のワークスペースのバージョン) が、最新のリポジトリ バージョンをベースとしていないことを示しています。

```
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

他の開発者と定期的に連絡し、特定のファイルに対する変更の性質を示して、手動によるマージが必要になる変更を重複させないようにします。2 人の開発者がコードの同じ部分で作業しないようにするには、連携をとることが良い方法です。それ以外については、複数の開発者が同じファイルの違う部分を編集しても、まったく問題はありません。

参考資料

- ワークスペースにおける保留中の変更の参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181400(VS.80).aspx の「方法 : ワークスペース内のすべての保留中の変更を表示および管理する」を参照してください。
- 他のワークスペースにおける保留中の変更の参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181401(VS.80).aspx の「方法 : 他のワークスペース内の保留中の変更を表示する」を参照してください。

チェックアウト、取得、およびロック

- **ファイルを最後に修正した開発者を見つけるにはどのようにすればよいですか?**
- **get コマンドはどのように機能しますか?**
- **共有チェックアウトと排他チェックアウトの違いは何ですか?**
- **lock コマンドはいつ使用すればよいですか?**
- **TFS はどのような種類のロックをサポートしていますか?**

ファイルを最後に修正した開発者を見つけるにはどのようにすればよいですか?

ファイルの履歴を調べて、誰が最後にファイルを修正したかを判断することができます。このようにするには、ソリューション エクスプローラでファイルを右クリックし、**[履歴の表示]** をクリックします。これにより、[履歴] ウィンドウが表示され、修正と、これらの修正を行ったユーザーの一覧が示されます。対象のファイルを最後に修正した開発者は、一覧の先頭に示されます。tf.exe コマンドライン ツ

ールから **history** コマンドを使用して、コマンドラインからファイルのバージョン履歴を参照することもできます。

参考資料

- 履歴の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181415(VS.80).aspx の「方法 ： 履歴データを表示する」を参照してください。

get コマンドはどのように機能しますか?

自分のマシンのファイルと、Team Foundation Server のファイルの同期をとるには **get** コマンドを使用します。**get** 操作を実行すると、Visual Studio は次の手順を実行します。次の手順では、パフォーマンスを向上させるために TFS プロキシがすでにインストールされていることを前提としています。

1. サーバーが知っているユーザーのディスク上のバージョンに基づいて、サーバーがどのファイルが古いかを判断します。
2. サーバーは、ワークスペースを更新するためにしなければならないことをクライアントに通知します。
3. クライアントは必要に応じてファイルをダウンロード、移動、および削除し、どの処理が完了したかをサーバーに通知します。

get 操作では、ファイルに編集のためのマークを付けません。また既定では、編集のためにチェックアウトしたファイルを上書きしません。

参考資料

- **get** コマンドの詳細については、http://msdn2.microsoft.com/ja-jp/library/fx7sdeyf(VS.80).aspx の「Get コマンド」を参照してください。

共有チェックアウトと排他チェックアウトの違いは何ですか?

Team Foundation Server のソース管理は、共有チェックアウトと排他チェックアウトの両方をサポートしています。

排他チェックアウトでは、ユーザーがソース管理にファイルをチェックインして戻すまで、他のユーザーはそのファイルをチェックアウトできません。これにより、開発プロセスでボトルネックが発生することがあります。

既定では、TFS は複数のユーザーが同じソース管理の項目を同時にチェックアウトできるようにしています。これは、共有チェックアウトと呼ばれます。このモデルでは、複数の開発者が、自身のワークスペース内で同じソース ファイルのコピーに対して作業することができます。Team Foundation Server は、ある開発者のワークスペース内にどのバージョンがあるかを認識しています。また、開発者はチェックインする前に競合を解決しなければなりません。

ほとんどの共同の開発環境では、自分のワークスペース内で行った変更が、他のユーザーのワークスペースにおける保留中の変更と競合する、またはその逆であることはほとんどありません。ワークスペースで発生する競合の大半は、TFS によって自動的に解決されます。自動的に解決できない競合に対しては、resolve コマンドを使用して、どの(自分の、または他の人の) 変更を保持しておくかを確実に判断します。

参考資料

- ロックの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181420(VS.80).aspx の「方法 : フォルダまたはファイルをロックおよびアンロックする」を参照してください。

lock コマンドはいつ使用すればよいですか?

自分が作業しているファイルをロック解除するまで、他の開発者が変更をチェックアウトまたはチェッ

クインしないようにするには、**lock** コマンドを使用します。複雑な手動のマージ処理を行うことになる競合が発生することを懸念する場合のみ、ファイルをロックします。大半の競合は自動的に解決できるため、**lock** コマンドを使用する場合は、慎重に使用します。

参考資料

- ロックの種類の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181419(VS.80).aspx を参照してください。

TFS はどのような種類のロックをサポートしていますか*?*

Team Foundation Server には、2 つの種類のロック、つまりチェックイン ロックとチェックアウト ロックがあります。チェックイン ロックはチェックアウト ロックよりも制約が厳しくありません。チェックイン ロックを適用すると、他のユーザーはその項目に対するローカルな変更を他のワークスペース内で継続して行うことができます。ワークスペースからチェックイン ロックを明示的に削除するか、またはファイルをチェックインするまで、その変更はチェックインできません。

チェックアウト ロックはチェックイン ロックよりも制約が厳しくなります。ソース管理のファイルまたはフォルダにチェックアウト ロックを適用すると、ユーザーは、編集のためにファイルをチェックアウトすることも、事前に保留中になっていた変更をチェックインすることもできません。項目に何らかの保留中の変更がある場合は、チェックアウト ロックを取得できません。

参考資料

- ロックの種類の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181419(VS.80).aspx を参照してください。

分散/リモート開発

- **オフラインでどのように作業すればよいですか?**

- **分散したチーム開発を最適化するにはどのようにすればよいですか?**
- **TFS のバージョン管理プロキシとは何ですか?**
- **TFS のバージョン管理プロキシのパフォーマンスを最適化するにはどのようにすればよいですか?**

オフラインでどのように作業すればよいですか?

Team Foundation のバージョン管理は、もともとオフラインの作業をサポートしていません。オフラインで作業する場合は、正確に以下のワークフローに従ってください。

1. 手動で読み取り専用フラグを外します。
2. ファイルを編集します。
3. ファイルを追加または削除します。
4. Team Foundation Power Tool (TFPT) online コマンドを実行します。

注意:オフライン中はどのファイルも名前を変更してはいけません。

1. 手動で読み取り専用フラグを外します。

既定では、ワークスペース内のファイルのうちまだチェックアウトの済んでいないファイルにはすべて、読み取り専用というマークが付いています。サーバーに接続せずに作業する場合は、その読み取り専用というフラグを手動で外さないと、各ファイルの編集も削除もできません。読み取り専用のフラグを外すときは、Windows エクスプローラで目的のファイルを右クリックし、**[プロパティ]** をクリックし、**[読み取り専用]** チェック ボックスをオフにし、**[OK]** をクリックしてください。**attrib -r** という DOS コマンド を使用しても同じことができます。

2. ファイルを編集します。

読み取り専用フラグを外したファイルは、自由に編集することができます。

3. ファイルを追加または削除します。

読み取り専用フラグを外したファイルは追加も削除もできます。TFPT online ツールでは、名前変更の

処理と追加/削除の処理とが区別ができないため、ファイルの名前は変更しないでください。

注意: Tfpt online コマンドで削除を行う場合はオプションを指定する必要がありますが、処理に時間がかかります。

4. TFPT online コマンドを実行します。

オンラインに戻って、コマンド ラインに「**TFPT online**」と入力して TFPT online コマンドを実行してください。このコマンドを実行すると、書き込み可能なファイルがないかどうかについてワークスペースが検索され、どの変更をサーバー上で保留にした方がよいかが決定されます。既にどれかファイルを削除してある場合は、/deletes コマンドライン オプションを使用して、ローカル ワークスペースに含まれている削除ファイルも検索されます。その後、どの変更をワークスペースに保留するかを選択できる **[Online]** ウィンドウが表示されます。

参考資料

- MSDN から TFPT オンライン ツールをダウンロードするには、http://www.microsoft.com/downloads/details.aspx?familyid=7324C3DB-658D-441B-8522-689C557D0A79&displaylang=en へアクセスしてください。
- Team Foundation Power Tool について詳しく学習するには、http://blogs.msdn.com/buckh/archive/2005/11/16/493401.aspx の「Power Tools: tfpt.exe」を参照してください。

分散したチーム開発を最適化するにはどのようにすればよいですか?

次の 3 つのソリューションのいずれかを選択して、インターネットを介して TFS へのアクセスを提供できます。

- **VPN 接続を使用する。**バーチャル プライベート ネットワーク (VPN) を介して TFS にアクセスを提供できます。
- **リバース プロキシを介して TFS をパブリッシュする。**Microsoft Internet Security and Acceleration (ISA) Server などのリバース プロキシを通じて TFS へアクセスを提供できます。
- **TFS をエクストラネットへ配置する (「ホステッド シナリオ」)。**エクストラネット上で TFS サ

ーバーをホストできます。

VPN アクセスを伴うリモート ユーザーをサポートしている場合は、VPN ソリューションを使用します。これは、汎用的なセキュリティを提供する、TFS のすべての機能に対してリモート アクセスできるようにする、TFS プロキシを使用してパフォーマンスを改善する、といったことを実現するための最も簡単なソリューションです。このソリューションでは、TFS を内部ネットワークの中に配置し、外部のユーザーは VPN を介して TFS へアクセスします。内部ユーザーは、TFS に直接アクセスします。

VPN アクセスを伴わない、またはドメインに対してアクセスしないリモート ユーザーをサポートする場合は、リバース プロキシのシナリオを使用します。このソリューションは、セットアップするのが少し難しくなりますが、リモート ユーザーは、VPN を必要とせずに、内部に配置されている TFS にアクセスできるようになります。このソリューションでは、TFS を内部ネットワークの中に配置し、ISA Server などの 1 つ以上のリバース プロキシ マシンが、クライアントの要求をインターネットから TFS へ提供します。

コミュニティ開発サイトなど、専用に使用する TFS インストレーションを使用する一部のリモート ユーザーをサポートしている場合は、エクストラネットのシナリオを使用します。このソリューションは、リモート ユーザーと内部ネットワーク リソースの間を最大に分離します。このソリューションでは、外部のクライアントのみ TFS にアクセスします。TFS はエクストラネット上のファイアウォールの外部に配置します。

リモートの Team Foundation Server へ接続する複数のクライアントを持つオフィスをサポートする場合は、リモート オフィス内に Team Foundation Server プロキシをインストールし、構成する必要があります。これにより、リモート オフィスのプロキシ サーバー上でソース管理ファイルをキャッシュするため、パフォーマンスが改善されます。

リモート TFS へ接続している 1 つのクライアントをサポートしている場合は、TFS へ直接接続するようクライアントを構成します。

参考資料

- TFS プロキシについて詳しく学習するには、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx の MSDN の「Team Foundation Server のプロキシとソース管理」を参照してください。
- TFS Proxy File Cache Server について詳しく学習するには、http://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?culture=ja-jp&EventID=1032291120&CountryCode=JP の MSDN Webcast を参照してください。

TFS のバージョン管理プロキシとは何ですか?

TFS におけるクライアント/サーバーの通信では、Hypertext Transfer Protocol (HTTP) を使用します。TFS のバージョン管理プロキシは、パフォーマンスを向上させ、サーバーに対する不要なラウンドトリップを回避するために、リモートの場所 (広域ネットワーク (WAN) 接続によって TFS データベースから分離されているチームの場所) のサーバーにインストールされます。このプロキシは、中枢の TFS サーバーから離れて、リモートの場所でソース管理されているファイルのコピーをキャッシュします。ファイルがローカル キャッシュにない場合は、プロキシによって TFS からローカル キャッシュへファイルがダウンロードされてから、ファイルがクライアントへ戻されます。

リモート アクセスは、プロキシを使用するうえでは最も一般的なシナリオですが、メイン サーバーの負荷を軽減したい場合にはいつでも使用することができます。たとえば、最新のソースを取得するために、多数のローカルな要求が同時に発生することによってサーバーの負荷が高くなっている場合は、この処理をプロキシへ移行して負荷を減らすことができます。すべてのアプリケーション層のインストールにはプロキシが含まれているため、一般的には、アプリケーション層の前面にプロキシを配置してもメリットはありません。

参考資料

- TFS のバージョン管理のプロキシの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252490(VS.80).aspx の「Team Foundation Server のプロキシとソース管理」を参照してください。

TFS のバージョン管理プロキシのパフォーマンスを最適化するにはどのようにすればよいですか*?*

プロキシのパフォーマンスを最適化するには、次のアプローチを考慮します。

- キャッシングが有効となっていることを確認し、キャッシュのパフォーマンスを監視します。プロキシがどのように実行されているかを確認するために、プロキシ サーバー上で定期的に、(デフォルトでインストールされている) パフォーマンス カウンタ、および (エラーと警告のための) イベント ログを監視します。TFS プロキシは、ProxyStatistics.xml という名前の Extensible Markup Language (XML) ファイルにキャッシュ パフォーマンスの統計を保存しています。これらの統計を保存する間隔は変更することができます。ProxyStatistics.xml ファイルは、プロキシのインストール ディレクトリの App_Data フォルダにあります。
- 定期的に最新ファイルをプロキシ サーバーに取得したいときは、スケジュール タスクを実行してください。こうすれば、必ずプロキシ キャッシュに最新バージョンのファイルが入っていることになりますし、そうしたファイルをクライアントが要求した場合に、必ずキャッシュ内に目的のファイルが見つかることになります。
- 帯域幅の狭いネットワーク (3 Mbps 未満) 経由で大きなファイルをダウンロードすることになることがわかっている場合は、Web.config の executionTimeout を適切な値に設定してください。デフォルトの値は 1 時間、つまり <httpRuntime executionTimeout="3600"/> です。

参考資料

- TFS のパフォーマンス調査の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms252455(VS.80).aspx を参照してください。

移行

- **TFS のバージョン管理は VSS とどのように違いますか?**
- **チェックアウトのモデルは VSS とどのように違いますか?**
- **VSS から TFS へソースをどのように移行すればよいですか?**

- **他のバージョン管理システムからソースをどのように移行すればよいですか?**

TFS のバージョン管理は *VSS* とどのように違いますか*?*

Microsoft Visual SourceSafe は個人の開発者および小規模のチームを対象にしていますが、TFS のバージョン管理は、プロフェッショナルな開発者、および 1 台のサーバーにつき最大 2500 人のユーザーの大規模なチームを対象にしています。その他に、TFS のバージョン管理によって実現した主な改善点としては、Visual Studio 2005 との高度な統合、および WAN 経由での効率的なファイル アクセスを実現した分散チームのサポートがあります。2 つのシステムの主な違いには、次のものがあります。

- **コード チャーン –** Visual SourceSafe には、コード チャーンのメトリックをサポートする機能がありません。TFS のバージョン管理には、データ ウェアハウスへチェックインするすべてのファイルに対するコード チャーンのメトリックが用意されており、データを任意に切り取ることができます。
- **チェックアウト –** Visual SourceSafe では、常に最新のファイルまたはピン設定されたファイルをデータベースから取得して、自身のローカル バージョンを上書きできます。TFS のバージョン管理では、ワークスペース内のファイルのバージョンを書き込み可能にできますが、データベースのバージョンでこのバージョンを上書きすることはできません。
- **ファイル ロッキング –** Visual SourceSafe では、チェックアウト時にファイルに対して自動的に排他ロックが適用されます。TFS のバージョン管理では、デフォルトで複数の開発者が 1 つのファイルを同時にチェックアウトすることができます。競合が発生した場合には、通常は自動的に解決されます。
- **分岐 –** Visual SourceSafe は固定と共有の分岐方法を使用します。TFS のバージョン管理ではピンはサポートしていません。代わりに path-space の分岐を使用します。これにより、ソフトウェアの古いバージョンの管理がより簡単になります。これは、ある分岐から別の分岐へ変更をマージしたり、2 つの分岐で同時に変更を行ったりすることをスムーズにできるためです。
- **共有 –** TFS のバージョン管理では共有をサポートしません。共有ファイルを移行すると、宛先フォルダにファイルがコピーされます。
- **ワークフローの統合 –** Visual SourceSafe はスタンドアロンのバージョン管理システムです。TFS のバージョン管理は、作業項目の追跡および TFS レポーティングと統合されています。

参考資料

- Visual SourceSafe と TFS のバージョン管理の違いの詳細については、http://msmvps.com/blogs/vstsblog/archive/2005/07/02/56401.aspx の「Visual Source Safe vs. Team Foundation Server」を参照してください。

チェックアウトのモデルは *VSS* とどのように違いますか*?*

TFS のバージョン管理からファイルをチェックアウトすると、ファイルは各自のワークスペース内で書き込み可能としてマークされますが、データベース サーバーとは同期しません。データベース サーバーから最新のバージョンを取得するには、最初に **get** コマンドを使用する必要があります。一方、VSS のチェックアウト処理では、データベース サーバーからファイルを取得し、各自のワークスペース内で書き込み可能としてマークします。

Visual SourceSafe はデフォルトで排他チェックアウトを使用しますが、TFS はデフォルトで共有チェックアウトを使用します。共有チェックアウトでは、複数の開発者が 1 つのファイルに対して同時に変更することができます。Visual SourceSafe とは異なり、TFS では大半の競合が自動的に解決されます。

参考資料

- VSS のチェックアウトと TFS のバージョン管理のチェックアウトの違いの詳細については、http://blogs.msdn.com/korbyp/archive/2004/07/28/199720.aspx の「Team Foundation vs. SourceSafe | Checking Out」を参照してください。

VSS から *TFS* へソースをどのように移行すればよいですか*?*

Team Foundation Server には VSS コンバータが付随しています。これを使用してファイル、フォルダ、バージョン履歴、ラベル、およびユーザー情報を VSS データベースから TFS のバージョン管理へ移行することができます。

このコンバータには、次のようにいくつかの重要な制約があります。

- ファイル共有の履歴情報は保持されません。Team Foundation Server は共有をサポートしません。共有ファイルを移行すると、宛先フォルダにファイルがコピーされます。
- 分岐の履歴情報は保持されません。
- Team Foundation Server はピンをサポートしません。ピン設定されたファイルは、2 つのラベルを作成することによって移行されます。
- アクションに関連付けられているタイムスタンプは、移行時には保持されません。

参考資料

- 移行の準備方法の詳細については、
  http://msdn2.microsoft.com/ja-jp/library/ms181246(vs.80).aspx を参照してください。
- 移行の実行方法の詳細については、
  http://msdn2.microsoft.com/ja-jp/library/ms181247(VS.80).aspx を参照してください。
- VSS コンバータの制約事項の詳細については、
  http://msdn2.microsoft.com/ja-jp/library/ms252491(VS.80).aspx を参照してください。
- VSS Analyze ツールの詳細については、
  http://msdn2.microsoft.com/ja-jp/library/ysxsfw4x(VS.80).aspx を参照してください。
- VSSConverter のコマンドライン ユーティリティの **migrate** コマンドの詳細については、
  http://msdn2.microsoft.com/ja-jp/library/ms400685(VS.80).aspx を参照してください。

他のバージョン管理システムからソースをどのように移行すればよいですか?

ファイルは、移行の際の起点となるバージョン管理システムから Team Foundation Server Version Control へと手動でエクスポートすることができます。
移行の際の起点となるバージョン管理システムからファイル履歴またはその他属性を保存する場合は、TFS のバージョン管理オブジェクト モデルを使用して独自の移行ツールを記述することができます。

Microsoft は現在 ClearCase コンバータに取り組んでいます。準備ができたら、

http://blogs.msdn.com/tfs_migration の TFS MIgration ブログでアナウンスされます。

Component Software は、VSS、GNU RCS、CS-RCS、GNU CVS、および Subversion (SVN) に対応できるコンバータを作成しました。

参考資料

- TFS のバージョン管理の拡張性の詳細については、http://msdn2.microsoft.com/en-us/library/bb187335(VS.80).aspx の「Walkthru: The Version Control Object Model」を参照してください。
- Component Software コンバータの詳細については、http://www.componentsoftware.com/Products/Converter/ を参照してください。

プロジェクト/ワークスペースの管理

- **チーム プロジェクトはどのように編成すればよいですか?**
- **プロジェクト間の依存関係はどのように管理すればよいですか?**
- **ワークスペースとは何ですか?**
- **開発者を分離するためにワークスペースをどのように使用すればよいですか?**
- **ワークスペースをマッピングするために実証されているプラクティスにはどのようなものがありますか?**
- **新しいチーム プロジェクトまたは新しい分岐はいつ作成すればよいですか?**
- **複数のプロジェクト間で共有しているソース コードはどのように管理すればよいですか?**
- **複数のプロジェクト間で共有しているバイナリはどのように管理すればよいですか?**
- **ソース ツリーはどのように編成すればよいですか?**

チーム プロジェクトはどのように編成すればよいですか?

チーム プロジェクトは、組織内で考えられる最大の作業単位を表すことを意図しています。これは、開発中の製品という形になることがよくあります。

チーム プロジェクトを管理するためには、次のいずれかの方針を考慮してください。

- **アプリケーションごとに 1 つのプロジェクトを作成する。**
- **リリースごとに 1 つのプロジェクトを作成する。**

アプリケーションごとに 1 つのプロジェクトを作成する

リリースが代わっても、ソース コードだけでなく、作業項目および TFS の他の資産も継続して使用する場合は、アプリケーションごとに 1 つのプロジェクトを使用することを考慮します。アプリケーションの複数のバージョンに対して 1 つのプロジェクトを使用している場合は、TFS のすべての資産は次のリリースでも自動的に継続して使用されます。アプリケーションの新しいバージョンをリリースする準備ができたら、プロジェクト内に、そのリリースを表す 1 つの分岐を作成し、コードを分離させます。

アプリケーションごとに 1 つのプロジェクトを使用しない理由は次のとおりです。

- 並行しているリリースでは、作業項目のスキーマ、チェックイン ポリシー、およびプロセス ガイダンスを共有することが強制されます。
- レポーティングはさらに複雑です。これは、デフォルトでプロジェクト全体を対象とするため、リリースごとにフィルタリングを追加する必要があるためです。
- 数百のアプリケーションがあり、それぞれが独自のプロジェクト内にある場合は、TFS のパフォーマンスおよび拡張性の制約の問題に直面します。
- 複数のリリースにわたって「荷物」が累積されます。これを取り除くための最も簡単な方法は、新しいチーム プロジェクトを作成し、そのプロジェクトで継続して使用したいコードを分岐することです。

リリースごとに 1 つのプロジェクトを作成する

作業項目や、その他の TFS 資産を移行せずに、リリースごとに 1 からやり直す場合は、リリースごとに 1 つずつプロジェクトを使用する方法を検討してみてください。

リリースごとに新規のプロジェクトを使用すると、前のリリースに影響を及ぼさずに、作業項目スキー

マ、ワークフロー、チェックイン ポリシー、その他資産を変更することができます。

たとえば、主要開発チームとはプロセスもワークフローも異なる可能性のある、維持管理チームのような別のチームが前のリリースを管理する場合は特に便利です。

リリース 1 つにつきプロジェクトを 1 つ使用するときは、以下の点に注意してください。

- ソース コードをプロジェクトから別のプロジェクトへ移動するのは非常に簡単ですが、作業項目やその他の TFS 資産をプロジェクトから別のプロジェクトへ移動するのは難しいです。作業項目は、一度に 1 つずつしか別のプロジェクトにコピーできないため、いくつかまとめて作業項目をコピーしたい場合は、ユーティリティを作成する必要があります。
- アプリケーションとリリースが何百もあり、それぞれが別々のプロジェクトに含まれている場合は、TFS のパフォーマンスと拡張性が限界に達します。

方針を選択する場合には、次のことに留意してください。

- 既存のチーム プロジェクトの構造を変えるのは難しいため、長い期間利用できる構造を選んでください。
- ソースは、次のようにしてチーム プロジェクトどうしで簡単に共有できます。
  - あるプロジェクトから別のプロジェクトへソースを分岐します。
  - 別のプロジェクトから自身のワークスペースへソースをマップします。
- Team Foundation Server で対応できるプロジェクトの個数は、MSF Agile プロセス テンプレートを使用した場合は 500 個、MSF CMMI プロセス テンプレートを使用した場合は 250 個まで拡張できます。MSF for CMMIR (MSF CMMI) プロセス テンプレートを使用した場合は 250 個まで拡張できます。自分だけのプロセス テンプレートを作成する場合、または既存のプロセス テンプレートをカスタマイズする場合は、サーバーの拡張性に一番大きく影響するのが作業項目スキーマであることを忘れないでください。スキーマが複雑だと、サーバーで対応できるプロジェクトの個数が減ります。

参考資料

- チーム プロジェクトの使用の詳細については、http://blogs.msdn.com/ericlee/archive/2006/08/09/when-to-use-team-projects.aspx の「When to use Team Projects」を参照してください。

プロジェクト間の依存関係はどのように管理すればよいですか?

バイナリの動的リンク ライブラリ (DLL) しか持っていないサードパーティのアセンブリなど、Visual Studio ソリューションの一部としてビルドされていない外部のアセンブリを参照する必要がある場合には、次の 3 つのいずれかを選択します。

- 外部のアセンブリをバイナリ リソースとして保存し、プロジェクトの一部とすることができます。この方法は、参照する可能性のある他のプロジェクトに対する依存関係を保持する計画がある場合、またはこのプロジェクトがこの依存関係を利用している唯一のプロジェクトである場合に、うまく機能します。
- 仮想ドライブ文字または Universal Naming Convention (UNC) パスのいずれかを使用して、ビルド サーバー上でアセンブリを参照することができます。
- 共有されているチーム プロジェクト内のフォルダに外部のアセンブリ セットを保存することができます。このアプローチは、この共有の依存関係を参照したいプロジェクトが多く、あなたのプロジェクトが明確な所有者ではない場合に、うまく機能します。
    - バイナリが変更されたときに、そのバイナリに対する即時の更新を受け取りたい場合は、このチーム プロジェクトに対するワークスペースのマッピングをローカル コンピュータに作成することができます。アセンブリに対するファイル参照を、ローカル ファイル構造の中に設定することができます。
    - この依存関係に対する統合スケジュールを制御したい場合は、共有プロジェクトから自身のプロジェクトへ分岐を作成し、変更を取得する用意ができたときに、いつでもそれをマージすることができます。

一般的には、チームのプロジェクト間にまたがる依存関係は使用せずに、代わりに、同じチーム プロジェクトのもとで関連/依存するすべてのソリューション/プロジェクトを持つようにします。これによって、ビルド スクリプトのカスタマイズに対する必要性が少なくなります。依存関係がある場合は、

プロジェクト参照を使用して依存関係を定義します。ファイル参照は管理が難しいため、使用しないようにします。

注意:プロジェクト対プロジェクトの参照は、ソリューション内でサポートされます。また、プロジェクト ファイル (csproj、vjsproj、vcproj、vbproj など) は複数のソリューション (sln ファイル) に含めることが可能であり、プロジェクトは複数の個別のソリューションで共有できることに注意してください。

参考資料

- プロジェクト参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- ワークスペースの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181383(VS.80).aspx を参照してください。

ワークスペースとは何ですか?

ワークスペースは、TFS バージョン管理サーバー上にあるファイルとフォルダの、クライアント側のコピーです。ローカルで保留中になっている変更は、これらの変更をサーバーへ (変更セットと呼ばれる) アトミックな単位としてチェックインするまで、ワークスペース内で分離されます。1 つのワークスペースに、複数のチーム プロジェクトに対する参照を含めることができます。複数のワークスペースを使用して、自身だけで使用する目的でファイルまたはバージョンを分離することができます。

注意:ワークスペースは、マシンごと、ユーザー アカウントごとに作成されます。したがって、自分で使用するそれぞれのコンピュータに対して、ワークスペースの複数の定義を持つことができます。また、コンピュータにログオンする複数のユーザー アカウントを使用して、1 つのコンピュータ上に複数のワークスペースを持つことも可能です。

参考資料

- ワークスペースの詳細については、
  http://msdn2.microsoft.com/ja-jp/library/ms181383(VS.80).aspx を参照してください。

開発者の作業を分離するためにワークスペースをどのように使用すればよいですか?

開発者は、2 つのワークスペースを作成することができます。つまり、1 つは、チームの残りの開発者が作業しているファイルとフォルダへの参照が含まれているもの、もう 1 つは、あなたが分離したいと考えているファイルとフォルダが含まれているものです。これらのファイルは、どこか他の場所で発生している作業と並行して、特定のファイルを変更するために分離するとよいでしょう。たとえば、このアプローチを使用して、リスクの高い保留中の変更の作業をすることも、コード レビューを実施することもできます。

参考資料

- ワークスペースの詳細については、
  http://msdn2.microsoft.com/ja-jp/library/ms181383(VS.80).aspx を参照してください。

ワークスペースをマッピングするために実証されているプラクティスにはどのようなものがありますか?

ワークスペースのマッピングは、サーバー上のファイルとフォルダを、自分のローカル ハード ドライブの場所へマップするための方法です。

ワークスペースのマッピングを作成するには次のプラクティスを考慮します。

- **チーム プロジェクトのルート レベルでマッピングを作成する。**新しいチーム プロジェクトの場合は、チーム プロジェクトのルート ($/MyTeamProject) を、ローカル ドライブ上で同じ名前を持つフォルダ (C:¥TeamProjects など) へマップします。マッピングは再帰的であるため、ローカル フォルダ全体の構造は自動的に作成され、ソース管理内の構造と全く同じになります。

- **共有のコンピュータ上の一意のローカル フォルダ パスを使用する。**1 つのコンピュータの 2 人のユーザーは、同じワークスペースのマッピングを共有できません。たとえば、あなたと同僚は、同じチーム プロジェクト ($/MyTeamProject) を、ローカル コンピュータの同じファイルへマップできません。かわりに、(この方法ではロケーション パスが長くなりますが) My Documents の下にマッピングを作成するか、ローカル コンピュータ上でチーム フォルダの命名規則を決定します (C:¥TeamProjects¥User1、C:¥TeampProjects¥User2 など)。
- **ツリー全体は必要でない可能性がある。**パフォーマンスを改善し、必要とするディスク サイズを減らすために、開発プロジェクトで必要なファイルのみをマップします。一般的には、作業するソリューションに関連付けられているファイルおよびプロジェクトのみが必要になります。
- **プロジェクト間の依存関係をサポートするためにワークスペースのマッピングの使用を回避する。**一般的に、複数のチーム プロジェクトを横断するような依存関係は避け、関連のあるソリューション/プロジェクトもしくは依存しているソリューション/プロジェクトはすべて、同じチーム プロジェクトの下に置いておく方がよいです。そうすれば、ビルド スクリプトをカスタマイズする必要性が減ります。依存関係がある場合は、その依存関係の定義にプロジェクト参照を使用するか、共有プロジェクトから自身のプロジェクトへその依存関係を分岐するか、いずれかを行ってください。ファイル参照は管理が難しいため、使用しない方がよいです。例外は、依存プロジェクトを並行して開発していて、リアルタイムに変更が必要な場合です。この場合は、ワークスペース マッピングの使用を検討してもよいです。しかし、依存コードが原因で大きな変更が多く発生しすぎるような場合であれば、分岐を使用して分離バッファを作成することは可能です。

参考資料

- ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。
- ワークスペースの編集の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms245466(VS.80).aspx の「方法 : ワークスペースを編集する」を参照してください。

新しいチーム プロジェクトまたは新しい分岐はいつ作成すればよいですか?

プロジェクト内で開発者が作業しているコードを分離する一方で、同時に TFS の他のすべての資産 (作業項目、プロセス ガイダンス、レポートなど) を共有したい場合は、分岐を作成します。

作業項目、プロセス ガイダンス、レポートなど、TFS の資産を別に使用したい場合は新しいチーム プロジェクトを作成します。コードを継続して使用したいけれども、TFS の他の資産は継続して使用しない場合も、新しいチーム プロジェクトを作成します。

参考資料

- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- 新しいプロジェクトの追加の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181373(VS.80).aspx の「ソリューション エクスプローラからソース管理へのプロジェクトとソリューションの追加」を参照してください。

複数のプロジェクト間で共有しているソース コードはどのように管理すればよいですか?

共有ソースの管理には、次の 2 つの決定ポイントがあります。

1. ソースをどこに保存したいか?
2. チームにどのようにして共有ソースにアクセスして欲しいか?

ソースの保存に対しては次のオプションを利用できます。

- 共有ソースが明らかに特定のチームに所有されている場合は、そのソースをそのチーム プロジェクトに保存します。
- 共有ソースの明確な所有者がいない場合は、共有コードの専用のチーム プロジェクトを作成します。

ソースを別のプロジェクトで使用したい場合は、次の 2 つのオプションのいずれかを選択します。

- 共有コードを常に同期させる必要がある場合は、共有場所のソースを、クライアント マシン上のローカル ワークスペースにマップします。

- 定期的に同期させればよい場合は、共有場所から、利用するチーム プロジェクトへ分岐します。共有場所から利用するプロジェクトへマージを実行するたびに、最新のソースが取得されます。

ワークスペースのマッピングまたは分岐のどちらを使用する場合でも、プロジェクト内で共有ソースが配置されている場所を明確にする命名規則を使用します。たとえば次のようにします。

- **Main** – 製品を出荷するために必要なすべての資産のコンテナ
  - **Source** – ビルドに必要なすべてのもの用のコンテナ
    - **Code** – ソース コード用のコンテナ
    - **Shared Code** – 他のプロジェクトから共有されるソース コード用コンテナ
    - **Unit Tests** – 単体テスト用コンテナ
    - **Lib** – バイナリ依存関係用のコンテナ
  - **Docs** – 製品と一緒に出荷する文書用のコンテナ
  - **Installer** – インストーラのソース コードおよびバイナリ用コンテナ
  - **Builds** – チーム ビルド スクリプト用コンテナ
  - **Tests** – テスト チームのテスト ケース用のコンテナ

参考資料

- プロジェクト参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- ワークスペースの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181383(VS.80).aspx を参照してください。

複数のプロジェクト間で共有しているバイナリはどのように管理すればよいですか?

共有バイナリの管理は、共有ソースの管理と似ており、バイナリを保存したい場所、およびチームでど

のようにしてバイナリにアクセスしたいかを決定する必要があります。

バイナリの保存に対しては次のオプションを利用できます。

- 共有バイナリが特定のチームに明らかに所有されている場合は、そのソースをそのチーム プロジェクトに保存します。
- 共有バイナリの明確な所有者がいない場合は、共有バイナリの専用のチーム プロジェクトを作成します。

バイナリを別のプロジェクトで使用する場合は、次のオプションを利用できます。

- 共有バイナリは通常定期的に更新されます。プロジェクトがこのケースの場合は、共有場所から利用するチーム プロジェクトへ分岐させます。バイナリが変更された場合は、マージを実行して最新バージョンを取得できます。
- 共有バイナリを常に同期させる必要がある場合は、共有場所のソースを、クライアント マシン上のローカル ワークスペースにマップします。

ワークスペースのマッピングまたは分岐のどちらを使用する場合でも、プロジェクト内で共有バイナリが配置されている場所を明確にする命名規則を使用します。たとえば次のようにします。

- **Main** – 製品を出荷するために必要なすべての資産のコンテナ
  - **Source** – ビルドに必要なすべてのもの用のコンテナ
    - **Code** – ソース コード用コンテナ
    - **Shared Code** – 他のプロジェクトから共有されるソース コード用コンテナ
    - **Unit Tests** – 単体テスト用コンテナ
    - **Lib** – バイナリ依存関係用コンテナ
  - **Docs** – 製品と一緒に出荷する文書用のコンテナ
  - **Installer** – インストーラのソース コードおよびバイナリ用コンテナ
  - **Builds** – チーム ビルド スクリプト用コンテナ
  - **Tests** – テスト チームのテスト ケース用のコンテナ

参考資料

- プロジェクト参照の詳細については、http://msdn2.microsoft.com/ja-jp/library/ez524kew(VS.80).aspx の「プロジェクト参照」を参照してください。
- 分岐の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181425(VS.80).aspx の「方法 : ファイルとフォルダを分岐する」を参照してください。
- ワークスペースの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181383(VS.80).aspx を参照してください。

ソース ツリーはどのように編成すればよいですか?

ソース ツリー構造は、フォルダ構造、ファイル構造、分岐構造の 3 つで構成されています。メイン分岐の内部では、以下のフォルダとファイルの構造がさまざまな規模のチームで機能することが既に証明されています。

- **Main** – 製品を出荷するために必要なすべての資産のコンテナ
  - **Source** – ビルドに必要なすべてのもの用のコンテナ
    - **Code** – ソース コード用コンテナ
    - **Shared Code** – 他のプロジェクトから共有されるソース コード用コンテナ
    - **Unit Tests** – 単体テスト用コンテナ
    - **Lib** – バイナリ依存関係用コンテナ
  - **Docs** – 製品と一緒に出荷する文書用のコンテナ
  - **Installer** – インストーラのソース コードおよびバイナリ用コンテナ
  - **Builds** – チーム ビルド スクリプト用コンテナ
  - **Tests** – テスト チームのテスト ケース用コンテナ

Main から作成する分岐は、このフォルダおよびファイル構造を新しい分岐へコピーします。たとえば次のようになります。

- **Development** – 開発の分岐
  - **Source** – ビルドに必要なすべてのものに用のコンテナ
    - **Code** – ソース コード用コンテナ
    - **Shared Code** – 他のプロジェクトから共有されるソース コード用コンテナ
    - **Unit Tests** – 単体テスト用コンテナ
    - **Lib** – バイナリ依存関係用のコンテナ
- **Main** – 統合分岐
  - **Source** – ビルドに必要なすべてのもの用のコンテナ
    - **Code** – ソース コード用コンテナ
    - **Shared Code** – 他のプロジェクトから共有されるソース コード用コンテナ
    - **Unit Tests** – 単体テスト用コンテナ
    - **Lib** – バイナリ依存関係用のコンテナ
  - **Docs** – 製品と一緒に出荷する文書用のコンテナ
  - **Installer** – バイナリ用コンテナ
  - **Builds** – チーム ビルド スクリプト用コンテナ
  - **Tests** – テスト チームのテスト ケース用のコンテナ

参考資料

- ワークスペースの作成の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx の「方法 : ワークスペースを作成する」を参照してください。

シェルビング

- **シェルビングとは何ですか?**
- **シェルブセットとは何ですか?**
- **一般的にシェルビングはいつ使用すればよいですか?**
- **シェルビングをどのように使用して作業をバックアップすればよいですか?**
- **シェルブセットをアンシェルブしたくなる理由は何ですか?**

シェルビングとは何ですか?

変更をシェルビングするプロセス (シェルブセットの作成とも呼ばれる) によって、変更されたファイルをチェックインせずに、保留中の変更をソース管理のもとに保存することができます。これにより、ファイルで完了していない可能性のある作業が定期的なビルドに影響を与えないこと、また定期的なビルドを破損しないことを保証したうえで、これらのファイルを (バックアップされる) サーバー上に置くことができるというメリットが得られます。

参考資料

- 保留中の変更のシェルビングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

シェルブセットとは何ですか?

シェルブセットは、保存されているけれども、ソース管理にコミットする準備ができていないファイルの集合です。保留中の変更をワークスペースへ保存するため、または他のチーム メンバーと共有してフィードバックに役立てるためにファイルをシェルブすることができます。シェルブされたファイルを使用して、完了していない作業を受け渡すこともできます。

参考資料

- 保留中の変更のシェルビングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

一般的にシェルビングはいつ使用すればよいですか?

シェルビングを使用する一般的なシナリオには、次のように多くのものがあります。

- ソース セットの変更中に、新しく優先度の高い作業が割り当てられた場合 (直ちにバグ修正をしなければならない場合など)。このときは、コードを安定したバージョンに戻さなければなりませんが、変更内容は失いたくありません。このような場合は、コードをシェルブしておけば、後で簡単に取り出すことができます。
- 一日の終わりに作業が完了していないけれども、現在の作業をサーバーへ確実にバックアップしたい場合。このような場合は、現在の変更内容をシェルビングすることによって、変更が Team Foundation Server へ適用され、別の日に自分で (または他のユーザーが) 変更を取り出すことができます。
- 部分的にしか完了していないコードについて、リモート チームのメンバーと検討またはレビューしたい場合。このような場合はコードを電子メールで送信するのではなく、コードをシェルブして、リモート ユーザーがシェルブからファイルを取り出すようにできます。
- 部分的にしか完了していない作業を、他の開発者に渡して完成させてもらう場合。

参考資料

- 保留中の変更のシェルビングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 : 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

シェルビングをどのように使用して作業をバックアップすればよいですか?

1 つまたは複数のソース ファイルにおいて、終業時に作業が完了していない場合は、コードをシェルブし、完了していない作業をチェックインせずにソースをサーバーへアップロードすることができます。

参考資料

- 保留中の変更のシェルビングの詳細については、

http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 ： 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

シェルブセットをアンシェルブしたくなる理由は何ですか?

次のような場合には、シェルブセットをアンシェルブできます。

- シェルブセットを使用して以前バックアップしたファイル セットで作業を再開する場合。
- シェルブした保留中の変更を、進行中の作業へ統合する場合。
- 他の人のコードをレビューする場合。
- 他の開発者の完了していない作業を受け取る場合。

参考資料

- 保留中の変更のアンシェルビングの詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181404(VS.80).aspx の「方法 ： 保留中の変更をシェルブおよびアンシェルブする」を参照してください。

ソース管理のリソース

- TFS のソース管理の詳細については、http://msdn2.microsoft.com/ja-jp/library/ms181237(VS.80).aspx の「Team Foundation ソース管理」を参照してください。

Visual Studio 2005 Team Foundation Server に関する How-To 集

この How-To 集では Visual Studio 2005 Team Foundation Server を一層効率的に、便利に活用いただくための様々な方法について紹介いたします。この How-To 集はマイクロソフトやパートナー企業における製品の活用の中で共有されたプラクティスを整理し、開発者の皆様へお伝えするためのもので、マイクロソフト本社において、パターンとプラクティスの提供を目的に組織されている Patterns & Practice グループ（通称「PAG（パグ）」もしくは「P&P」と呼ばれるグループ）により作成されています。
原文は CodePlex において公開されている「Team Development with TFS Guide」の How-To 集にです。（http://www.codeplex.com/TFSGuide から原文を入手いただけます）

[HOWTO]: #1
Visual Studio Team Foundation Server でプロジェクトに新規開発者を追加する方法

[HOWTO]: #2
Visual Studio Team Foundation Server でチーム ビルドを使用してコード分析を自動実行する方法

[HOWTO]: #3
Visual Studio Team Foundation Server のカスタム レポートを作成する方法

[HOWTO]: #4
Visual Studio Team Foundation Server のリスク オーバー タイム レポートを作成する方法

[HOWTO]: #5
Visual Studio Team Foundation Server のカスタム チェックイン ポリシーを作成するステップ

[HOWTO]: #6
Visual Studio Team Foundation Server でソース ツリーを作成するステップ

[HOWTO]: #7

Process Editor を使用してプロセス テンプレートをカスタマイズする方法

[HOWTO]: #8

Visual Studio Team Foundation Server でレポートをカスタマイズする方法

[HOWTO]: #9

Visual Studio Team Foundation Server でプロジェクトを管理する方法

[HOWTO]: #10

Visual SourceSafe から Visual Studio Team Foundation Server にソース コードを移行する方法

[HOWTO]: #11

Visual Studio Team Foundation Server で基点のないマージを実行する方法

[HOWTO]: #12

Visual Studio Team Foundation Server で常時結合をセットアップする方法

[HOWTO]: #13

Visual Studio Team Foundation Server でスケジュール ビルドをセットアップする方法

[HOWTO]: #14

Team Foundation Server 用に ASP.NET アプリケーションを構造化する方法

[HOWTO]: #15

Visual Studio Team Foundation Server 用に Windows アプリケーションを構造化する方法

[HOWTO]: #16

Visual Studio Team Foundation Server でソース管理フォルダを構造化する方法

[HOWTO]: Visual Studio Team Foundation Server でプロジェクトに新規開発者を追加する方法

適用対象

• Visual Studio Team Foundation Server

プロジェクトに新規開発者を追加する方法

プロジェクトに新規開発者を追加するには、Team Foundation プロジェクトおよびそれに関連した SharePoint プロジェクト サイトへのアクセスに適した許可を、その開発者に認可する必要があります。また、新規開発者のアカウントは、チーム サイト ポータルで提示されるようなレポートを表示するのに十分なアクセス許可を SQL Server Reporting Services に対して持っている必要もあります。

チーム プロジェクトにアクセス権を認可するには:

1. Team Foundation Administrators アプリケーション グループのメンバであるアカウントを使用して、Visual Studio にログオンします。
2. 必要なプロジェクトをチーム エクスプローラに追加します (まだリスト表示されていない場合)。
3. チーム プロジェクトを右クリックして、[チーム プロジェクトの設定] を指し、[グループ メンバシップ] をクリックします。
4. [Project¥Contributors] を選択し、[プロパティ] をクリックしてから、新規開発者のアカウントをこのグループに追加します。

貢献者グループのメンバシップには、開発者にとって必要な一連の標準的なアクセス許可が備わっていることに注意してください。そのようなアクセス許可によって、たとえばチーム プロジェクト内のアイテムの追加、変更、および削除を行うと共に、ビルドを実行することができます。

SharePoint プロジェクト サイトへのアクセス権を認可するには:

1. SharePoint 管理者サイト グループのメンバであるアカウントを使用して、チーム プロジェクト サイトにアクセスします。既定では、YourProject という名前のプロジェクトのプロジェクト サイト

は、http://server/sites/YourProject/default.aspx に位置しています。
2. [サイトの設定] をクリックします。
3. 管理タイトルの下の [ユーザーの管理] をクリックします。
4. [ユーザーの追加] をクリックします。
5. 新規開発者のアカウント名を domain¥useraccount の形式で入力し、[Contributor] を選択してから、[次へ] をクリックします。
6. 開発者の電子メール アドレスを入力してから、任意選択で、サイトへのアクセスに備えて歓迎メッセージを入力します。
7. [完了] をクリックします。

開発者は、貢献者グループのメンバシップを使用して、既存のドキュメント ライブラリおよびリストの表示やコンテンツの追加を行えることに注意してください。新規開発者のニーズによっては、サイトへの読み取り専用アクセス権のみを与えられる閲覧者グループのメンバシップで間に合うこともあります。

SQL Server Reporting Services にアクセス権を認可するには:
1. 管理者アカウントを使用して、SQL Server Reporting Services の管理 Web サイトにログオンします。このサイトは、http://server/reports に位置しています。
2. チーム プロジェクト名をクリックします。
3. [プロパティ] タブをクリックします。
4. [セキュリティ] タブをクリックします。
5. [新しいロールの割り当て] をクリックします。
6. 開発者の Windows アカウント名を入力してから [ブラウザ] を選択し、次に [OK] をクリックします。
開発者は、ブラウザ グループのメンバシップを使用して、レポートの表示や購読を行えることに注意してください。

詳細

セキュリティ グループ

チームに新しい開発者を追加するには、その開発者のアカウントを、該当するセキュリティ グループのメンバに追加して、以下の 3 つの個別領域にアクセスできるようにする必要があります。
-- TFS チーム プロジェクト
-- SharePoint ポータル プロジェクト サイト
-- SQL Server Reporting Services
これは、手動で行うことができます。つまり、各ツールの管理機能を使用するか、または以下からダウンロードできる TFS 管理ツール (現在ベータ 1) を使用します。

TFS サーバー レベル

TFS サーバー レベルでは、TFS は以下の 3 つの TFS アプリケーション グループ (AD グループではない) を定義しています。
SERVER¥Service Accounts - TFS サービス アカウントが含まれます。既定では、メンバは次のようになります。
-- シェルブされた変更の管理
-- 管理ウェアハウス
-- ワークスペースの管理
-- ワークスペースの作成
-- トリガ イベント
-- システムの同期情報の表示
既定メンバ
-- SERVER¥TFSSERVICE
SERVER¥Team Foundation Administrators - メンバは、サーバーでの管理操作を実行できます。既定では、メンバは次のようになります。
-- シェルブされた変更の管理
-- 管理ウェアハウス
-- ワークスペースの管理
-- トレース設定の更新
-- ワークスペースの作成
-- 新しいプロジェクトの作成

-- サーバー レベル情報の編集

-- プロセス テンプレートの管理

-- トリガ イベント

-- サーバー レベル情報の表示

-- システムの同期情報の表示

既定メンバ

-- SERVER¥ Service Accounts

-- Builtin¥ Administrator

-- Builtin¥ Administrators

SERVER¥Team Foundation Valid Users - サーバーにアクセスできるすべてのユーザーおよびグループが含まれます。既定では、メンバは次のようになります。

-- ワークスペースの作成

-- サーバー レベル情報の表示

既定メンバ

-- Project¥Build Services

-- Project¥Contributors

-- Project¥Project Administrators

-- Project¥Readers

SERVER¥Team Foundation Licensed Users - TFS の Workgroup Edition を使用するライセンスを交付されたユーザー。

-- 既定では、何の許可も受けていません。

既定メンバ

-- Builtin¥Administrator

チーム プロジェクト レベル

チーム プロジェクト レベルでは、TFS は以下のアプリケーション グループ (AD グループではない) を定義しています。

TeamProject¥BuildServices - ビルド サービスの許可 (サービス アカウント用)

-- ビルド品質の編集

-- テスト結果の発行

-- ビルドの開始

-- プロジェクト レベル情報の表示

-- ビルドの操作可能ストアへの書き込み

既定メンバ

-- なし

TeamProject¥Contributors - チーム プロジェクト内の項目を追加、変更、および削除することができます。

-- テスト結果の発行

-- ビルドの開始

-- プロジェクト レベル情報の表示

既定メンバ - なし

TeamProject¥Project Administrators - チーム プロジェクト内のすべての操作を実行できます。

-- ビルドの管理

-- 該当プロジェクトの削除

-- ビルド品質の編集

-- プロジェクト レベル情報の編集

-- テスト結果の発行

-- ビルドの開始

-- プロジェクト レベル情報の表示

-- ビルドの操作可能ストアへの書き込み

既定メンバ

-- Builtin¥Administrator

TeamProject¥Readers - チーム プロジェクトに読み取りアクセスします。

-- プロジェクト レベル情報の表示

既定メンバ

-- なし

[HOWTO]: Visual Studio Team Foundation Server でチーム ビルドを使用してコード分析を自動実行する方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、チーム ビルドのコード分析をオンにする方法を示しています。チーム ビルドの一環としてコード分析を自動的に実行し、その分析の結果をビルド結果としてレポートします。

目次

目的

- ビルドの一環としてコード分析を実行して、コード品質を検証する方法の習得。

概要

チーム ビルドを使用して、プロジェクト用のビルド タイプを定義することができます。それにより、ビルド サーバーがアプリケーションをコンパイルして、ネットワーク共有で利用可能になります。チーム ビルドのコード分析をオンにすれば、各ビルドが自動的にコード分析を実行し、その結果をビルド結果レポート ページで報告できます。この [HOWTO] では、コード分析がビルド ステップに含ま

れるようにチーム ビルドを構成するプロセスを 1 つずつ順に解説していきます。

ステップの概要

- ステップ 1 – ビルドのテスト
- ステップ 2 – ビルドのコード分析のオン
- ステップ 3 – コード分析のテスト

始める前に

チーム ビルドのコード分析をオンにする前に、以下の前提条件が揃っていることを確認する必要があります。

- ご使用の Team Foundation ユーザー ID が、ビルドを管理する許可を受けている。そのユーザー ID が許可を受けているかどうか不明の場合は、管理者に問い合わせてください。
- プロジェクトのチーム ビルドが既に存在していなければならない。これを調べるには、Visual Studio チーム エクスプローラのチーム ビルドをご覧ください。

ステップ 1 – ビルドのテスト

まず、チーム ビルドをテストして、問題がないことを確認してから、コード分析をオンにします。それには、次のようなステップを行います。

1. Visual Studio で [**チーム エクスプローラ**] を開きます。
2. チーム プロジェクトのノードを展開します。
3. [**チーム ビルド**] ノードを展開します。
4. 既存のチーム ビルドを右クリックし、[**チーム プロジェクトのビルド**] を選択します。ビルドが正常に完了したことを確認します。ビルドの中断がある場合や、ビルドを完了できない場合、エラーを訂正してから次のステップに進んでください。

ステップ 2 – ビルドのコード分析のオン

ビルドが順調に稼働することを確認したら、コード分析をオンにすることができます。

コード分析をオンにするには、次のようなステップを行います。

1. [**ソース管理エクスプローラ**] を開きます。
2. チーム プロジェクト フォルダを [**ソース管理エクスプローラ**] 内で展開します。
3. [**TeamBuildTypes**] フォルダを展開します。
4. コード分析をオンにするチーム ビルド フォルダを選択します。
5. **TFSBuild.proj** ファイルをソース管理からチェックアウトします。必要な場合は、先にフォルダに対して [**最新バージョンの取得**] を実行してください。
6. [**ソース管理エクスプローラ**] から、[**TFSBuild.Proj**] をダブルクリックして開きます。
7. プロジェクトの設定に関係なくすべてのプロジェクトでコード分析を実行したい場合、**<RunCodeAnalysis>** タグを **Always** に変更します。
8. プロジェクト設定に基づいて、プロジェクトごとにコード分析を実行したい場合、**<RunCodeAnalysis>** タグを **Default** に変更します。
9. プロジェクトに対して個別設定を使用している場合に、1 つのプロジェクトのコード分析をオンにしたいときは、次のようにします。
    a. ソリューションを Visual Studio で開きます。
    b. **[ソリューション エクスプローラ**] のプロジェクトを右クリックします。
    c. [**プロパティ**] を選択します。
    d. [**コード分析**] をクリックします。
    e. [**コード分析の有効化**] をオンにします。
    f. プロジェクトの .csproj ファイルをソース管理からチェックアウトします。
    g. [プロパティ] ウィンドウが表示されている間に、ツール バーの [**保存**] ボタンをクリックして、ファイルを保管します。
    h. プロジェクトの .csproj ファイルを元のソース管理にチェックインして戻します。

ステップ 3 – コード分析のテスト

チーム ビルドのコード分析をオンにしたら、それが順調に稼働することを確認するためのテストを行うことができます。ビルドのコード分析をテストにするには、次のようなステップを行います。

1. [**チーム エクスプローラ**] のビルド タイプを右クリックしてから、[**チーム プロジェクトのビルド**] をクリックします。
2. ビルドが完了したら、ビルド ログへのリンクをクリックします。

3. ビルド ログの末尾に、コード分析の何らかの警告が示されているはずです。警告 ID は、次のように、CA で始まっています。

- 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.
  (アセンブリ HelloWorldTest の有効な許可要求が見つかりませんでした。必ず SecurityAction.RequestMinimum を使用して、最小限のセキュリティ許可を指定する必要があります。)
- MSBUILD : warning : CA2210 : Microsoft.Design : Sign 'HelloWorldTest' with a strong name key.
  (厳密な名前キーを使用して HelloWorldTest に署名してください。)
- MSBUILD : warning : CA1014 : Microsoft.Design : 'HelloWorldTest' should be marked with CLSCompliantAttribute and its value should be true.
  (HelloWorldTest には CLSCompliantAttribute のマークが付いていて、その値は true でなければなりません。)

参考資料

- コード分析ツールの詳細は、「コード分析ツールを使用するためのガイドライン」(http://msdn2.microsoft.com/ja-jp/library/ms182023(VS.80).aspx) を参照してください。
- チーム ビルドの詳細は、「Team Foundation ビルドの概要」(http://msdn2.microsoft.com/ja-jp/library/ms181710(VS.80).aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server のカスタム レポートを作成する方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、新規のカスタム レポートを作成して、Team Foundation Server 内のチーム レポート ポータルに対してそれを発行する方法を説明します。

目次

目的

- Visual Studio でのレポート プロジェクトの作成
- レポート プロジェクトでの新しいカスタム レポートの作成
- レポート サーバーに対する新規レポートの発行

概要

VSTS に付属しているレポートは、SQL Server Reporting Services レポートです。このレポートに手を加えることもできますが、SQL Server 2005 クライアント ツールに付属している SQL Server 2005 Reporting Services Designer を使用して、Visual Studio (Business Intelligence Development Studio) 内で独自のカスタム レポートを作成することもできます。カスタム レポートを作成するには、Visual Studio でレポート プロジェクトを作成し、データ ソースを作成して、TFS リ

レーショナル データベースおよび OLAP データベースに接続します。

ステップの概要

- ステップ 1 – 新しいレポート プロジェクトの作成
- ステップ 2 – データ ソースの作成
- ステップ 3 – プロジェクトでの新規レポートの作成
- ステップ 4 – レポートの変更
- ステップ 5 – Team Foundation Server に対するレポートの配置
- ステップ 6 – レポートのテスト

始める前に

Team Foundation Server 用のレポートをカスタマイズするときは、以下の前提条件が揃っていることを事前に確認する必要があります。

- レポートをカスタマイズするときに使用する Business Intelligence Development Studio をマシンにインストールしておく必要があります。それがインストールされているかどうかを確認するには、Visual Studio を調べて、新しいプロジェクトの作成時に、[ビジネス インテリジェンス プロジェクト] がプロジェクトの種類にあるかどうかを確かめます。
- 使用するユーザー アカウントは、データ層サーバー上の Microsoft 分析サーバーの TfsWarehouseDataReaders のセキュリティ ロールのメンバでなければなりません。
- 使用するユーザー アカウントは、データ層上の TFSWarehouse データベースに対する管理者権限を持っていなければなりません。
- 使用するユーザー アカウントは、アプリケーション層サーバー上の SQL Server Reporting Services Publisher ロールのメンバでなければなりません。

ステップ 1 - 新しいレポート プロジェクトの作成

新規のレポート プロジェクトを作成し、新しいレポートをプロジェクトに追加してカスタマイズできるようにします。新規のレポート プロジェクトを Visual Studio で作成するには、次のようなステップを行います。

1. [**ファイル**] をクリックしてから [**新規作成**] をクリックし、次に [**プロジェクト**] をクリックします。
2. プロジェクトの種類で [**ビジネス インテリジェンス プロジェクト**] を選択します。
3. [**レポート サーバー プロジェクト**] テンプレートを選択します。
4. [**プロジェクト名**] および [**場所**] を設定してから、[OK] をクリックします。

ステップ 2 - データ ソースの作成

カスタマイズしたレポートを編集して発行するには、Team Foundation Server のデータ ウェアハウスおよび OLAP キューブ用のデータ ソースを追加する必要があります。このデータ ソースを Visual Studio プロジェクトに追加し終わったら、サーバーからレポートにデータをプルすることができます。

ウェアハウス データ ソースを作成するには:

1. Visual Studio の [**ソリューション エクスプローラ**] で、[**共有データ ソース**] を右クリックしてから、[**新しいデータ ソースの追加**] をクリックします。
2. [**一般**] タブの [**名前**] テキスト ボックスに **TfsReportDS** と入力します。
3. [**種類**] コンボ ボックスで、[**Microsoft SQL Server**] を選択します。
4. [**編集...**] ボタンをクリックします。
5. データ層サーバー名を入力します。
6. データベース **TFSWarehouse** を選択します。
7. [**OK**] ボタンを 2 回クリックして、データ ソースを追加します。

OLAP データ ソースを作成するには:

1. [**ソリューション エクスプローラ**] で、[**共有データ ソース**] を右クリックしてから、[**新しいデータ ソースの追加**] をクリックします。
2. [**一般**] タブの [**名前**] テキスト ボックスに **TfsOlapReportDS** と入力します。
3. [**種類**] コンボ ボックスで、[**Microsoft SQL Server Analysis Services**] を選択します。
4. [**編集...**] ボタンをクリックします。
5. データ層サーバー名を入力します。
6. データベース **TFSWarehouse** を選択します。

7. [**OK**] ボタンを 2 回クリックして、データ ソースを追加します。

ステップ 3 - プロジェクトでの新規レポートの作成

プロジェクトへのデータ ソースの追加が完了したら、次に新規レポートを追加することができます。プロジェクトに新規レポートを追加してカスタマイズするには、次のようなステップを行います。

1. [**ソリューション エクスプローラ**] で、[**レポート**] を右クリックしてから、[**追加**] -> [**新しい項目...**] を選択します。
2. [**レポート**] テンプレートを選択します。
3. レポートに名前を付けてから、[**ok**] をクリックします。

ステップ 4 – レポートの変更

プロジェクトにレポートを追加し終わったら、次のようにしてそれを修正することができます。

1. [**レポート デザイナ**] が自動的に開かない場合、[**ソリューション エクスプローラ**] で、修正するレポートをダブル クリックして開きます。
2. [**データセット**] ドロップダウンを選択してから、**<新しいデータセット...>** を選択します。
3. データセットに、たとえば TestDataSet などの名前を付けます。
4. [**TfsOlapReportDS (共用)**] を選択します。
5. [**OK**] をクリックします。
6. [**データセット**] ドロップダウンのすぐ下の [**ビルド**] の横の [**...**] ボタンをクリックして、Team System を選択します。

これで、レポートを修正する準備ができました。修正するには、[**データセット**] ツリーのメジャーとディメンションを [**クエリ ペイン**] および [**フィルタ ペイン**] にドラッグします。レポートのレイアウトを修正するには、[**レイアウト**] タブをクリックします。[**プレビュー**] タブをクリックすれば、レポートのプレビューを表示することができます。

ステップ 5 - Team Foundation Server に対するレポートの配置

レポートに修正を加えた後、次のようなステップを行って、チーム プロジェクトのレポート ポータルに対してそれを配置することができます。

1. [**ソリューション エクスプローラ**] 内のレポート プロジェクトを右クリックしてから、[**プロパティ**] をクリックします。
2. [**OverwriteDataSources**] が **false** に設定されていることを確認します。
3. [**TargetDataSourceFolder**] をご自分のチーム プロジェクトの名前に変更します。例: **TargetDataSourceFolder** = TestProject。
4. [**TargetReportFolder**] をご自分のチーム プロジェクトの名前に変更します。例: **TargetDataSourceFolder** = TestProject。
5. [**TargetDataSourceFolder**] を http://<データ層のサーバー名>/reportserver に変更します。例: **TargetDataSourceFolder** = http://tfsrtm/reportserver。
6. [**OK**] をクリックします。
7. [**ソリューション エクスプローラ**] 内の rdl ファイルを右クリックしてから、[**配置**] をクリックします。
8. [**出力ペイン**] を調べて、正常に完了したことを確認します。

ステップ 6 – レポートのテスト

チーム プロジェクトのレポート サーバーに対してレポートを発行し終わったら、次のようにしてそのテストを行って、正常に配置されたことを確認することができます。

1. チーム プロジェクトを右クリックして、[**プロジェクト ポータルの表示**] を選択します。
2. ポータル Web サイトの左側のクイック起動バー内にある [**レポート**] をクリックします。
3. 作成したレポートを選択します。
4. レポートが、想定したとおりであることを確認します。

参考資料

- レポート プロジェクトの処理方法を説明したチュートリアルは、「Reporting Services のチュートリアル」(http://msdn2.microsoft.com/ja-jp/library/ms170246.aspx) を参照してください。
- レポートの編集に関する MSDN 記事は、「方法: レポート デザイナでレポートを編集する」(http://msdn2.microsoft.com/ja-jp/library/ms244655(VS.80).aspx) を参照してください。
- データ層におけるセキュリティ ロールに関する詳細は、「Analysis Services を使用したアクセスのセキュリティ保護」(http://msdn2.microsoft.com/ja-jp/library/ms174839.aspx) を参照してく

ださい。

- アプリケーション層におけるセキュリティ ロールに関する詳細は、「Reporting Services の保護」(http://msdn2.microsoft.com/ja-jp/library/ms157198.aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server のリスク オーバータイム レポートを作成する方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、**リスク**作業項目の一定期間中の傾向を示す新規のレポートを作成してから、Team Foundation Server 内のチーム レポート ポータルに対してそれを発行する方法を説明します。

目次

目的

- Visual Studio でのレポート プロジェクトの作成
- レポート プロジェクトでの新しいリスク オーバー タイム レポートの作成
- レポート サーバーに対するリスク オーバー タイム レポートの発行

概要

VSTS に付属しているレポートは、SQL Server Reporting Services レポートです。このレポートに手を加えることもできますが、SQL Server 2005 クライアント ツールに付属している SQL Server 2005 Reporting Services Designer を使用して、Visual Studio (Business Intelligence Development Studio) 内で独自のカスタム レポートを作成することもできます。カスタム レポートを作成するには、Visual Studio でレポート プロジェクトを作成し、データ ソースを作成して、TFS リレーショナル データベースおよび OLAP データベースに接続します。この [HOWTO] では、リスク オーバー タイム レポートというシンプルなレポートを最初から作成します。このレポートは、ある一定期間中のリスク作業項目がいくつあるかを示します。

ステップの概要

- ステップ 1 – 新しいレポート プロジェクトの作成
- ステップ 2 – データ ソースの作成
- ステップ 3 – プロジェクトでの新規レポートの作成
- ステップ 4 – レポートの変更
- ステップ 5 – Team Foundation Server に対するレポートの配置
- ステップ 6 – レポートのテスト

始める前に

Team Foundation Server 用のレポートをカスタマイズするときは、以下の前提条件が揃っていることを事前に確認する必要があります。

- レポートをカスタマイズするときに使用する Business Intelligence Development Studio をマシンにインストールしておく必要があります。それがインストールされているかどうかを確認するには、

Visual Studio を調べて、新しいプロジェクトの作成時に、[ビジネス インテリジェンス プロジェクト] がプロジェクトの種類にあるかどうかを確かめます。

- 使用するユーザー アカウントは、データ層サーバー上の Microsoft 分析サーバーの TfsWarehouseDataReaders のセキュリティ ロールのメンバでなければなりません。
- 使用するユーザー アカウントは、データ層上の TFSWarehouse データベースに対する管理者権限を持っていなければなりません。
- 使用するユーザー アカウントは、アプリケーション層サーバー上の SQL Server Reporting Services Publisher ロールのメンバでなければなりません。

ステップ 1 - 新しいレポート プロジェクトの作成

新規のレポート プロジェクトを作成し、新しいレポートをプロジェクトに追加してカスタマイズできるようにします。新規のレポート プロジェクトを Visual Studio で作成するには、次のようなステップを行います。

1. [**ファイル**] をクリックしてから [**新規作成**] をクリックし、次に [**プロジェクト**] をクリックします。
2. プロジェクトの種類で [**ビジネス インテリジェンス プロジェクト**] を選択します。
3. [**レポート サーバー プロジェクト**] テンプレートを選択します。
4. [**プロジェクト名**] および [**場所**] を設定してから、[OK] をクリックします。

ステップ 2 - データ ソースの作成

カスタマイズしたレポートを編集して発行するには、Team Foundation Server のデータ ウェアハウスおよび OLAP キューブ用のデータ ソースを追加する必要があります。このデータ ソースを Visual Studio プロジェクトに追加し終わったら、サーバーからレポートにデータをプルすることができます。

ウェアハウス データ ソースを作成するには:

1. Visual Studio の**ソリューション エクスプローラ**で、[**共有データ ソース**] を右クリックしてから、[**新しいデータ ソースの追加**] をクリックします。
2. [**一般**] タブの [**名前**] テキスト ボックスに **TfsReportDS** と入力します。
3. [**種類**] コンボ ボックスで、[**Microsoft SQL Server**] を選択します。

4. [**編集...**] ボタンをクリックします。
5. データ層サーバー名を入力します。
6. データベース **TFSWarehouse** を選択します。
7. [**OK**] ボタンを 2 回クリックして、データ ソースを追加します。

OLAP データ ソースを作成するには:

1. [**ソリューション エクスプローラ**] で、[**共有データ ソース**] を右クリックしてから、[**新しいデータ ソースの追加**] をクリックします。
2. [**一般**] タブの [**名前**] テキスト ボックスに **TfsOlapReportDS** と入力します。
3. [**種類**] コンボ ボックスで、[**Microsoft SQL Server Analysis Services**] を選択します。
4. [**編集...**] ボタンをクリックします。
5. データ層サーバー名を入力します。
6. データベース **TFSWarehouse** を選択します。
7. [**OK**] ボタンを 2 回クリックして、データ ソースを追加します。

ステップ 3 - プロジェクトでの新規レポートの作成

プロジェクトへのデータ ソースの追加が完了したら、次に新規レポートを追加することができます。プロジェクトに新規レポートを追加してカスタマイズするには、次のようなステップを行います。

1. [**ソリューション エクスプローラ**] で、[**レポート**] を右クリックしてから、[**追加**] -> [**新しい項目...**] を選択します。
2. [**レポート**] テンプレートを選択します。
3. レポートに名前を付けてから、[**ok**] をクリックします。

ステップ 4 – レポートの変更

プロジェクトにレポートを追加し終わったら、次のようにしてそれを修正することができます。

1. [**レポート デザイナ**] が自動的に開かない場合、[**ソリューション エクスプローラ**] で、修正するレポートをダブル クリックして開きます。
2. [**データセット**] ドロップダウンを選択してから、**<新しいデータセット...>** を選択します。
3. データ セットに、たとえば TestDataSet などの名前を付けます。

4. [**TfsOlapReportDS (共用)**] を選択します。
5. [**OK**] をクリックします。
6. [**データセット**] ドロップダウンのすぐ下の [**ビルド**] の横の [**...**] ボタンをクリックして、Team System を選択します。
7. [**データセット ツリー**] 内の [**メジャー**] を展開します。
8. [**データセット ツリー**] 内の [**現在の作業項目**] を展開します。
9. [**現在の作業項目のカウント**] をメイン クエリ ウィンドウにドラッグします。
10. [**データセット ツリー**] 内の [**メジャー**] を折りたたみます。
11. 下方の [**チーム プロジェクト**] までスクロールし、これを [**ディメンション グリッド**] にドラッグします。
12. [**ディメンション グリッド**] で、[**フィルタ式**] セルをクリックして、チーム プロジェクトの名前を選択します。これで、結果がフィルタされて、該当するチーム プロジェクトにのみ絞られます。
13. [**データセット ツリー**] 内の [**作業項目**] ディメンションを展開します。
14. [**データセット ツリー**] から [**ディメンション グリッド**] に、[**System_WorkItemType**] をドラッグします。
15. [**データセット ツリー**] からメイン クエリ ウィンドウに [**System_WorkItemType**] をドラッグし、作業項目カウント列の前にドロップします。
16. [**ディメンション グリッド**] 内の [**フィルタ式**] セルをクリックし、種類で [**リスク**] を選択します。これで、結果がフィルタされて、リスク作業項目の種類のみに絞られます。
17. [**データセット ツリー**] 内の [**日付**] ディメンションを展開します。
18. [**日付**] ディメンション値をメイン クエリ ウィンドウにドラッグし、作業項目種類列の前にドロップします。
19. [**レイアウト**] タブをクリックします。
20. [**ツールボックス**] ウィンドウを開きます。
21. [**グラフ**] 項目を [**ツールボックス**] からレイアウト グリッドにドラッグします。
22. グリッドに収まるようにグラフのサイズを調整します。
23. グラフを右クリックし、[**グラフの種類**] -> [**線**] -> [**平滑線**] を選択します。
24. [**データセット ペイン**] を開きます。
25. TestDataSet などの作成したデータ セットを展開します。
26. グラフを強調表示して、データ、系列、およびカテゴリ ドロップ ターゲットを表示します。

27. [**Current_Work_Item_Count**] を [**ここにデータ フィールドをドロップします**] ドロップ ターゲット ボックスにドロップします。
28. [**Work_Item_Type**] を [**ここに系列フィールドをドロップします**] ドロップ ターゲット ボックスにドロップします。
29. [**Date**] を [**ここにカテゴリ フィールドをドロップします**] ドロップ ターゲット ボックスにドロップします。
30. グラフを右クリックして、[**プロパティ**] を選択します。
31. グラフのタイトルを入力します。
32. [**プレビュー**] タブをクリックし、レポートを表示して確かめます。

ステップ 5 - Team Foundation Server に対するレポートの配置

リスク オーバー タイム レポートを作成した後、次のようなステップを行って、チーム プロジェクトのレポート ポータルに対してそれを配置することができます。

1. [**ソリューション エクスプローラ**] 内のレポート プロジェクトを右クリックしてから、[**プロパティ**] をクリックします。
2. [**OverwriteDataSources**] が **false** に設定されていることを確認します。
3. [**TargetDataSourceFolder**] をご自分のチーム プロジェクトの名前に変更します。例: **TargetDataSourceFolder** = TestProject。
4. [**TargetReportFolder**] をご自分のチーム プロジェクトの名前に変更します。例: **TargetDataSourceFolder** = TestProject。
5. [**TargetDataSourceFolder**] を http://<データ層のサーバー名>/reportserver に変更します。例: **TargetDataSourceFolder** = http://tfsrtm/reportserver。
6. [**OK**] をクリックします。
7. [**ソリューション エクスプローラ**] 内の rdl ファイルを右クリックしてから、[**配置**] をクリックします。
8. [**出力ペイン**] を調べて、正常に完了したことを確認します。

ステップ 6 – レポートのテスト

チーム プロジェクトのレポート サーバーに対してレポートを発行し終わったら、次のようにしてその

テストを行って、正常に配置されたことを確認することができます。

1. チーム プロジェクトを右クリックして、[**プロジェクト ポータルの表示**] を選択します。
2. ポータル Web サイトの左側のクイック起動バー内の [**レポート**] をクリックします。
3. 作成したレポートを選択します。
4. レポートが、想定していたとおりであることを確認します。

参考資料

- レポート プロジェクトの処理方法を説明したチュートリアルは、「Reporting Services のチュートリアル」(http://msdn2.microsoft.com/ja-jp/library/ms170246.aspx) を参照してください。
- レポートの編集に関する MSDN 記事は、「方法: レポート デザイナでレポートを編集する」(http://msdn2.microsoft.com/ja-jp/library/ms244655(VS.80).aspx) を参照してください。
- データ層におけるセキュリティ ロールに関する詳細は、「Analysis Services を使用したアクセスのセキュリティ保護」(http://msdn2.microsoft.com/ja-jp/library/ms174839.aspx) を参照してください。
- アプリケーション層におけるセキュリティ ロールに関する詳細は、「Reporting Services の保護」(http://msdn2.microsoft.com/ja-jp/library/ms157198.aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server のカスタム チェックイン ポリシーを作成するステップ

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、Team Foundation Server のカスタム チェックイン ポリシーの作成、登録、および適用のステップを 1 つずつたどります。チェックイン ポリシーを使用すると、開発者がソース ファイルのチェックインを試みるたびに、ソース ファイルとチェックインが一連の指定基準を満たしていることを確認するためのルールを実行することができます。たとえば、この [HOWTO] では、コ

メント チェックイン カスタム ポリシーを使用して、すべてのチェックインにチェックイン コメントが付随するようにします。カスタム チェックイン ポリシーを実装するには、**PolicyBase** を派生元とするクラスを作成し、**IPolicyDefinition** および **IPolicyEvaluation** を実装します。ポリシー アセンブリを Windows レジストリに登録して、ポリシーをチーム プロジェクトに対して適用します。

目次

目的

- カスタム チェックイン ポリシーの作成、登録、および適用の方法の習得

概要

チェックイン ポリシーは、ソース管理へのファイルのチェックイン時に、その都度制約を適用します。Team Foundation Server には、多数のチェックイン ポリシーが用意されています。たとえば、単体テストが実行されて合格したかどうかをチェックするポリシーや、静的コード分析を実行して、そのコードがコーディング標準および .NET ガイドラインを順守しているかどうかの確認を行うポリシー、作業項目がチェックインに関連付けられているかどうかのチェックを行うポリシーなどがあります。また、Team Foundation Power Tool には、さらに別の多数のチェックイン ポリシーが用意されています。この [HOWTO] では、カスタム ポリシーを作成、登録、および適用する方法を示します。サンプル ポリシーでは、開発者は、ファイルのチェックインごとにチェックイン コメントを必ず提供するよ

うになっています。

始める前に

注意点として、チェックイン ポリシーを作成するには、**操作**セキュリティ設定を**許可**に設定する必要があります。

ステップの概要

- ステップ 1 - カスタム ポリシー クラスの作成およびビルド
- ステップ 2 - Windows レジストリへのカスタム ポリシーの登録
- ステップ 3 - カスタム ポリシーの適用
- ステップ 4 - カスタム ポリシーの検証

ステップ 1 - カスタム ポリシー クラスの作成およびビルド

このステップでは、**Microsoft.TeamFoundation.VersionControl.Client** 名前空間内の **PolicyBase** からの派生によって、カスタム ポリシー クラスを作成します。この基本クラスからの派生によって、**IPolicyDefinition** および **IPolicyEvaluation** インターフェイスがクラスに実装されます。以下に示されているサンプル ポリシー コードでは、開発者は、ソース ファイルのチェックインのたびに、チェックイン コメントの提供を義務付けられます。

1. Visual Studio を使用して、新しい C# クラス ライブラリ プロジェクトを作成します。
2. System.Windows.Forms.dll へのアセンブリ参照を追加します。このアセンブリは、メッセージ ボックスを表示するときに使用します。
3. Microsoft.TeamFoundation.VersionControl.Client.dll へのアセンブリ参照を追加します。既定では、これは以下のフォルダにインストールされます。

¥Program Files¥Visual Studio 2005 Team Foundation Server¥Tools

4. スケルトン クラス コードの実装を、以下のソースに置き換えます。クラスは **PolicyBase** から派生し、シリアル化可能のマークが付けられていることに注意してください。

```
using System;
using System.Windows.Forms;
using Microsoft.TeamFoundation.VersionControl.Client;
```

```
[Serializable]
public class CheckForCommentsPolicy : PolicyBase
{
public override string Description
{
get { return "Remind users to add meaningful comments to their
checkins"; }
}
// これは、ソース管理サーバー上でポリシー定義と一緒に保管される
// 文字列です。ユーザーがポリシー プラグインをインストールしていない場合、この文字列が
// 表示されます。これを使用して、ポリシー プラグインをどのようにインストールする必要があるかを
// ユーザーに説明することができます。
public override string InstallationInstructions
{
get { return "To install this policy, read
InstallInstructions.txt."; }
}
// この文字列は、ポリシーのタイプを識別します。これは、チーム プロジェクトに新しいポリシーを追加するときに、
// ポリシー リストに表示されます。
public override string Type
{
get { return "Check for Comments Policy"; }
}
// この文字列は、ポリシーのタイプの説明です。これは、チェックイン ポリシーの追加ダイアログ ボックスで
// ポリシーを選択するときに表示されます。
public override string TypeDescription
{
get { return "This policy will prompt the user to decide whether or
not they should be allowed to check in."; }
}
// 新しいチェックイン ポリシーを作成するときや、既存のチェックイン ポリシー
// を編集するときに、このメソッドがポリシー フレームワークによって呼び出されます。
// これを使用して、このポリシーに固有の UI を表示することができます。
// それによって、ユーザーがポリシーのパラメータを変更できるようになります。
public override bool Edit(IPolicyEditArgs args)
{
// カスタム構成は一切必要ありません。
return true;
}
// このメソッドは、実際のポリシー評価を実行します。
// ポリシーを評価する必要が生じた様々な時点で、これがポリシー フレームワークによって
```

```
// 呼び出されます。この例では、現在の障害リストの無効化につながったかも
// しれない様々な非同期イベントが発生したときに、
// このメソッドが起動されます。
public override PolicyFailure[] Evaluate()
{
string proposedComment = PendingCheckin.PendingChanges.Comment;
if (String.IsNullOrEmpty(proposedComment))
{
return new PolicyFailure[] {
new PolicyFailure("Please provide some comments about your
checkin", this) };
}
else
{
return new PolicyFailure[0];
}
}
// ユーザーが UI 中のポリシー障害をダブルクリックすると、このメソッドが呼び出されます。
// この場合、何らかのコメントを提供するようユーザーに対して指示する
// メッセージが表示されます。
public override void Activate(PolicyFailure failure)
{
MessageBox.Show("Please provide comments for your checkin.", "How to
fix your policy failure");
}
// UI 内でポリシー障害がアクティブになっているときに、ユーザーが F1 を押すと、
// このメソッドが呼び出されます。この例では、メッセージ ボックスが表示されます。
public override void DisplayHelp(PolicyFailure failure)
{
MessageBox.Show("This policy helps you to remember to add comments
to your checkins.", "Prompt Policy Help");
}
}
```

ステップ 2 - Windows レジストリへのカスタム ポリシーの登録

このステップでは、Windows レジストリにエントリを追加します。それによって、該当するポリシーが、[**チェックイン ポリシーの追加**] ダイアログ ボックスに表示されます。ポリシー アセンブリを参照する必要のあるすべてのコンピュータにそのポリシー アセンブリをインストールする必要があるこ

とに注意してください。それには、ポリシーをチーム プロジェクトに関連付ける必要のあるチーム プロジェクトの管理者のコンピュータや、ポリシーが実際に評価される場所であるチーム メンバのすべてのコンピュータなどが含まれます。

重要事項: 開発者がファイルをチェックインするときに、ポリシーはクライアント上で評価されます。

カスタム ポリシー クラスを登録するには:

1. Regedit.exe を開始し、以下のキーを見つけ出します。

HKEY_LOCAL_MACHINE¥Software¥Microsoft¥VisualStudio¥8.0¥TeamFoundation¥SourceControl¥Checkin Policies

登録済みのポリシーは、右側のペインにリストされています。

2. 右側のペインを右クリックし、[**新規**] をポイントして、[**文字列値**] をクリックします。

3. 上記の例では、カスタム ポリシー DLL **CheckForCommentsPolicy** を (DLL 拡張子なしで) 入力します。

重要事項: 新しい文字列名は、DLL 拡張子を省いた DLL ファイル名に正確に一致していなければなりません。

4. 新しい文字列値をダブルクリックし、その値を、カスタム ポリシーを収めた .dll の完全修飾パスおよびファイル名に設定します。

ステップ 3 - カスタム ポリシーの適用

このステップでは、カスタム ポリシーをチーム プロジェクトに追加します。それによって、開発者がこのチーム プロジェクトにファイルをチェックインするたびに、そのポリシーが必ず評価されることになります。

カスタム ポリシーを適用するには:

1. [**チーム エクスプローラ**] で、チーム プロジェクトを右クリックし、[**チーム プロジェクトの設定**] をポイントして、[**ソース管理**] をクリックします。

2. [**チェックイン ポリシー**] タブをクリックしてから、[**追加**] をクリックします。

3. カスタム ポリシーの [**Check for Comments Policy**] を選択し、[**OK**] をクリックしてからもう一度 [**OK**] をクリックします。

これで、開発者がこのチーム プロジェクトにファイルをチェックインするたびに、このポリシーが必ず適用されることになります。

ステップ 4 - カスタム ポリシーの検証

このステップでは、カスタム ポリシーが動作することを確認するために、ソース ファイルをチェックインします。

1. ソース ファイルに変更を加えてから、チェックイン コメントを添付しないで、このファイルをチェックインします。
2. ポリシー ルールが満たされないので、そのチェックインは阻止されることを確認します。
3. 何らかのコメントを追加してから、チェックインを完了します。コメントを添付したので、チェックインは正常に完了するはずであり、ポリシー障害の通知も表示されません。

参考資料

- チェックイン ポリシーのカスタマイズ方法を学習するには、「チュートリアル: チェックイン ポリシーとチェックイン メモのカスタマイズ」(http://msdn2.microsoft.com/ja-jp/library/ms181281(VS.80).aspx) を参照してください。
- 特定パターンのチェックインを禁止するサンプル コードの詳細は、「Checkin Policy to Disallow Certain Patterns」(http://blogs.msdn.com/jmanning/archive/2006/02/02/523125.aspx) を参照してください。
- チェックインでのコメントを義務付けるサンプル コードの詳細は、「Sample Checkin Policy: Make Sure the Comment Isn't Empty」(http://blogs.msdn.com/jmanning/archive/2006/01/21/515858.aspx) を参照してください。
- 新しいチェックイン ポリシーを登録する方法の詳細は、「I've Made a New Check-In Policy! How Do I Add It?」(http://blogs.msdn.com/jmanning/archive/2006/02/07/526778.aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server でソース ツリーを作成するステップ

適用対象

• Visual Studio Team Foundation Server

概要

この [HOWTO] では、Team Foundation Server で新しいソース コード ツリー構造を作成するプロセスを 1 つずつたどります。この [HOWTO] の目的は、ソース ツリーを作成するのに必要な具体的なステップを習熟することにあります。

目次

目的

• Team Foundation Server でのソース ツリーの作成方法の習得

概要

[ソリューション エクスプローラ] でソリューションを右クリックしてから [**ソリューションをソース管理に追加**] をクリックすれば、ソリューションをすぐにソース管理に追加できますが、この方法では、ソース管理内でソース ツリー構造を明示的に構築できません。ソース管理でソース ツリー構造を明示的に定義することによって、最上位フォルダの下のソースの配置を調整したり、別の最上位フォルダを使用して、メインのソース ベースのほかに、稼働リリースの開発や保守に使用する分岐などの、分岐したソース ベースを持たせておくことができます。
この [HOWTO] では、ソース管理のツリー構造を明示的に作成するのに必要なステップを示していま

す。

ステップの概要

- ステップ 1 - 新しいチーム プロジェクトの作成
- ステップ 2 - ワークスペース マッピングの作成
- ステップ 3 - ソース管理でのフォルダ構造の作成
- ステップ 4 - ソース ツリーへのソース コードの追加

ステップ 1 - 新しいチーム プロジェクトの作成

このステップでは、既定の設定値を使用して新しいチーム プロジェクトを作成します。

新しいチーム プロジェクトを作成するには:

1. [**チーム エクスプローラ**] で、Team Foundation Server を右クリックし、[**新しいチーム プロジェクト...**] をクリックします。
2. [**新しいチーム プロジェクト**] ダイアログ ボックスで、**MyTeamProject1** などのプロジェクト名を入力し、[**次へ**] をクリックします。
3. [**プロセス テンプレートの選択**] ページで、既定の [**MSF for Agile Software Development - v4.0**] をそのまま残して、[**次へ**] をクリックします。
4. [**プロジェクト ポータルの設定を指定します**] ページで、チーム プロジェクト ポータル名を **MyTeamProject1** のまま残し、チーム プロジェクト ポータルの説明を入力してから、[**次へ**] をクリックします。
5. [**ソース管理の設定を指定します**] ページで、既定オプション [**空のソース管理フォルダの作成**] を選択済みのまま残してから、[**次へ**] をクリックします。
6. [**完了**] をクリックして、プロジェクトを作成します。

選択された、空のソース管理ノードを持ったプロセス テンプレートを使用して、新しいチーム プロジェクトが Team Foundation Server 上に作成されます。

ステップ 2 - ワークスペース マッピングの作成

このステップでは、TFS サーバーおよびクライアント上のフォルダ構造間のマッピングを定義するワ

ークスペース マッピングを作成します。これは、ソース ツリー構造の作成を可能にするために行う必要があります。まず、ソース ツリー構造がローカルのワークスペース内に作成された後、Team Foundation Server へのチェックインを実行する必要があります。

以下の 2 通りの方法のいずれかでワークスペース マッピングを作成することができます。

- **ワークスペース マッピングを明示的に設定する。**
- **チーム プロジェクトに対して get 操作を実行する。**

ワークスペース マッピングを明示的に設定するには:

1. Visual Studio で [**ファイル**] メニューから、[**ソース管理**] をポイントして、[**ワークスペース**] をクリックします。
2. [**ワークスペースの管理**] ダイアログ ボックスで、コンピュータ名を選択し、[**編集**] をクリックします。
3. [**ワークスペースの編集**] ダイアログ ボックスの [**作業フォルダ**] リストで、[**新しい作業フォルダを入力するには、ここをクリックします**] をクリックします。
4. 省略記号ボタンをクリックし、たとえば、**MyTeamProject1** といった、チーム プロジェクト名を選択して、[**OK**] をクリックします。
5. ローカル フォルダ セルをクリックして、別の省略記号ボタンを表示します。
6. [**ローカル フォルダ**] の下の省略記号ボタンをクリックし、チーム プロジェクトの格納先として、たとえば C:¥DevProjects¥MyTeamProject1 などの、開発コンピュータ上のローカル フォルダを参照して選択します。
7. [**OK**] をクリックしてから、もう一度 [**OK**] をクリックして、[**ワークスペースの編集**] ダイアログ ボックスを閉じます。
8. 1 つ以上の作業フォルダが変更されたことが Microsoft Visual Studio メッセージ ボックスで通知されたら、それに対する応答として、[**OK**] をクリックします。
9. [**閉じる**] をクリックして、[**ワークスペースの管理**] ダイアログ ボックスを閉じます。

チーム プロジェクトに対して Get 操作を実行するには:

1. [**チーム エクスプローラ**] で、チーム プロジェクト ノード **MyTeamProject1** を展開します。
2. チーム プロジェクトの下の [**ソース管理**] をダブルクリックします。
3. [**ソース管理エクスプローラ**] で、ルート フォルダ **MyTeamProject1** を右クリックし、[**最新のバージョンを取得**] をクリックします。
4. [**フォルダの参照**] ダイアログ ボックスで、たとえば C:¥DevProjects¥MyTeamProject1 などの

ローカル パスを選択し、[**OK**] をクリックします。これで、Team Foundation Server 内のチーム プロジェクトのルート フォルダは、コンピュータ上のローカル パスにマップされます。

ステップ 3 - ソース管理でのフォルダ構造の作成

このステップでは、各自のストラテジとプロジェクトの要件に応じて、ソース管理フォルダ構造をサーバー上に作成します。一般的にこれは、/Main/Source フォルダ構造で始まっているはずです。それによって、その後、**Main** と同じレベルでの **Development** および **Maintenance** 分岐の作成が可能になります。たとえば、**Maintenance** フォルダは、ソフトウェア リリースに対応する分岐コードを格納するのに使用します。**Development** フォルダには、作業中の開発分岐が格納されます。

/Main

/Source

/MyApp1　MyApp1.sln を格納します。

/Source　コンテナ フォルダ

/ClassLibrary1　ClassLibrary1.csproj を格納します。

/MyApp1Web　Default.aspx を格納します。

/UnitTests　単体テスト プロジェクトを格納します。

/ClassLibrary1Tests　ClassLibrary1 用のテスト プロジェクト

/MyApp1WebTests　MyApp1Web 用のテスト プロジェクト

/Build　ビルド出力 (バイナリ) を格納します。

/Docs　デザイン ドキュメントなどを格納します。

/Scripts　ビルド スクリプトを格納します。

/TestCases　テスト ケース ドキュメントを格納します。

/Development

/FeatureBranch1

/Source

/MyApp1

/Source

/MyApp1Web

/ClassLibrary1

/UnitTests

/ClassLibrary1Tests

/MyApp1WebTests

/FeatureBranch2

/**Maintenance**

/Release1

/MyApp1

/Source

/ClassLibrary1

/MyApp1Web

/UnitTests

/ClassLibrary1Tests

/MyApp1WebTests

/Release 1.1

/Release 1.2

サーバー上でフォルダ構造を作成するには:

1. [**チーム エクスプローラ**] で、チーム プロジェクト ノード **MyTeamProject1** を展開します。
2. チーム プロジェクトの下の [**ソース管理**] をダブルクリックします。
3. [**ソース管理エクスプローラ**] で、ルート ノードを選択し、[**ローカル パス:**] ペイン内を右クリックし、[**フォルダの新規作成**] をクリックします。
4. 名前 **Main** を入力し、Enter キーを押します。
5. Main の下に **Source** フォルダを作成します。
6. 上記のステップを繰り返して、**Development** および **Maintenance** フォルダを含む、必要な他のすべてのルート フォルダを作成します。
7. ツリー構造の作成が完了したら、[**ソース管理エクスプローラ**] 内の **MyTeamProject1** ルート ノードを右クリックし、[**保留中の変更のチェックイン**] をクリックします。
8. [**チェックイン**][**ソース ファイル**] の [**ワークスペース**] ダイアログ ボックスで、チェックインしたいフォルダを選択し、コメントを追加してから、[**チェックイン**] をクリックします。これで、ローカル フォルダ構造が作成されて、構造が TFS ソース管理に追加されます。

ステップ 4 - ソース ツリーへのソース コードの追加

このステップでは、ローカル ドライブからサーバー上のソース管理ツリーに、ソース コードを追加します。この例では、新しい Web アプリケーションおよびクラス ライブラリ プロジェクトを作成し、それらをソース管理に追加します。

新しい Visual Studio ソリューション ファイルを作成するには:

1. [**ファイル**] メニューで、[**新規作成**] をポイントして、[**プロジェクト**] をクリックします。
2. [**その他のプロジェクトの種類**] を展開し、[**Visual Studio ソリューション**] をクリックします。
3. [**テンプレート**] ペインで [**空のソリューション**] を選択します。
4. [**プロジェクト名**] テキスト ボックスに **MyApp1** と入力し、[**場所**] テキスト ボックスに C:¥DevProjects¥MyTeamProject1¥Main¥Source と入力します。
5. [**OK**] をクリックします。Visual Studio によって 新しいソリューションが作成されて、ソリューション (.sln) ファイルが C:¥DevProjects¥ MyTeamProject1¥Main¥Source¥MyApp1 フォルダに置かれます。

新規の Web サイトをソリューションに追加するには:

1. [ソリューション エクスプローラ] 内のソリューションを右クリックし、[**追加**] をポイントして、[**新しい Web サイト**] をクリックします。
2. [**テンプレート**] リストで [**ASP.NET Web サイト**] を選択し、[**場所**] を [**ファイル システム**] にしてし、パスをC:¥DevProjects¥MyTeamProject1¥Main¥Source¥MyApp1¥Source¥MyApp1Web にします。
3. [**OK**] をクリックします。Visual Studio が Web サイトを作成します。

新規のクラス ライブラリ プロジェクトをソリューションに追加するには:

1. [ソリューション エクスプローラ] 内のソリューションを右クリックし、[**追加**] を指して、[**新しいプロジェクト**] をクリックします。
2. [**Visual C#**] を [**プロジェクトの種類**] リストで選択し、[**クラス ライブラリ**] を [**テンプレート**] リストで選択します。
3. 名前は **ClassLibrary1** のまま変更しないで、[**場所**] を C:¥DevProjects¥MyTeamProject1¥Main¥Source¥MyApp1¥Source に設定します。
4. [**OK**] をクリックします。Visual Studio が新しいプロジェクト構造を作成します。ローカル ファ

イル構造は次のようになっているはずです。

ソース管理にソリューションを追加するには:

1. [ソリューション エクスプローラ] 内のソリューションを右クリックし、[**ソリューションをソース管理に追加**] をクリックします。
2. そのソリューションと 2 つのプロジェクトが Team Foundation のソース管理に追加されます。

ソース管理のツリー構造は次のようになるはずです。

参考資料

- ワークスペースの作成に関する詳細は、「方法: ワークスペースを作成する」(http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx) を参照してください。

[HOWTO]: Process Editor を使用してプロセス テンプレートをカスタマイズする方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、プロセス テンプレートを修正して、チームのニーズに対する適合性を高める方法を説明します。プロセス テンプレートは、チーム プロジェクトの初期プロセス設定を定義します。プロセス テンプレートをカスタマイズすることによって、プロジェクト ポータル上で使用可能なテンプレートの種類、ソース コード管理のチェックイン メモ、新しい作業項目の種類とクエリ、モニタおよび状態に関するレポート、およびどのイテレーションおよび組織単位を使用するかといった、チーム プロジェクト管理におけるセキュリティを定義することができます。

目次

- ステップの概要
- ステップ 1 – Process Editor のインストール
- ステップ 2 – プロセス テンプレートの選択
- ステップ 3 – プロセス テンプレートのダウンロード
- ステップ 4 – Process Editor でプロセス テンプレートを開く
- ステップ 5 – プロセス テンプレート プラグインの変更
- ステップ 6 – 作業項目の種類の変更
- ステップ 7 – 既定の作業項目の変更
- ステップ 8 – クエリの変更および管理
- ステップ 9 – 区分およびイテレーションの変更
- ステップ 10 – グループおよびアクセス許可の変更
- ステップ 11 – ソース管理の設定の変更
- ステップ 12 – プロジェクト ポータルの変更
- ステップ 13 – レポートの変更
- ステップ 14 – 変更後のプロセス テンプレートのアップロード
- 参考資料

目的

- Process Editor ツールを使用してプロセス テンプレートをカスタマイズする方法の習得
- プロセス テンプレートの様々な編集可能パーツに関する学習

概要

Visual Studio Team System および Team Foundation Server は、ソフトウェア開発プロジェクトに関わるプロセス アクティビティの大半をサポートする統合環境として機能します。TFS は、プロセス テンプレートを使用してそのライフ サイクル方法論を実装します。プロセス テンプレートとは、方法論の様々な成果物およびプロセスの仕様を規定した XML ファイル セットのことです。プロセス テンプレートを変更することによって、新しいチーム プロジェクトの作成時にセットアップされる既定の作業項目の種類、セキュリティ設定、ソース管理の設定、およびレポートを変更することができます。

プロセス テンプレートを変更するには、まず、変更したいテンプレートをダウンロードしてから、Process Editor を使用して、必要な変更を加える必要があります。

ステップの概要

- ステップ 1 – Process Editor のインストール
- ステップ 2 – プロセス テンプレートの選択
- ステップ 3 – プロセス テンプレートのダウンロード
- ステップ 4 – Process Editor でプロセス テンプレートを開く
- ステップ 5 – プロセス テンプレート プラグインの変更
- ステップ 6 – 作業項目の種類の変更
- ステップ 7 – 既定の作業項目の変更
- ステップ 8 – クエリの変更および管理
- ステップ 9 – 区分およびイテレーションの変更
- ステップ 10 – グループおよびアクセス許可の変更
- ステップ 11 – ソース管理の設定の変更
- ステップ 12 – プロジェクト ポータルの変更
- ステップ 13 – レポートの変更
- ステップ 14 – 変更後のプロセス テンプレートのアップロード

ステップ 1 – Process Editor のインストール

このステップでは、**Process Editor** をインストールします。これは、プロセス テンプレートを表示およびカスタマイズするのに便利な UI ベースの手段を提供するツールです。**Process Editor** は、**Power Tool** に含まれています。

1. Power Tool インストーラを実行する前に、**DSL Tools for Visual Studio 2005** をインストールします。これは、Team System Process Editor の前提条件です。これがインストールされていない場合には、Power Tool インストーラによって Process Editor は除かれてインストールされます。DSL ツールは、http://go.microsoft.com/fwlink/?LinkId=82410 からダウンロードできます。

2. **Power Tool** を以下の URL からダウンロードします。

http://www.microsoft.com/downloads/details.aspx?familyid=7324c3db-658d-441b-8522-68

9c557d0a79&displaylang=en

3. 既定通りのインストールを実行した後、**Power Tool** が **Process Editor** と共に正しくインストールされたことを、次の手順で確認します。

a. [**スタート**] をクリックしてから [**すべてのプログラム**] をクリックします。

b. [**Microsoft Team Foundation Server Power Tool**] が選択肢にあるはずですので、それをクリックします。

c. これにより、[**Microsoft Visual Studio Team System Process Editor**] が表示されるはずです。

d. 表示されている場合には、Process Editor が正しくインストールされています。表示されていない場合、Power Tool をアンインストールし、上記のステップどおりの正しい順序で再インストールしてください。

ステップ 2 – プロセス テンプレートの選択

このステップでは、採択しているプロセスに最も近い、既成のテンプレートを選択します。それによって、カスタマイズ プロセスにあわせるために加える必要のある変更点が非常に少なくて済みます。既成のプロセス テンプレートの選択に役立つ詳細情報を以下に示します。

MSF for Agile Software Development

MSF for Agile Software Development は、短納期の小規模なプロジェクトに対して最適の設計になっています。MSF for Agile Software Development テンプレートは、次のような業務に適していると考えられます。

- 文書化されたプロセスがそれほど大量にはなく、正式なプロセスの開発が重要ではない。
- 小規模なソフトウェア開発チームが存在する。
- 様々なやり方で作業する多数の小規模開発チームを擁する。
- ソフトウェア開発サイクルが短い (週または月単位で測定した場合)。
- 1 年間につき複数のソフトウェア リリースをサポートする。

詳細は、Visual Studio Team System: MSF for Agile Software Development (http://go.microsoft.com/fwlink/?LinkId=51460) を参照してください。

MSF for CMMI Process Improvement

MSF for CMMI Process Improvement Process モデルは、主にソフトウェア開発の文化の外の実業界の専門家たちによって作成されました。彼らは実践によって学んだことを通してベスト プラクティスを正式化し、また改善することが、長期的に見て成功につながるという信念を持っています。MSF for

CMMI Process Improvement テンプレートは、次のような業務に適していると考えられます。

- 現在のビジネス活動を評価したいと考えている。
- 大規模なソフトウェア開発チームが存在する。
- 従来は別々であったグループまたは組織を統合する必要がある。
- 品質プロセス用のガイダンスを用意することができる。
- ソフトウェア開発サイクルが比較的長い。

詳細は、Visual Studio Team System: MSF for CMMI Process Improvement (http://go.microsoft.com/fwlink/?LinkId=51462) を参照してください。

ステップ 3 – プロセス テンプレートのダウンロード

このステップでは、選択したプロセス テンプレートをダウンロードして、カスタマイズ プロセスに適合するように修正します。

1. Visual Studio で、[**チーム**] をクリックし、[**Team Foundation Server の設定**] を選択します。
2. [**プロセス テンプレート マネージャ...**] をクリックします。
3. [**プロセス テンプレート マネージャ**] ダイアログで、修正したいプロセス テンプレートを選択し、[**ダウンロード**] をクリックします。
4. [**プロセス テンプレートのダウンロード**] ダイアログで、ローカル ドライブ上のフォルダ場所を選択し、[**保存**] をクリックします。

ステップ 4 – Process Editor でプロセス テンプレートを開く

このステップでは、ダウンロードしたプロセス テンプレートを Process Editor にロードして各種の設定を修正します。

1. Visual Studio インスタンスで、[**チーム**] メニューを選択します。
2. [**Process Editor**] をクリックします。その後 [**プロセス テンプレートを開く**] をクリックします。
3. [**プロセス テンプレート ファイル セットを開く**] ダイアログで、ダウンロードしたプロセス テンプレートを選択して、[**開く**] をクリックします。
4. これで、**ProcessTemplate.xml** ファイルが Visual Studio インスタンス内に開かれます。
5. カスタマイズする方法論の [**名前**] を設定します。

ステップ 5 – プロセス テンプレート プラグイン選択の変更

このステップでは、新しいチーム プロジェクトの作成時に実行するプロセス テンプレート プラグイン選択を修正します。プラグインは、必須ファイルのセットアップや、その区分のデータの構成を行います。Microsoft は、Microsoft Visual Studio 2005 Team System に対して次のようなプラグインを用意しています。

- **Classification** – プロジェクトの構造定義
- **Reporting** – プロジェクト レポートのアップロード
- **Portal** – プロジェクト サイトの作成
- **Work item tracking** – 作業項目定義のアップロード
- **Version control** - バージョン コントロールの作成
- **Groups** – グループの作成とアクセス許可の割り当て

[**方法の詳細**] ページで、プロジェクトの作成時に表示したいウィザードを選択し、ユーザーが必要に応じて修正できるようにします。

ステップ 6 – 作業項目の種類の変更

このステップでは、プロセス独自の新規の作業項目の種類を追加するか、または既定の作業項目の種類を修正することができます。

1. プロセス テンプレート エクスプローラで、[**作業項目の追跡**] をクリックします。
2. 次に、右側のペインで [**種類の定義**] タブを選択します。
3. 新しい作業項目を作成するには、右側のペインのツールバーにある [**追加**] をクリックします。
4. [**新しい作業項目の種類**] ダイアログで、作業項目の種類の名前を入力し、既存の作業項目の種類を [**コピー元**] ドロップダウン リストで選択します。
5. 新しい作業項目の種類が作成されて、右側のペインの [**種類の定義**] タブ内 [**項目一覧**] 中で有効になります。
6. 変更内容を保管するには、[**ファイル**] メニューの下の [**保存**] オプションをクリックします。
7. 新しい作業項目の種類または既存の作業項目の種類を対象に、属性あるいはフィールドの追加または除去を行いたい場合は、次のようにします。
8. [**種類の定義**] タブで、編集したい作業項目の種類を右クリックします。

9. [**開く**] を選択します。
10. 選択した作業項目の種類が、新しい Visual Studio ウィンドウに開かれます。
11. ここで、任意の属性を追加または除去することができます。

ステップ 7 – 既定の作業項目の変更

このステップでは、チーム プロジェクトの作成時に作成した既定の作業項目を追加または除去することができます。

1. プロセス テンプレート エクスプローラで、[**作業項目の追跡**] をクリックします。
2. 次に、右側のペインで [**既定の作業項目**] タブを選択します。
3. 新しい既定の作業項目の種類を作成するには、右側のペインのツールバーにある [**追加**] をクリックします。
4. [**種類の選択**] ダイアログ ボックスで、作業項目の種類を選択します。
5. 開いたダイアログ ボックスで、該当するフィールドを入力します。
6. [**OK**] をクリックします。

ステップ 8 – クエリの変更および管理

このステップでは、チーム プロジェクトの作成時に作成した既定のクエリを修正、追加、または除去することができます。

1. プロセス テンプレート エクスプローラで、[**作業項目の追跡**] をクリックします。
2. 次に、右側のペインで [**クエリ**] タブを選択します。
3. 新しいクエリを作成するには、右側のペインのツールバーにある [**追加**] をクリックします。
4. [**クエリ参照**] ダイアログ ボックスで、新しいクエリの名前を入力します。
5. 次に、[**クエリ定義の編集**] ボタンをクリックします。
6. [**クエリ編集**] ダイアログ内の [フィールド]、[並べ替え]、[条件] タブを使用して、クエリを定義します。
7. [**OK**] をクリックします。

重要事項 - Process Editor には、クエリを編集するための包括的な機能は揃っていません。クエリの編集およびテストは、プロセス テンプレート内で行うよりも、進行中のチーム プロジェクトで、Visual Studio で行うほうが望ましいと考えられます。その後それをファイルに保管しておけば、他のプロジ

ェクトにインポートしたり、プロセス テンプレートにコピーしたりすることができます。

ステップ 9 – 区分およびイテレーションの変更

このステップでは、チーム プロジェクトの作成時に使用できる既定のイテレーションおよび区分をセットアップすることができます。

1. プロセス テンプレート エクスプローラで、[**区分およびイテレーション**] をクリックします。
2. 次に、右側のペインで [**区分**] または [**イテレーション**] タブを選択します。
3. 区分またはイテレーションを追加、除去、または移動するには、右側のペインのツールバーを使用します。

ステップ 10 – グループおよびアクセス許可の変更

このステップでは、チーム プロジェクトの作成時に使用できるグループとそのアクセス許可の修正、除去、または追加を行うことができます。

1. プロセス テンプレート エクスプローラで、[**グループおよびアクセス許可**] をクリックします。
2. 新しいグループを作成するには、右側のペインのツールバーにある [**追加**] をクリックします。
3. [**グループ**] ダイアログ ボックスで、新しいグループの名前と、そのグループのメンバが実行できる内容の簡単な説明を入力してから、[**OK**] をクリックします。
4. 右側のペインの最上部のセクションのグループを選択し、下のセクション内でそのグループにアクセス許可を割り当てます。[**許可**] または [**拒否**] を選択できますが、[**未設定**] のまま残してもかまいません。[**未設定**] のアクセス権とは、暗黙の拒否のことです。

ステップ 11 – ソース管理の設定の変更

このステップでは、並列編集およびチェックイン編集用のソース管理設定を修正することができます。

1. プロセス テンプレート エクスプローラで、[**ソース管理**] をクリックします。
2. 次に、右側のペインで [**チェックアウトの設定**] タブを選択します。
3. 複数の人が 1 つのファイルを同時に編集できるようにしたい場合、[**複数のチェックアウトを有効にする**] チェック ボックスを選択します。
4. 右側のペインで [**チェックイン メモ**] タブを選択します。

5. 新しい [チェックイン メモ] フィールドを作成するには、右側のペインのツールバーにある [**追加**] をクリックします。
6. 既存の [チェックイン メモ] フィールドを編集するには、そのフィールドを選択して [**開く**] をクリックします。
7. [**チェックイン メモ**] ダイアログ ボックスで、必要な変更を行います。
8. 次に、[**OK**] ボタンをクリックします。
9. 現行バージョンの Process Editor では、[**アクセス許可**] タブに変更を加えることは一切できません。

ステップ 12 – プロジェクト ポータルの変更

このステップでは、プロジェクト ポータルを修正して、特定のレポート、文書、およびプロセス ガイダンスを表示します。

1. プロセス テンプレート エクスプローラで、[**ポータル**] をクリックします。
2. 新しいライブラリを作成するには、中央ペインの [**ポータル**] ノードを右クリックし、[**新しいドキュメント ライブラリ**] をクリックします。
3. [**ドキュメント ライブラリ**] ダイアログ ボックスに、ライブラリの名前と説明を入力します。
4. 次に、[**OK**] をクリックします。
5. ドキュメント ライブラリを右クリックし、[**フォルダの新規作成**] をクリックして新しいフォルダを作成します。
6. [**フォルダのプロパティ**] ダイアログ ボックスで、新しいフォルダの名前を入力します。
7. フォルダを選択し、右側のペインのツールバーにある [**追加**] をクリックして、新しい文書をアップロードします。
8. [**ファイルのインポート**] ダイアログ ボックスで、アップロードしたいファイルを参照します。
9. [**移動先フォルダ**] および [**共有ポイント フォルダ**] には自動的にデータが取り込まれます。[**クエリ ID**] フィールドは未変更のままにします。
10. 次に、[**インポート**] をクリックします。
11. ドキュメント ライブラリ内の既存の文書のプロパティを編集するには、右側のペインでその文書を選択し、ツールバーの [**開く**] をクリックします。
12. [**ファイルの編集**] ダイアログ ボックスで、[**名前**] フィールド内のファイル名を変更します。他の

フィールドは変更しないでください。

13. 次に、[**OK**] をクリックします。

ステップ 13 – レポートの変更

このステップでは、チーム プロジェクトの作成時に作成される、既定のレポートを追加、除去、および修正します。

1. プロセス テンプレート エクスプローラで、[**レポート**] をクリックします。
2. ツールバーの [**追加**] をクリックします。
3. [**レポート**] ダイアログで、[**レポートの詳細**] タブにレポートの名前を入力します。
4. 次に [**ファイル名**] フィールドで、追加したい .rdl ファイルを参照します。他のフィールドは未変更のままにします。
5. [**プロパティ**] および [**パラメータ**] タブ内のデータは一切変更しません。
6. [**DataSources**] タブで、該当するデータ ソースを入力します。Team Foundation Server に付属しているプロセス テンプレート用の既定のデータ ソースは、/TfsOlapReportDS と /TfsReportDS です。
7. 次に、[**OK**] をクリックします。

ステップ 14 – 変更後のプロセス テンプレートのアップロード

このステップでは、新しいチーム プロジェクトの作成時に使用できるように、修正後のプロセス テンプレートを TFS サーバーにアップロードします。

1. Visual Studio で、[**チーム**] をクリックし、[**Team Foundation Server の設定**] を選択します。
2. [**プロセス テンプレート マネージャ...**] をクリックします。
3. [**プロセス テンプレート マネージャ**] ダイアログで、アップロードボタンをクリックします。
4. プロセス テンプレートをダウンロードして修正した、ローカル フォルダを参照します。
5. [**プロセス テンプレートのアップロード**] ダイアログ上の [**アップロード**] ボタンをクリックします。
6. 新しいプロセス テンプレートが、[**プロセス テンプレート:**] リスト中に表示されているはずです。
7. [閉じる] ボタンをクリックします。

これで、新しいチーム プロジェクトを作成するときは、プロセス テンプレート選択時のオプションの

1 つとして、新たに作成したプロセスを利用できます。

参考資料

- プロセス テンプレートのカスタマイズに関する詳細は、「Process Template Customization Overview」(http://msdn2.microsoft.com/enus/library/ms194945(VS.80).aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server でレポートをカスタマイズする方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、既存のレポートをカスタマイズしてから、Team Foundation Server 内のチーム レポート ポータルに対してそれを発行する方法を説明します。

目次

- ステップ 6 – Team Foundation Server に対するレポートの配置
- ステップ 7 – レポートのテスト
- 参考資料

目的

- Visual Studio でのレポート プロジェクトの作成
- レポート プロジェクトでの既存のレポートのカスタマイズ
- レポート サーバーに対する新規レポートの発行

概要

VSTS に付属しているレポートは、SQL Server Reporting Services レポートです。このレポートに手を加えることもできますが、SQL Server 2005 クライアント ツールに付属している SQL Server 2005 Reporting Services Designer を使用して、Visual Studio (Business Intelligence Development Studio) 内で独自のカスタム レポートを作成することもできます。
新しいレポートを最初から作成しなくても、レポートをカスタマイズすれば、既存のレポートに機能を追加することができます。必要なレポートが、既に存在するものに似通っている場合、その既存のレポートをカスタマイズすれば、時間の節約になります。既存のレポートをカスタマイズするには、そのレポートをレポート サーバーからエクスポートし、Visual Studio 内の既存のレポート プロジェクトに追加してから、変更を加えた後でレポート ポータルに対して再配置する必要があります。

ステップの概要

- ステップ 1 – 新しいレポート プロジェクトの作成
- ステップ 2 – カスタマイズするレポートのエクスポート
- ステップ 3 – データ ソースの作成
- ステップ 4 – プロジェクトへのレポートの追加
- ステップ 5 – レポートの変更
- ステップ 6 – Team Foundation Server に対するレポートの配置
- ステップ 7 – レポートのテスト

始める前に

Team Foundation Server 用のレポートをカスタマイズするときは、以下の前提条件が揃っていることを事前に確認する必要があります。

- レポートをカスタマイズするときに使用する Business Intelligence Development Studio をマシンにインストールしておく必要があります。それがインストールされているかどうかを確認するには、Visual Studio を調べて、新しいプロジェクトの作成時に、 [ビジネス インテリジェンス プロジェクト] がプロジェクトの種類にあるかどうかを確かめます。
- 使用するユーザー アカウントは、データ層サーバー上の Microsoft 分析サーバーの TfsWarehouseDataReaders のセキュリティ ロールのメンバでなければなりません。
- 使用するユーザー アカウントは、データ層上の TFSWarehouse データベースに対する管理者権限を持っていなければなりません。
- 使用するユーザー アカウントは、アプリケーション層サーバー上の SQL Server Reporting Services Publisher ロールのメンバでなければなりません。

ステップ 1 - 新しいレポート プロジェクトの作成

新規のレポート プロジェクトを作成し、既存のレポートをプロジェクトに追加してカスタマイズできるようにします。新規のレポート プロジェクトを Visual Studio で作成するには、次のようなステップを行います。

1. [**ファイル**] をクリックしてから [**新規作成**] をクリックし、次に [**プロジェクト**] をクリックします。
2. プロジェクトの種類で [**ビジネス インテリジェンス プロジェクト**] を選択します。
3. [**レポート サーバー プロジェクト**] テンプレートを選択します。
4. [**プロジェクト名**] および [**場所**] を設定してから、[OK] をクリックします。

ステップ 2 - カスタマイズするレポートのエクスポート

カスタマイズしたいレポートをプロジェクト ポータルからエクスポートして、新規のレポート プロジェクトにインポートできるようにします。レポートをエクスポートするには、次のようなステップを行

います。

1. チーム プロジェクトを右クリックして、[**プロジェクト ポータルの表示**] を選択します。
2. ポータル Web サイトの左側のクイック起動バー内の [**レポート**] をクリックします。
3. カスタマイズしようとしているレポートをクリックします。
4. [**プロパティ**] をクリックします。
5. [**編集**] を選択します。
6. レポート rdl ファイルを、ステップ 1 で作成したレポート プロジェクト フォルダに保管します。

ステップ 3 - データ ソースの作成

カスタマイズしたレポートを編集して発行するには、Team Foundation Server のデータ ウェアハウスおよび OLAP キューブ用のデータ ソースを追加する必要があります。このデータ ソースを Visual Studio プロジェクトに追加し終わったら、サーバーからレポートにデータをプルすることができます。

ウェアハウス データ ソースを作成するには:

1. Visual Studio の [**ソリューション エクスプローラ**] で、[**共有データ ソース**] を右クリックしてから、[**新しいデータ ソースの追加**] をクリックします。
2. [**一般**] タブの [**名前**] テキスト ボックスに **TfsReportDS** と入力します。
3. [**種類**] コンボ ボックスで、[**Microsoft SQL Server**] を選択します。
4. [**編集...**] ボタンをクリックします。
5. データ層サーバー名を入力します。
6. データベース **TFSWarehouse** を選択します。
7. [**OK**] ボタンを 2 回クリックして、データ ソースを追加します。

OLAP データ ソースを作成するには:

1. [**ソリューション エクスプローラ**] で、[**共有データ ソース**] を右クリックしてから、[**新しいデータ ソースの追加**] をクリックします。
2. [**一般**] タブの [**名前**] テキスト ボックスに **TfsOlapReportDS** と入力します。
3. [**種類**] コンボ ボックスで、[**Microsoft SQL Server Analysis Services**] を選択します。
4. [**編集...**] ボタンをクリックします。
5. データ層サーバー名を入力します。
6. データベース **TFSWarehouse** を選択します。

7. [**OK**] ボタンを 2 回クリックして、データ ソースを追加します。

ステップ 4 - プロジェクトへのレポートの追加

プロジェクトへのデータ ソースの追加が完了したら、ステップ 2 でエクスポートしたレポートを追加することができます。

1. [**ソリューション エクスプローラ**] で、[**レポート**] を右クリックしてから、[**追加**] -> [**既存の項目...**] を選択します。
2. ステップ 2 でエクスポートした rdl ファイルを参照します。

ステップ 5 - レポートの変更

プロジェクトにレポートを追加し終わったら、修正を加えて、ご自分のニーズに合わせてカスタマイズすることができます。修正用のレポートを開くには、[**ソリューション エクスプローラ**] 内のそのレポートをダブルクリックします。レポートを開いたら、次のような修正を加えることができます。

- [**データ ペイン**] 内のクエリ ステートメントを変更する。
- 新しいメジャーまたはメンバを [**データ ペイン**] 内にドラッグする。
- [**レイアウト ペイン**] 内でレポート レイアウトを変更する。

ステップ 6 - Team Foundation Server に対するレポートの配置

レポートに修正を加えた後、次のようなステップを行って、チーム プロジェクトのレポート ポータルに対してそれを配置することができます。

1. [**ソリューション エクスプローラ**] 内のレポート プロジェクトを右クリックしてから、[**プロパティ**] をクリックします。
2. [**OverwriteDataSources**] が **false** に設定されていることを確認します。
3. [**TargetDataSourceFolder**] をご自分のチーム プロジェクトの名前に変更します。例: **TargetDataSourceFolder** = TestProject。
4. [**TargetReportFolder**] をご自分のチーム プロジェクトの名前に変更します。例: **TargetDataSourceFolder** = TestProject。
5. **TargetDataSourceFolder** を http://<データ層のサーバー名>/reportserver に変更します。

例: **TargetDataSourceFolder** = http://tfsrtm/reportserver。

6. [**OK**] をクリックします。

7. [**ソリューション エクスプローラ**] 内の rdl ファイルを右クリックしてから、[**配置**] をクリックします。

8. [**出力ペイン**] を調べて、正常に完了したことを確認します。

ステップ 7 - レポートのテスト

チーム プロジェクトのレポート サーバーに対してレポートを発行し終わったら、次のようにしてそのテストを行って、正常に配置されたことを確認することができます。

1. チーム プロジェクトを右クリックして、[**プロジェクト ポータルの表示**] を選択します。
2. ポータル Web サイトの左側のクイック起動バー内の [**レポート**] をクリックします。
3. カスタマイズしたレポートを選択します。
4. 行ったカスタマイズが、想定していたとおりであることを確認します。

参考資料

- レポート プロジェクトの処理方法を説明したチュートリアルは、「Reporting Services のチュートリアル」(http://msdn2.microsoft.com/ja-jp/library/ms170246.aspx) を参照してください。
- レポートの編集に関する MSDN 記事は、「方法: レポート デザイナでレポートを編集する」(http://msdn2.microsoft.com/ja-jp/library/ms244655(VS.80).aspx) を参照してください。
- データ層におけるセキュリティ ロールに関する詳細は、「Analysis Services を使用したアクセスのセキュリティ保護」(http://msdn2.microsoft.com/ja-jp/library/ms174839.aspx) を参照してください。
- アプリケーション層におけるセキュリティ ロールに関する詳細は、「Reporting Services の保護」(http://msdn2.microsoft.com/ja-jp/library/ms157198.aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server でプロジェクトを管理する方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、選択したテンプレートに基づいて、新しいチーム プロジェクトを作成して構成することによって、新しいソフトウェア開発プロジェクトを開始する方法を示します。また、Team Foundation Server ツールを使用して、ソフトウェア開発のライフ サイクル全体を通して、進行中のソフトウェア開発プロセスを管理、制御、およびモニタする方法も説明します。

目次

目的

• TFS でプロジェクトを管理する方法の習得

概要

Visual Studio Team System には、プロジェクトの管理者がソフトウェア開発ライフ サイクル全体を、一律の一元化されたやり方で制御およびサポートするときに役立つツールおよびレポートが揃っています。Visual Studio Team System と、すべてのチーム メンバが共用する集中データベースを通してソフトウェア プロセスを直接制御することによって、チームのコミュニケーションが改善し、チーム メンバ間で必要な受け渡しが自動化されます。Team Foundation Server のデータ ウェアハウスによって運用されるレポートを使用すれば、プロジェクトの進行状況のモニタおよび追跡が一層簡単になります。

この [HOWTO] では、新しいチーム プロジェクトを作成および構成するプロセスと、進行中のプロジェクトの進行状況をモニタする方法を 1 つずつ順に解説していきます。

ステップの概要

- ステップ 1 – プロセス テンプレートの選択
- ステップ 2 – チーム プロジェクトの作成
- ステップ 3 – セキュリティ グループの作成 (オプション)
- ステップ 4 – Team Foundation Server でのチーム メンバの追加
- ステップ 5 – イテレーション サイクルの決定
- ステップ 6 – TFS でのプロジェクト シナリオの取り込み
- ステップ 7 – イテレーションのシナリオの特定
- ステップ 8 – サービス品質要求の定義
- ステップ 9 – イテレーションの計画
- ステップ 10 – 進行状況のモニタ

始める前に

新しいチーム プロジェクトの作成を開始する前に、次のような情報を収集する必要があります。

- **管理する対象のプロジェクトのスタイル。**これにより、プロジェクトの基盤となる初期テンプレー

トの選択が決まります。また、提供されたテンプレートをどの程度カスタマイズすればよいかを判定するときの参考にもなります。アジャイル スタイルが望ましい場合、まず MSF for Agile Software Development テンプレートを選択してから、必要なカスタマイズを検討することができます。

- **プロジェクト名。**作成するチーム プロジェクトで使用する命名規則を慎重に検討します。プロジェクト名が、他と重複しないように気を付けて、他の担当者が簡単にプロジェクトを識別できるような名前を付ける必要があります。そのような名前の一部としてプロジェクト コードを使用する組織もあれば、部署とプロジェクトのタイトルを組み合わせて使用する組織もあります。
- **ソース管理の構造。**空のソース管理ツリーから始める必要があるか、または既存のソース コードをプロジェクトのベースにする必要があるかどうかを検討します。使用するコードおよび分岐に関する要件に基づいて、該当するソース管理構造を評価します。詳細は、「[How To]: ソース管理フォルダを構造化する方法」を参照してください。

ステップ 1 – プロセス テンプレートの選択

まず、プロセス テンプレートを選択します。管理しようとしているプロジェクトのスタイルを評価し、提供された 2 つのテンプレートに用意されているさまざまな機能を調べます。評価する機能には、作業項目の定義 (要件に対して十分なフィールドが提供されているか)、文書テンプレート、ワークフロー、チェックイン ポリシー、およびレポートなどがあります。

Team Foundation Server には、次のようなプロセス テンプレートが用意されています。

• **MSF for Agile Software Development** – これは、小規模でしかも形式にとらわれないソフトウェア プロジェクトを対象とした軽量プロセスです。これは、シナリオおよびコンテキストに基づいた行動をベースにしており、プロジェクトおよびチーム メンバ (担当者) を中心としています。

• **MSF for CMMI® Process Improvement** – これは、より成熟したソフトウェア プロジェクト向けに設計されています。これは、監査、検証、および正式なプロセスのサポートを提供することによって、MSF for Agile Software Development を拡張しています。これは、プロセスおよびプロセスへの準拠を拠り所とし、組織が中心に置かれます。

必要があれば、提供されたプロセス テンプレートをカスタマイズして、プロセスにもっと忠実に即したものにすることができます。プロセス テンプレートのカスタマイズに関する詳細は、「[How To]: プロセス テンプレートをカスタマイズする方法」を参照してください。

この [HOWTO] では、以後は、**MSF for Agile Software Development** テンプレートを選択したことを前提とします。

ステップ 2 – チーム プロジェクトの作成

新しいチーム プロジェクトを作成するには:

1. Visual Studio が Team Foundation Server インスタンスに接続していることを確認します。
2. [**チーム エクスプローラ**] で、サーバー ノードを右クリックし、[**新しいチーム プロジェクト...**] をクリックします。
3. プロジェクト作成ウィザードの先頭ページで、チーム プロジェクトの名前を入力してから、[**次へ**] をクリックします。
4. [**プロセス テンプレートの選択**] ページで、ドロップダウンからステップ 1 で選んだプロセス テンプレートを選択します。この例の場合、[**MSF for Agile Software Development –v4.0**] を選択し、[**次へ**] をクリックします。
5. [**プロジェクト ポータルの設定を指定します**] ページで、チーム プロジェクト ポータルの名前と説明を入力し、[**次へ**] をクリックします。ここで入力した名前は、プロジェクト ポータルの Windows SharePoint Services Web サイトの作成時に使用されます。
6. [**ソース管理の設定を指定します**] で、[**空のソース管理フォルダを作成する**] を選択してから、[**次へ**] をクリックします。
7. [**チーム プロジェクト設定の確認**] で、設定を確認してから、[**完了**] をクリックします。

これで、MSF for Agile Software Development プロセス テンプレートに基づいたチーム プロジェクトが TFS サーバー上に作成されます。

ステップ 3 – セキュリティ グループの作成 (オプション)

Team Foundation Server でプロジェクトを作成すると、選択したプロセス テンプレートが何であっても、そのプロジェクト用に 4 つの既定のグループが作成されます。既定では、どのグループに対しても一連のアクセス権が定義されていて、何を実行する許可がグループ内のメンバに与えられるかが規制されます。その 4 つのグループは次のとおりです。

- プロジェクト管理者 (Project Administrator)
- 貢献者 (Contributor)

- 閲覧者 (Reader)
- ビルド サービス (Build Services)

チーム プロジェクトのセキュリティ グループを作成して、組織のセキュリティ要件がさらに忠実に守られるようにすることができます。セキュリティ グループの作成は、チーム プロジェクトのユーザーグループにそれぞれ個別に一連のアクセス権を認可する効率のよい手段になります。グループごとに最低限必要なアクセス権のみを許可し、この新しいチーム プロジェクト グループに所属する必要のあるユーザーまたはグループのみを追加するように注意します。

この手順を実行するには、**Project Administrator** グループのメンバでなければなりません。ただし、Windows 管理者である必要はありません。

1. [チーム エクスプローラ] で、グループを作成するチーム プロジェクトを選択します。
2. [**チーム**] メニューで、[**チーム プロジェクト設定**] をポイントしてから、[**グループ メンバシップ**] をクリックします。
3. [**プロジェクト グループ**] ダイアログ ボックスで、[**新規作成**] をクリックします。
4. [**新しい Team Foundation Server グループの作成**] ダイアログ ボックスの [**グループ名**] ボックスに、チーム プロジェクト グループの名前を入力します。
5. [**説明**] ボックスに、グループの説明を入力します。
6. [**OK**] をクリックしてから、[**閉じる**] をクリックします。

チーム プロジェクト グループの作成が完了したら、そのグループに該当するアクセス権を与えて、メンバをグループに追加します。既定では、チーム プロジェクト グループには、その作成時には何もアクセス権は与えられていません。

ステップ 4 – Team Foundation Server でのチーム メンバの追加

このステップでは、プロジェクトで作業するリソースとその役割を識別し、それらのチーム メンバを Team Foundation Server に割り当てます。既存のチーム プロジェクト グループまたはサーバー レベル グループにユーザーを追加することができます。また、作成したグループにメンバを追加する必要もあります。この手順を実行するには、**Team Foundation Administrators** グループのメンバでなければなりません。

1. [チーム エクスプローラ] で、チーム プロジェクトを選択します。
2. [**チーム**] メニューで、[**チーム プロジェクト設定**] をポイントし、[**グループ メンバシップ**] をク

リックします。ただし、サーバー レベル グループにユーザーを追加するには、[**Team Foundation Server の設定**] をポイントし、[**グループ メンバシップ**] をクリックします。

3. [**プロジェクト グループ**] ダイアログ ボックスで、ユーザーの追加先のグループを選択し、次に [**プロパティ**] をクリックします。

4. [**Team Foundation Server グループ プロパティ**] ダイアログ ボックスの [**メンバ**] タブの [**メンバの追加**] の下で、[**Windows ユーザーまたはグループ**] を選択します。

5. [**追加**] をクリックします。

6. [**ユーザーまたはグループの選択**] ダイアログ ボックスで、[**選択するオブジェクト名を入力してください**] の下に、追加するユーザーのドメイン名とエイリアスを **domain¥username** の形式で入力します。複数のユーザーを一度に追加するには、項目をそれぞれセミコロン (;) で区切ります。

7. [**OK**] を 2 回クリックしてから、[**閉じる**] をクリックします。

ステップ 5 – イテレーション サイクルの決定

イテレーションとは、固定長の時間の長さのことであり、その間に作業を計画し、スケジュールして実行します。

要件の定義から、分析、設計、開発、コーディング、およびテストにいたるまでのソフトウェア開発ライフ サイクルのすべてのコンポーネントがこのイテレーションを構成する各グループに分けられますが、これは通常は 2 ～ 6 週間程度のものです。

このステップでは、プロジェクトのイテレーション サイクルを決定します。イテレーション サイクルの決定において注意すべき重要点は次のとおりです。

- イテレーション サイクルは、かなりの作業量の実行と少なくともいくつかのシナリオに対処するのに十分な大きさでなければなりません。
- イテレーション サイクルは、変更や優先順位に柔軟に対応するのに十分な小ささでなければなりません。

これは、プロジェクトの規模および複雑さによって異なります。実際問題として、たいていのプロジェクトにとって、2 週間のイテレーション サイクルが最も効率が上がります。

ステップ 6 – TFS でのプロジェクト シナリオの取り込み

このステップでは、TFS でプロジェクト シナリオを取り込みます。それによって、プロジェクトの実

行やその進行状況の追跡の向上を目指した計画をたてることができます。

1. 顧客、ビジネス アナリスト、エンド ユーザー、およびプロダクト マネージャを含め、さまざまな利害関係者から得た情報を使用して、プロジェクト バック ログ (PBL) 文書を作成します。一般的に、PBL は、主に要件を取り込むために設計された Word 文書です。

2. PBL を情報源として使用して、プロジェクトのシナリオを特定します。

3. 開始時点にすべてのシナリオを作成することはできますが、イテレーションを繰り返しながらシナリオを作成してもかまいません。プロジェクトの全体像をより的確に把握し、進行状況の追跡に役立つように、すべてのシナリオを前もって取り込んでおくことをお勧めします。

4. プロジェクト用として「Iteration 999」というイテレーションを作成し、シナリオをこれに割り当てて、簡単に検索および追跡できるようにします。

MSF Agile Software Development テンプレートを使用するプロジェクトでシナリオを作成するには:

1. [**チーム エクスプローラ**] で、プロジェクト ノードを展開し、[**作業項目**] フォルダを右クリックします。

2. [**作業項目の追加**] をポイントしてから、[**シナリオ**] をクリックします。

3. [**新しいシナリオ**] ページで、シナリオの詳細を入力します。イテレーションが「Iteration 999」に必ず設定されるようにします。

4. 新しいシナリオを保存します。

5. プロジェクト用として特定したすべてのシナリオに対して、上記のステップを繰り返します。

ステップ 7 – イテレーションのシナリオの特定

このステップでは、個々のイテレーションで取り組むシナリオを特定します。このステップは、イテレーション サイクルごとに繰り返します。

1. プロジェクトの関係者からの情報および機能の優先順位に基づいて、個々のイテレーション中に作業の対象となる作業項目を識別します。

2. そのような作業項目をイテレーションに割り当てます。

作業項目を検索し、それを個々のイテレーションを割り当てるには:

1. プロジェクト内の [**作業項目**] フォルダを右クリックし、[**クエリの追加**] をクリックします。

2. [**新しいクエリ**] で、新たに AND 句を追加します。[**イテレーション パス**] を [**フィールド**] ドロ

ップダウンで選択し、その値を **Iteration 999** に設定します。

3. 新たに AND 句を追加して、[**作業項目の種類**] を [**シナリオ**] に指定します。

4. F5 を押してクエリを実行し、保留中のすべてのシナリオを表示します。

5. イテレーション内で処理したいシナリオをダブルクリックします。

6. [**イテレーション**] を現在のイテレーション パスに変更し、[上書き保存] ボタンをクリックします。

7. 特定したすべてのシナリオに対して、上記のステップを繰り返します。

重要事項:– 将来の利用に備えて、クエリを保存することができます。

ステップ 8 – サービス品質要求の定義

このステップでは、イテレーション サイクル中に処理するシナリオごとにサービス品質要求 (QOS) を定義します。このステップは、イテレーション サイクルごとに繰り返します。これは、シナリオの受け入れ基準を定義するときに役立ちます。QOS の元となるデータは、プロジェクトの目標、要件、および仕様書 (ある場合) から得ることができます。

1. プロジェクト内の [**作業項目**] フォルダを右クリックし、[**作業項目の追加**] をポイントしてから、[**サービス品質要求**] をクリックします。

2. [**新しいサービス品質要求**] ページで、次のような詳細を追加します。

a. [**種類**] を、パフォーマンス、スケーラビリティ、またはセキュリティなどの該当する値に設定します。

b. [イテレーション] を現在のイテレーション サイクルに設定します。

c. [**リンク**] タブから QOS を特定のシナリオにリンクし、追跡しやすいようにします。

3. 新しい QOS を上書き保存します。

4. どのシナリオにも複数の QOS がありうることに留意しながら、各規範または品質の種類の要件ごとに QOS を 1 つずつ作成します。

5. イテレーション サイクルごとに取り扱うどのシナリオに対しても、QOS を必ず作成してください。

重要事項 – 後で、QOS を複数のテスト タスクに分割することができます。

ステップ 9 – イテレーションの計画

このステップでは、シナリオをタスクに分割することによって、イテレーションの計画をたてます。つまり、それぞれのタスクを見積もり、関連チーム メンバにタスクを割り当てます。このステップは、

イテレーション サイクルごとに繰り返します。

開発者のタスク

1. 選択したシナリオを開発者のシナリオに分割します。
2. 開発者のシナリオを開発者のタスクにさらに分割します。
3. TFS での開発者のタスクをタスク作業項目として取り込みます。
    a. [チーム エクスプローラ] のプロジェクト ノードの下で、[**作業項目**] フォルダを右クリックします。
    b. [**作業項目の追加**] をポイントしてから、[**タスク**] をクリックします。
    c. [**新しいタスク**] ページで、次のような詳細を追加します。
        i. [**分野**] を [**開発**] に設定します。
        ii. [**サイクル**] を現在のイテレーション サイクルに設定します。
        iii. [**リンク**] タブで、タスクを特定のシナリオにリンクし、追跡しやすいようにします。
        iv. ここで、説明とともに、タスクの受け入れ基準を取り込むことができます。これは、タスクが無事に完了したかどうかを判別するためのものです。
        v. [**担当者**] フィールドを、タスクに従事する開発者に設定します。
    d. 新しいタスクを上書き保存します。
    e. 特定したすべてのタスクに対して、上記のステップを繰り返します。
4. 特定したイテレーションのすべてのシナリオに対して、上記のステップを繰り返します。

テスト タスク

1. 特定したシナリオの QOS を複数のテスト ケースに分割します。
2. テスト ケースを複数のテスト タスクに分割します。
3. このテスト タスクは、TFS でタスク作業項目として取り込まれます。
    a. [チーム エクスプローラ] のプロジェクト ノードの下で、[**作業項目**] フォルダを右クリックします。
    b. [**作業項目の追加**] をポイントしてから、[**タスク**] をクリックします。
    c. [**新しいタスク**] ページで、次のような詳細を追加します。
        i. [**分野**] を [**テスト**] に設定します。

ii. [**サイクル**] を現在のイテレーション サイクルに設定します。

iii. [**リンク**] タブで、タスクを特定の QOS にリンクし、追跡しやすいようにします。

iv. ここで、説明とともに、タスクの受け入れ基準を取り込むことができます。これは、タスクが無事に完了したかどうかを判別するためのものです。

v. [**担当者**] フィールドを、タスクに従事するテスト担当者に設定します。

d. 新しいタスクを上書き保存します。

e. 特定したすべてのタスクに対して、上記のステップを繰り返します。

4. イテレーションのすべての QOS に対して、上記のステップを繰り返します。

その他

1. アーキテクチャ、リリース管理、プロジェクトの管理および要件などの、追跡の必要のある他の規範に関しても、イテレーションの関連タスクを取り込みます。
2. これらのタスクをそれぞれ該当するシナリオにリンクし、簡単に追跡できるようにします。

大量の作業項目からなる大型プロジェクトの場合、Excel 統合機能を使用して、Excel スプレッドシートで作業項目を作成してから、TFS にアップロードすることができます。詳細は、「Microsoft Excel で作業項目リストを操作する」(http://msdn2.microsoft.com/ja-jp/library/ms181694(VS.80).aspx) を参照してください。

多数のリソースが関与する大型プロジェクトの場合、Microsoft Project 統合機能を使用して、作業項目を追跡し、タスクの割り当ての平衡をとることができます。詳細は、「Microsoft Project で作業項目を操作する」(http://msdn2.microsoft.com/ja-jp/library/ms244368(VS.80).aspx) を参照してください。

ステップ 10 – 進行状況のモニタ

このステップでは、主要業績評価指標 (KPI) を識別し、レポート機能にリンクして、プロジェクトを簡単に追跡できるようにします。

区分およびイテレーションに関する考慮事項

- 開始区分としては、ルート レベルを使用することができます。

- その後、たとえば UI、Biz、Data などの、プロジェクト内の各種の区分が論理的に差異化された時点で、シナリオ/作業項目をそれぞれ個別区分に割り当てることができます。

関連資料

- 詳細は、「Microsoft Excel で作業項目リストを操作する」(http://msdn2.microsoft.com/ja-jp/library/ms181694(VS.80).aspx) を参照してください。
- 詳細は、「Microsoft Project で作業項目を操作する」(http://msdn2.microsoft.com/ja-jp/library/ms244368(VS.80).aspx) を参照してください。

[HOWTO]: Visual SourceSafe から Visual Studio Team Foundation Server にソース コードを移行する方法

適用対象

- Visual Studio Team Foundation Server

概要

バージョン管理されたソース コードを VSS から TFS に移行するには、VSS コンバータ ツールを使用します。このツールを使うと、ファイル、フォルダ、バージョン ヒストリ、ラベル、およびユーザー情報を VSS データベースから TFS ソース管理データベースに移行することができます。ソースを移行する場合、事前に VSS ソース データベースをバックアップして準備する必要があります。

目次

- ステップ 2 - データ保全性の問題を解決するための VSS データベースの分析
- ステップ 3 - VSS でのプロジェクトの分析
- ステップ 4 - プロジェクトの移行の準備
- ステップ 5 - プロジェクトの移行
- その他の考慮事項
- 参考資料

目的

• TFS への Visual Source Safe プロジェクトの移行

概要

Team Foundation Server には、Visual SourceSafe (VSS) データベース内で保守されている既存のソースを、Team Foundation Server ソース管理に移行するときに役立つツールが用意されています。TFS には、ファイル、フォルダ、バージョン ヒストリ、ラベル、およびユーザー情報を移行するための VSS コンバータが用意されています。

始める前に

この [HOWTO] に書かれているステップを正しく実行するには:

• Team Foundation Administrators グループのメンバであるアカウントを使用して、ログインする必要があります。

• コンバータを実行するコンピュータに VSS 2005 クライアントをインストールしておく必要があります。それより前のバージョンの VSS を実行すると、警告と共にコンバータが停止します。
また、VSS 2005 クライアントの Analyze コマンドを必ず使用する必要もあります。それは、前のバージョンでは検出不可能であった問題を、2005 Analyze コマンドは検出できるからです。VSS データベースは、ネイティブの 2005 データベースである必要はありません。

• コンバータを実行するコンピュータ上で、SQL Express がインストール済みであって、有効になっている必要があります。コンバータ ツールは、変換時に一時データベースとしてローカルの SQL インスタンスを使用します。SQL Express は、既定として Visual Studio 2005 と一緒にインストール

されます。

• Active Directory に定義されている TFS ドメイン名と TFS ユーザー名リストを必ず収集しておきます。

• VSS 管理者のユーザー名およびパスワードと、TFS プロジェクト管理者のユーザー名およびパスワードを保有していることを確認します。

ステップの概要

- ステップ 1 - VSS データベースのバックアップ
- ステップ 2 - データ保全性の問題を解決するための VSS データベースの分析
- ステップ 3 - VSS でのプロジェクトの分析
- ステップ 4 - プロジェクトの移行の準備
- ステップ 5 - プロジェクトの移行

ステップ 1 - VSS データベースのバックアップ

移行を実行する前に、移行したい Visual SourceSafe データベースのバックアップ コピーを先に作成します。

VSS データベースをバックアップするには:

1. すべてのユーザーに対して、ファイルをチェックインしてから VSS データベースをログオフするよう指示します。その際、Visual Studio IDE と VSS エクスプローラを両方とも閉じるよう指示します。

重要事項: チェックアウトしたファイルは、TFS には移行されません。

2. データベースに誰も現在ログインしていないことを確認します。
3. データベースに対する分析ジョブの実行がスケジュールされていないことを確認します。
4. 以下のフォルダをバックアップ先にコピーします。

¥DATA

¥Temp

¥USERS

5. 以下のファイルをバックアップ先にコピーします。

User.txt

Srcsafe.ini

ステップ 2 - データ保全性の問題を解決するための VSS データベースの分析

このステップでは、Visual SourceSafe 分析ユーティリティを使用して、データベース内のデータ保全性に関する問題を見つけ出して解決します。

VSS ANALYZE ユーティリティを使用するには:

1. Analyze.exe を実行して、データベース破壊またはデータベース エラーを検索します。次のようなコマンド ラインを使用します。

analyze "<SourceSafe のデータ ディレクトリ>"

自動実行の場合は、**-I** オプションを使用します。このコマンドは、潜在的な問題をレポートします。

2. アクセス権の問題がある場合や、「unable to checkout files (ファイルをチェックアウトできません)」エラーまたは「losing checkout status (チェックアウト状況を喪失しました)」エラーや、Status.dat ファイルまたは Rights.dat ファイルについて述べたその他のエラーが表示された場合、Ddconv.exe プログラムまたは Ddconvw.exe プログラムを実行します。これらのプログラムは、SourceSafe データベースを古い形式から現在の形式に更新します。既定では、これらのプログラムは、¥Admin サブディレクトリにインストールされます。

ステップ 3 - VSS でのプロジェクトの分析

このステップでは、どのプロジェクトを移行するかを決定してから、TFS コマンドライン ツール VSSConverter.exe を実行して、移行問題の原因になりうる潜在的な問題に関して VSS データベースを分析します。

VSSConverter.exe を使用してプロジェクトを分析するには:

1. 次のような XML 設定ファイル (仮に ConversionSettings.xml と呼びます) を作成します。

上記のサンプルでは、**MyFirstProject** および **MySecondProject** は、移行しようとしている VSS 内のプロジェクト フォルダの名前を表します。VSS データベース全体を移行するには、**<Project Source="$/"></Project>** を使用します。

2. 次のようにして Visual Studio コマンド プロンプトから **Analyze** スイッチを引き渡して VSSConverter.exe を実行し、プロジェクトを分析します。

VSSConverter Analyze ConversionSettings.xml

プロンプトで指示されたら、Visual SourceSafe の管理者パスワードを入力します。

3. 出力レポートを調べます。VSSAnalysisReport.xml という出力レポートと、Usermap.xml というユーザー マッピング ファイルが、変換ツールによって現行ディレクトリ内に作成されます。以下に説明されているとおり、ユーザー マッピング ファイルを使用して、Visual SourceSafe ユーザーを Team Foundation ユーザーにマップすることができます。

4. VSS ユーザーを TFS ユーザーにマップします。VSSConverter ツールは UserMapping.xml ファイルを作成します。この中には、VSS データベースに対して少なくとも 1 回の操作を実行したすべてのユーザーのリストが入っています。UserMapping.xml ファイルを編集し、関連した TFS ユーザー名を追加します。ドメインを付けてユーザー名を指定する必要があります (MyDomain¥MyUserName)。

ステップ 4 - プロジェクトの移行の準備

このステップでは、VSSConverter.exe ツールを使用して、VSS プロジェクトを移行します。

1. ステップ 3 で作成した ConversionSettings.xml ファイルを変更し、以下に示してあるとおり、**<TeamFoundationServer>** 要素を入れた新しい **<Settings>** 要素を追加します。**<Settings>** 要素を、**<SourceControlConverter>** の子要素として、**</ConverterSpecificSettings>** のすぐ後に追加します。

2. 作成した ConversionSettings.xml ファイルを変更して、次のように、**Destination** 属性を **<Project>** 要素に追加します。**Destination** 属性の値を、ファイルの移行先の TFS チーム プロジェクト内のフォルダ保管場所に設定します。

<ProjectMap> セクションで、移行しようとしている VSS フォルダごとに、**MyFirstProject** をソース VSS フォルダに置き換え、さらに、**MyTeam_ProjectOne** を、Team Foundation ソース管理内の移動先フォルダに置き換えます。移行したいすべてのフォルダに、**<Project>** エントリを付け加えます。

ステップ 5 - プロジェクトの移行

分析と移行を実行しようとしているコンピュータ上で、Visual SourceSafe データベースをローカルフォルダにコピーします。リモート コンピュータ上の共有フォルダに Visual SourceSafe データベースを移行することはできますが、移行の完了までの時間が長くなってしまいます。

移行を実行するには:

1. コマンド プロンプトで次のように入力します。

VSSConverter Migrate Conversionsettings.xml

2. **Y** を入力して、移行を確認します。プロンプトで指示されたら、Visual SourceSafe 管理者ユーザーのパスワードを入力します。

その他の考慮事項

共有、分岐、または固定を VSS で使用していた場合、次のような問題に注意する必要があります。

- **共有**。Team Foundation Server は共有をサポートしません。共有ファイルは、共有の開始時点のバージョンのファイルを移動先フォルダにコピーすることで移行されます。その共有ファイルに対してその後加えた変更内容は、共有コピーとオリジナル コピーの両方に対して複製されます。
- **分岐**。VSS での分岐は共有を使用するので、分岐ファイルを移行すると、TFS ソース管理内の移動先フォルダにそのファイルがコピーされることになります。分岐イベント後にいずれかの分岐に加えた変更は、Team Foundation ソース管理内の該当する各コピーに移行されます。
- **固定**。Team Foundation Server は固定をサポートしません。Team Foundation ソース管理内で、これまでに VSS データベースで固定されていたアイテムを容易に見つけ出せるよう、VSSConverter ツールでは、「PINNED」ラベルを使用して固定されていたすべてのファイルにラベルが付けられます。

パフォーマンスを最大化する (大型の VSS データベースの場合は特に重要) ためには、TFS サーバーコンピュータ上で変換を実行します。

参考資料

- 移行の準備の方法に関する詳細は、

http://msdn2.microsoft.com/ja-jp/library/ms181246(vs.80).aspx を参照してください。

- 移行の実行方法に関する詳細は、

http://msdn2.microsoft.com/ja-jp/library/ms181247(VS.80).aspx を参照してください。

- Visual SourceSafe コンバータの制限事項に関する詳細は、

http://msdn2.microsoft.com/ja-jp/library/ms252491(VS.80).aspx を参照してください。

- VSS ANALYZE ツールに関する詳細は、

http://msdn2.microsoft.com/ja-jp/library/ysxsfw4x(VS.80).aspx を参照してください。

• VSSConverter 移行コマンドに関する詳細は、

http://msdn2.microsoft.com/ja-jp/library/ms400685(VS.80).aspx を参照してください。

[HOWTO]: Visual Studio Team Foundation Server で基点のないマージを実行する方法

適用対象

• Visual Studio Team Foundation Server

概要

この [HOWTO] では、相互に関連し合っていない 2 つの分岐からファイルをマージする方法を示します。それぞれ相手からは直接分岐していない 2 つのアイテムをマージするプロセスを基点のないマージ と呼びます。基点のないマージを実行するには、Tf マージ コマンドを使用する必要があります。基点のないマージ操作を Visual Studio から実行することはできません。

目次

目的

- 基点のないマージを実行する方法の習得

概要

この [HOWTO] では、基点のないマージを行う方法を学習します。それぞれ相手からは直接分岐していない 2 つのアイテムをマージするプロセスを基点のないマージ と呼びます。たとえば、互いに兄弟である 2 つのリリース分岐の間で変更内容をマージしたいけれども、親分岐にまでさかのぼってマージしたくない場合があります。基点のないマージは、Tf.exe コマンド ライン ユーティリティを使用してのみ実行することができます。基点のないマージを Visual Studio から実行することはできません。基点のないマージの実行時には、Team Foundation Server には、分岐内のファイルの関係に関する情報はまったくありません。たとえば、ファイルの名前を変更した場合、それは分岐内の削除済みのファイルおよび新規のファイルとみなされます。こういった理由で、通常のマージを実行する場合よりも、多くの競合を手動で解決する必要が生じます。ただし、そのような競合の解決は 1 回実行するだけで済みます。基点のないマージの実行が完了すれば、Team Foundation Server にはヒストリが記録され、フォルダとファイルの間の関係が確立されます。

ステップの概要

- ステップ 1 - 基点のないマージが必要かどうかの評価
- ステップ 2 - Trif.exe を使用した基点のないマージの実行
- ステップ 3 - マージの競合の解決
- ステップ 4 - マージ後の変更のチェックイン

ステップ 1 - 基点のないマージが必要かどうかの評価

このステップでは、マージする分岐とアイテムを評価して、基点のないマージまたは通常のマージのどちらを実行する必要があるかを決定します。

チーム プロジェクトの所有者または管理者の場合、分岐またはアイテムの関係を知ることができます。Visual Studio からマージできるのは、親子関係にある分岐またはアイテムのみです。そのような関係にない分岐またはアイテムをプロジェクトで取り扱う場合、基点のないマージが必要になります。

分岐またはアイテムのすべての関係を把握しているわけではない場合、次のようにして、基点のないマ

ージを使用する必要があるかどうかを評価することができます。

- [**ソース コード エクスプローラ**] を開きます。
- 分岐フォルダを右クリックし、[**マージ**] をクリックします。
- [**ソース管理マージ ウィザード**] ダイアログ ボックスで、[**ターゲット分岐**] の下のドロップダウンを探します。
- マージしようとしている分岐の項目が見当たらない場合、それは、それらの分岐の間に関係はないことを示します。その場合、基点のないマージを実行する必要があります。

ステップ 2 - Tf.exe を使用した 基点のないマージの実行

このステップでは、Team Foundation Server コマンド ライン ツール Tf.exe を使用して、基点のないマージを実行します。

基点のないマージを実行するには:

1. マージする 2 つの分岐に対して最新バージョンの取得操作を行って、両方の分岐のワークスペースを設定します。
    1.1. [**ソース コード エクスプローラ**] を開きます。
    1.2. マージする一方の分岐のフォルダを右クリックし、[**最新のバージョンを取得**] をクリックします。
    1.3. マージするもう一方の分岐に対しても上記と同じステップを行います。
    ワークスペースをマップしておかないと、ローカル ドライブ上でフォルダを選択するよう Visual Studio からプロンプトで指示されることに注意してください。
2. Visual Studio のコマンド ウィンドウを開きます。
3. コマンド ラインから次のような Tf.exe コマンドを実行します。

Tf merge /baseless <<source path>> <<target path>> /recursive

例

Tf merge /baseless c:¥data¥proj1 c:¥data proj2 /recursive

特定のバージョンのコード変更をマージする必要がある場合、次のように、Tf.exe でバージョン スイッチを使用することができます。

tf merge /baseless <<source path>> <<target path>> / recursive /version:<<from Changeset>>~<<to Changeset>>

例

tf merge /baseless c:¥data¥proj1 c:¥data¥proj2 /recursive /version:C123~C125

ステップ 3 - マージの競合の解決

このステップでは、基点のないマージの実行中に発生する可能性のある競合を解決します。Tf.exe コマンドの実行が完了すると、競合するファイルのリストを示した [**競合の解決**] ダイアログ ボックスが表示されます。

マージの競合を解決するには:

1. 各ファイルを選択し、[**解決**] をクリックします。
2. [**バージョンの競合の解決**] ダイアログ ボックスで、内容の変更がない場合は、[**ターゲット分岐の変更を維持**] 解決オプションを選択し、[**OK**] をクリックします。
3. 内容の変更がある場合、[**マージ ツールで変更をマージ**] 解決オプションを選択してから、[**OK**] をクリックします。
4. マージ ツールで、上のペイン内の競合領域をクリックするか、または、下のペイン内に入力して、競合行ごとに変更を適用します。
5. すべての競合に対して変更を適用した後、マージ ツールの [**OK**] をクリックします。
6. すべての競合を解決したら、[**閉じる**] をクリックします。

ステップ 4 - マージ後の変更のチェックイン

このステップでは、基点のないマージで行なわれた変更をチェックインします。

マージ後の変更をチェックインするには:

1. [**ソース コード エクスプローラ**] を開きます。
2. 変更をマージした先のターゲット フォルダを右クリックし、[**保留中の変更のチェックイン**] をクリックします。
3. [**ソース ファイルのチェックイン**] ダイアログ ボックスで、チェックインするすべてのファイルが選択済みになっていることを確認します。
4. [**チェックイン**] をクリックします。

参考資料

- ファイルとフォルダをマージする方法の詳細は、「方法: ファイルおよびフォルダをマージする」(http://msdn2.microsoft.com/ja-jp/library/ms181428(VS.80).aspx) を参照してください。
- 基点のないマージに関する詳細は、「Merge コマンド」(http://msdn2.microsoft.com/ja-jp/library/bd6dxhfy(VS.80).aspx) の /baseless オプションを参照してください。
- その他のマージ オプションの詳細は、「Merge コマンド」(http://msdn2.microsoft.com/ja-jp/library/bd6dxhfy(VS.80).aspx) を参照してください。
- 「Team Foundation Server Branching Guidance」(http://www.CodePlex.com/BranchingGuidance) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server で常時結合をセットアップする方法

適用対象

- Visual Studio Team Foundation Server

概要

Team Foundation Server 2005 には、すぐに使用できる CI (Continuous Integration: 常時結合、もしくは常時結合) ソリューションは用意されていませんが、独自の CI ビルド ソリューションを実装するためのフレームワークは用意されています。この [HOWTO] では、Visual Studio Team System で提供されているソリューションを使用して、Team Foundation Server 2005 で常時結合ビルドをセットアップする方法を示します。このソリューションは、TFS サーバーにアクセスできるアカウントのもとで実行する Web サービスをインストールします。つまり、**CheckinEevnt** 用の Web サービスを登録する必要があります。それによって、チェックインの発生のたびに、Web サービスはチーム ビルドを開始します。

目次

目的

- 常時結合ビルドとは何かの習得
- 常時結合ビルドのセットアップの習得

概要

開発者は、実行したチェックインの品質に関する継続的なフィードバックを常に必要とします。行った変更が、他の開発者による変更と対立するかどうかに関するフィードバックが特に必要です。フィードバックは、迅速であればあるほど有用です。コードを修正する機会がすぐに開発者に与えられるからです。そうすれば、他の開発者やテスト担当者を妨げることなく、チームの全員の生産性が向上します。常時結合ビルドが登場するのは、このような場面です。これは、開発者がコードをソース管理にチェックインするたびに、ビルドを作成するプロセスです。CI は、開発者向けの迅速なフィードバックを実現します。それに加えて、ビルド品質を常に保つための品質の扉として CI ビルドを構成することもできます。CI ビルドはボトルネックの原因になる可能性があり、開発者のコードが喪失する危険性を招くことがあります。なぜなら、開発者は、部分的に完成したコードをチェックインできないからです。

ステップの概要

- ステップ 1 - ビルドの作成とテスト
- ステップ 2 - 常時結合アドオンのインストール
- ステップ 3 - 常時結合アドオンの構成
- ステップ 4 - CeckinEvent の登録
- ステップ 5 - 常時結合ビルドのテスト
- ステップ 6 - 電子メール通知のセットアップ

始める前に

ビルド サービスを実行するアカウントが、Team Foundation Server において次のようなアクセス権を持っていることを確認します。

- アクセス許可 Start a build = Allow

ステップ 1 - ビルドの作成とテスト

このステップでは、テスト CI ビルドを作成し、Visual Studio IDE 内からチーム ビルドが順調に稼働することを確認します。稼働しない場合、先に進む前に修正を加えます。

1. ビルドをテストするときに使用できる Windows フォーム プロジェクトを作成します。
2. プロジェクトをビルドし、正しく稼働することを確認します。
3. プロジェクトをソース管理にチェックインします。
4. チーム ビルドを作成します。
   - [チーム エクスプローラ] で、[**チーム ビルド**] を右クリックし、[**新しいチーム ビルドの種類**] をクリックします。
   - **チーム ビルドの種類の作成ウィザード**の説明に従って入力を完成します。
5. チーム ビルドが順調に稼働することを確認します。
   - [チーム エクスプローラ] で、作成したチーム ビルド プロジェクトをダブルクリックします。
   - Visual Studio の [**ビルド**] メニューで、[**チーム プロジェクト <<チーム ビルドの名前>> のビルド**] をクリックします。
   - 正しいビルドの種類を選択したことを確認してから、[**ビルド**] ボタンをクリックします。

6. ビルドの出力を調べて、ビルドの過程でエラーがないことを確認します。

重要事項 – 使用するビルド サービス アカウント (TFSService) が、**チーム ビルドの種類ウィザード**で指定した、共有のドロップ フォルダに対するフル コントロール アクセス権を持っていることを確認してください

。

ステップ 2 - 常時結合アドオンのインストール

このステップでは、CI ビルドのセットアップ用として Visual Studio Team System チームに用意されているソリューションをインストールします。このソリューションに関する詳細情報は、http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx を参照してください。

1. 以下の場所から TFS サーバーにソリューションをインストールします。
http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi
2. インストーラ ウィザードの 2 ページ目で、必ず「Team Foundation Server」サイトを選択します。
3. セットアップ インストールによって、Team Foundation Server 用のアプリケーション層サーバーのルートの下に、仮想ディレクトリが作成されます。これは、ポート 8080 上の「Team Foundation Server」という名前の IIS Web サイトのはずです。

ステップ 3 - 常時結合アドオンの構成

このステップでは、常時結合の設定を構成して、ビルドするプロジェクト、使用するビルド マシン、および使用するチーム ビルドの種類を指定します。

1. 仮想ルートが「TfsAppPool」アプリケーション プール内で稼働することを確認します。
   - [**スタート**] をクリックしてから [**管理ツール**] をクリックし、最後に [**IIS マネージャ**] を選択して、IIS マネージャを開きます。
   - [**Web サイト**] の下で、[**Team Foundation Server**] Web サイトを展開します。
   - [**CI Web アプリケーション**] を右クリックし、[**アプリケーション プール**] が **TfsAppPool** に設定されていることを確認します。
2. 常時結合の設定値を構成します。
   - C:¥Program Files¥Microsoft Visual Studio 2005 Team Foundation Server¥Web

Services¥CI¥ に置かれている **CI Web アプリケーション**の web.config ファイルを開きます。

- **<appsettings>** ノードの下に新しいキーを追加して、以下のプロパティを設定します。
  - o **TeamFoundationServer** — http://machine:8080 形式のアプリケーション層の URL。
  - o **TeamProject** — 常時結合を有効にしたいチーム プロジェクト。
  - o **BuildType** — 常時結合ビルドの作成中に使用するビルドの種類。通常、これは、ビルドのためにだけ構成します。また、基本レベルのテストおよび静的分析を追加することもできます。
  - o **Build Machine** —オプションのパラメータ。これを使用して、ビルドの種類に指定されている既定のビルド マシンを指定変更します。

以下に設定を構成する例を示してあります。

653.

```
<add key="1"
value="TeamServer=http://TFSRTM:8080;TeamProjectName=AdventureWorks;Bui
ldType=Test Build"/>
```

653.

ステップ 4 - CheckinEvent の登録

このステップでは、Team Foundation Server で使用できる **bissubscribe** ツールを使用して、**CheckinEvent** イベントの登録を行います。

1. コマンド プロンプトを開いて、C:¥Program Files¥Microsoft Visual Studio 2005 Team Foundation Server¥TF Setup¥ に移動します。
2. 以下のコマンドを実行します。

```
Bissubscribe /eventType CheckinEvent /address
http://TFSRTM:8080/ci/notify.asmx /deliveryType Soap /domain
http://TFSRTM:8080
```

3. エラーが表示された場合や、正しく登録したことを確認したい場合、次のようにします。

- **SQL Server Management Studio** を開きます。
- **tfsIntegration** データベースを開きます。
- **tbl_subscription** テーブルを開きます。

このテーブルには、既にサブスクライブ済みのすべてのイベントが記載されていますが、必要があれば、イベントをアンサブスクライブすることもできます。また、CI ソリューションが既にイベントにサブスクライブ済みであれば表示されます。

ステップ 5 - 常時結合ビルドのテスト

このステップでは、常時結合ビルドが正しく稼働することを検証します。

1. ステップ 1 で作成した Windows フォーム アプリケーションを開きます。
2. 若干の変更を加えます。ただし、ビルドが損なわれないように注意します。
3. 加えたコード変更をチェックインします。
4. ビルド マシンにアクセスできる場合は、タスク マネージャを開いて CPU 使用率を確認します。これは、チェックインの直後に上昇して、ビルドの発生を示すはずです。プロセス リストを見れば、msbuild および csc や、aspnet_compiler が CPU サイクルを使用していることが分かります。これは、ビルドが進行中であることを示します。
5. ビルドが完了するまで待機してから、[**チーム エクスプローラ**] 内の [**すべてのビルド の種類**] をダブルクリックし、完了したビルドを見つけ出します。

トラブル シューティング

ビルドがビルド リストに示されていない場合、次のようにしてトラブルシュートすることができます。

1. イベントを正しくサブスクライブしたことを確認します。詳しくは、ステップ 4 の中のサブステップ 3 を参照してください。
2. CI Web アプリケーションの web.config が正しく構成されていることを確認します。詳しくは、ステップ 3 を参照してください。
3. CI Web アプリケーションが正しいコンテキストで実行されていて、TFS サーバーにアクセスできることを確認します。
4. 上記がいずれも正しい場合に、ビルドが失敗するときは、次のようにして CI Web アプリケーションをデバッグすることができます。
    - 新しい Web サイト プロジェクトを作成します。
    - 既存の Web サイトを追加し、CI Web アプリケーションをローカル IIS から選択します。
    - notify.cs を開いて、**Notify** メソッド上にブレークポイントを設定します。
    - F5 をクリックしてデバッグを開始します。
    - Windows フォーム テスト アプリケーションを変更して、コードをソース管理にチェックインします。

o ブレークポイントがヒットしない場合、イベントが捕そくされていないということです。もう一度サブスクライブしてください。
o ブレークポイントがヒットする場合、デバッグを続けてください。
o スクリプトのデバッグが IE で有効になっている (拡張オプション) ことを確認してください。

ステップ 6 - 電子メール通知のセットアップ

このステップでは、ビルドの完了に対する電子メール通知をセットアップします。それによって、関係者に通知します。

1. [チーム エクスプローラ] 内の該当するチーム プロジェクトを右クリックします。
2. [**プロジェクト警告**] を選択します。
3. [**ビルドが完了したとき**] オプションをチェックし、通知用として電子メールアドレスを入力します。

参考資料

- 「Continuous Integration Using Team Foundation Build」(http://msdn2.microsoft.com/en-us/library/ms364045(VS.80).aspx)
- Visual Studio Team System の CI ソリューション MSI - (http://download.microsoft.com/download/6/5/e/65e300ce-22fc-4988-97de-0e81d3de2482/ci.msi)
- アジャイル開発記事 - （http://www.microsoft.com/japan/msdn/msdnmag/issues/06/03/TeamSystem/default.aspx）

[HOWTO]: Visual Studio Team Foundation Server でスケジュール ビルドをセットアップする方法

適用対象

- Visual Studio Team Foundation Server

概要

Team Foundation Server 2005 には、すぐに使用できるスケジュール ビルド機能は用意されていませんが、**TFSBuild** コマンドを使用して、スケジュール ビルドを実装することができます。この [HOWTO] では、**TFSBuild** コマンドおよび Windows タスク スケジューラを使用して、Team Foundation Server 2005 でスケジュール ビルドをセットアップする方法を示します。

目次

目的

- TFSBuild および Windows スケジューラを使用したスケジュール ビルドの作成

概要

開発チームにとって、プロジェクトでの進行状況を全員が共有し、タイムリーにフィードバックを受け取ることは重要です。それは、スケジュール ビルドを使用すれば可能になります。ソース コードをコンパイルし、たとえばテスト担当者などの、他のチームに対して定期的にリリースして、フィードバッ

クを得ることができます。スケジュール ビルドは、プロジェクトの規模や要件に応じて、毎晩、毎週、隔週、または月単位にすることができます。
Team Foundation Server のチーム ビルド機能は、すぐに使用できるスケジュール ビルドをサポートしません。ただし、コマンド ラインからの TFSBuild コマンド ユーティリティによるチーム ビルドの開始には対応します。Windows タスク スケジューラなどの任意のスケジューラを使用して、TFSBuild コマンド ユーティリティを実行し、事前に取り決めた時刻にビルドを開始することができます。

ステップの概要:

- ステップ 1 - ビルドの作成とテスト
- ステップ 2 - TFSBuild コマンド ラインの作成
- ステップ 3 - TFSBuild コマンド ラインのテスト
- ステップ 4 - バッチ ファイルの作成
- ステップ 5 - バッチ ファイルのテスト
- ステップ 6 - スケジュールされたタスクの追加
- ステップ 7 - スケジュールされたタスクのテスト

始める前に

ビルド サービスを実行するアカウントが、Team Foundation Server において次のようなアクセス権を持っていることを確認します。

- アクセス許可 - Start a build = Allow

ステップ 1 - ビルドの作成とテスト

このステップでは、テスト スケジュール ビルドを作成し、Visual Studio IDE 内からチーム ビルドが順調に稼働することを確認します。稼働しない場合、先に進む前に修正を加えます。

1. ビルドをテストするときに使用できる Windows フォーム プロジェクトを作成します。
2. プロジェクトをビルドし、正しく稼働することを確認します。
3. プロジェクトをソース管理にチェックインします。

4. チーム ビルドを作成します。
   - [チーム エクスプローラ] で、[**チーム ビルド**] を右クリックし、[**新しいチーム ビルドの種類**] をクリックします。
   - **チーム ビルドの種類の作成ウィザード**の説明に従って入力を完成します。
5. チーム ビルドが順調に稼働することを確認します。
   - [チーム エクスプローラ] で、作成したチーム ビルド プロジェクトをダブルクリックします。
   - Visual Studio の [**ビルド**] メニューで、[**チーム プロジェクト <<チーム ビルドの名前>> のビルド**] をクリックします。
- 正しいビルドの種類を選択したことを確認してから、[**ビルド**] ボタンをクリックします。
6. ビルドの出力を調べて、ビルドの過程でエラーがないことを確認します。

重要事項 – 使用するビルド サービス アカウント (TFSService) が、**チーム ビルドの種類ウィザード**で指定した、共有のドロップ フォルダに対するフル コントロール アクセス権を持っていることを確認してください。

ステップ 2 - TFSBuild コマンド ラインの作成

このステップでは、ビルドを開始するための **TFSBuild** ユーティリティ用のコマンドを作成します。

1. TFSBuild コマンド ユーティリティは、開始のためのパラメータをいくつか必要とします。まず、使用するパラメータを判別する必要があります。
   - **Team Foundation Server** – ビルドするソリューションのチェックイン先の、Team Foundation サーバーの URL。
   - **チーム プロジェクト** – ビルドするチーム プロジェクトの名前。
   - **ビルドの種類** – ビルドの種類ウィザードを使用して作成したビルドの種類。
   - **ビルド マシン** – プロジェクトのビルドに使用するビルド サーバーの名前。これはオプションのパラメータです。既定では、チーム ビルドの種類で指定したビルド サーバーが使用されます。
   - **ビルド ディレクトリ** – ビルドを実施するディレクトリのパス。これはオプションのパラメータです。既定では、チーム ビルドの種類に指定したパスが使用されます。
2. 作成したビルド コマンドは次のようになります。

```
TfsBuild start <<TeamFoundationServer>> <<TeamProject>>
<<BuidlTypeName>>
```

ビルドの種類の作成中に指定したビルド マシンの名前とビルド ディレクトリ パスを変更したい場合、

作成するコマンドは次のようになります。

TfsBuild start <<TeamFoundationServer>> <<TeamProject>>
<<BuidlTypeName>> /m:<<MachineName>> /d:<<BuildDirectory>>

ステップ 3 - TFSBuild コマンド ラインのテスト

このステップでは、TFS コマンドが正しく稼働することを検証します。

1. Visual Studio のコマンド プロンプトを開きます。
2. ステップ 2 で作成した **TfsBuild** コマンドを入力します。
3. 出力を確認して、ビルドが正しく稼働することを確認します。

ステップ 4 - バッチ ファイルの作成

このステップでは、ビルドをスケジューリングするときに使用するバッチ ファイルを作成します。

1. メモ帳を開いて以下のコマンドを入力します。Windows コマンド プロンプトから実行できるように、TFSBuild.exe ファイルの完全パスが指定されていることに注意してください。以下にコマンドの例を示します。

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

ビルド マシンとビルド ディレクトリ パスを変更する場合、バッチ ファイル内のコマンドは次のようになります。

"C:¥Program Files¥Microsoft Visual Studio 8¥Common7¥IDE¥TFSBuild" start
<<TeamFoundationServer>> <<TeamProject>> <<BuildTypeName>>
/m:<<BuildMachineName>> /d:<<BuildDirectoryPath>>

2. このファイルを、たとえば batchbuild.bat といったバッチ ファイルとして保存します。
3. TFS サーバーのソース管理内の、たとえば **Main¥Scripts** などのビルド スクリプトの保管場所にこのファイルを入れます。

ステップ 5 - バッチ ファイルのテスト

このステップでは、作成したバッチ ファイルが正しく稼働することを検証します。

1. VS コマンド プロンプトではなく、Windows コマンド プロンプトを開きます。既定では、

Windows スケジューラは Windows コマンド プロンプト内でバッチ ファイルを実行するからです。

2. コマンド ラインでバッチ ファイルを開始します。

3. ビルドがエラーを生じないで正しく稼働することを確認します。

ステップ 6 - スケジュールされたタスクの追加

このステップでは、プロジェクトに適した必要な一定間隔でビルド コマンドを実行するためのスケジュールされたタスクを追加します。

1. [**スタート**] メニューから、[**コントロール パネル**] をクリックします。

2. 次に、[**タスク**] オプションを選択し、[**スケジュールされたタスクの追加**] をクリックします。

3.**タスク ウィザード**で、[**次へ**] をクリックします。

4. [**参照**] ボタンをクリックして、ステップ 4 で作成したバッチ ファイルを選択します。

5. タスクの名前を入力します。

6. たとえば、[**日単位**] などの、タスクを実行する頻度を選択してから、[**次へ**] をクリックします。

7. [**開始時刻**]、[このタスクの実行]、および [**開始日**] を構成して、[**次へ**] をクリックします。

8. ビルドの開始の許可を持ったアカウントのユーザー名とパスワードを入力し、[**次へ**] をクリックします。

9. 次に、[**完了**] をクリックします。

ステップ 7 - スケジュールされたタスクのテスト

このステップでは、スケジュールされたタスクが正しい時刻に実行されて、ビルドが正しく稼働することを検証します。

1. スケジュールされたタスクが実行される予定の時刻を待ちます。あるいは、[**スタート**] メニューから、[**コントロール パネル**] をクリックします。

2. 次に、[**タスク**] オプションを選択し、リスト上のスケジュールされたタスクを右クリックし、[**実行**] を選択します。

3. コマンド プロンプトが表示されて、ビルドが開始します。

4. タスクの実行予定時刻にビルド マシンにアクセスできない場合でも、後で調べてビルドが完了したかどうかを確認することができます。

- [**チーム エクスプローラ**] で、[**すべてのビルドの種類**] をダブルクリックします。

- ビルドのリストを調べて、スケジュールどおりの時刻にビルドが完了したかどうかを確認します。

参考資料

- 詳細は、「方法: スケジュールされたビルドを構成する (コマンド ライン)」(http://msdn2.microsoft.com/ja-jp/library/ms181727(VS.80).aspx) を参照してください。

[HOWTO]: Team Foundation Server 用に ASP.NET アプリケーションを構造化する方法

適用対象

- Team Foundation Server
- Visual Studio Team System
- ASP.NET アプリケーション

概要

この [HOWTO] では、Team Foundation Server 向けに ASP.NET Web アプリケーションを編成して構造化する方法を示します。ここでは、Team Foundation Server ソース管理内で使用するのに適したソース ツリー構造について説明します。

目次

- ステップ 5 - ソリューション構造のチェック
- ステップ 6 - ローカル フォルダの構造のチェック
- ステップ 7 - ソース管理へのソリューションの追加
- 共有コードに関する考慮事項
- 参考資料

目的

- Team Foundation Server ソース管理での ASP.NET アプリケーションの構造化の方法の習得。

概要

この [HOWTO] では、ASP.NET Web アプリケーションに適したソース管理フォルダ構造を作成する方法を説明します。多くの場合、ASP.NET Web プロジェクトには追加のクラス ライブラリが組み込まれているので、それらに対処する構造が必要になります。ASP.NET Web プロジェクトを保守するフォルダは、ソース管理内の最上位構造 **/Main/Source** の下に置かれます。そのおかげで、独立した開発やリリースの保守用として分岐を作成する必要が生じた場合に、追加の **Development** および **Maintenance** フォルダを簡単に使用することができます。この最上位フォルダ構造の作成に関する詳細は、「[How To]: ソース管理フォルダを構造化する方法」を参照してください。

ステップの概要

- ステップ 1 - Web プロジェクト用のローカル フォルダの作成
- ステップ 2 - 空のソリューションの作成
- ステップ 3 - ソリューションへの Web サイトの追加
- ステップ 4 - クラス ライブラリの追加 (オプション)
- ステップ 5 - ソリューション構造のチェック
- ステップ 6 - ローカル フォルダの構造のチェック
- ステップ 7 - ソース管理へのソリューションの追加

ステップ 1 - Web プロジェクト用のローカル フォルダの作成

このステップでは、開発コンピュータ上で、Web プロジェクトに適したローカル フォルダ構造を作成します。チーム開発時に一貫したアプローチを堅持し、開発コンピュータ上でチーム プロジェクトを手際よく整理整頓しておくために、作業の対象であるすべてのチーム プロジェクトの開発ソース全体を 1 つのグループとして、たとえば C:¥DevProjects などの単一のルート フォルダの下にまとめておく必要があります。

C:¥DevProjects などの最上位フォルダを作成します。

ステップ 2 - 空のソリューションの作成

ASP.NET Web アプリケーションを作成するには、まず Visual Studio ソリューション (.sln) ファイルを明示的に作成してから、Web サイトや、クラス ライブラリなどの必要と考えられる補足的なプロジェクトを追加します。以下のステップでは、最上位の C:¥DevProjects フォルダの下にソリューションを作成します。

1. [**ファイル**] メニューで、[**新規作成**] をポイントして、[**プロジェクト**] をクリックします。
2. [**その他のプロジェクトの種類**] を展開し、[**Visual Studio ソリューション**] を選択します。
3. [**空のソリューション**] をクリックします。
4. ソリューションに **MyWebAppSln** という名前を付けます。
5. [**場所**] を C:¥DevProjects に設定し、[**OK**] をクリックします。

これで、C:¥DevProjects¥MyWebAppSln フォルダが作成されます。Visual Studio によって、ソリューション (.sln) ファイルとソリューション ユーザー オプション (.suo) ファイルが、このフォルダに追加されます。その後、.sln ファイルのみが、ステップ 7 でソース管理に追加されることに注意してください。

ステップ 3 - ソリューションへの Web サイトの追加

このステップでは、ASP.NET Web サイトをソリューションに追加します。Visual Studio 開発 Web サーバーを使用する新規のファイル システム ベースの Web サイトを作成する場合と、IIS を使用する HTTP Web サイトを作成する場合とでは、サイトを追加する方法は若干異なります。

ファイル システム

ファイル システム ベースの Web プロジェクトをソリューションに追加するには:

1. [**ソリューション エクスプローラ**] で、[**ソリューション MyWebAppSln**] を右クリックし、[**追加**] をポイントしてから、[**新しい Web サイト**] をクリックします。
2. [**新しい Web サイトの追加**] ダイアログ ボックスで、[**場所**] を [**ファイル システム**] のままに残し、[**言語**] を [**Visual C#**] のままにします。
3. [**場所**] ディレクトリを C:¥DevProjects¥MyWebAppSln¥Source¥MyWebAppWeb に設定します。
4. [**OK**] をクリックして、[**新しい Web サイトの追加**] ダイアログ ボックスを閉じます。

ここの例で Web という接尾部が使用されているのは、フォルダが Web サイト ルート フォルダであることを明確に示すためであることに注意してください。

HTTP

IIS を使用する HTTP ベースの ASP.NET Web サイトを作成するには、まず仮想ディレクトリを明示的に作成します。それにより、使用する Web サイト ディレクトリは、¥inetpub¥wwwroot ではなく、定義した場所になります。

Web サイトの仮想ディレクトリを作成するには:

1. Windows エクスプローラを使用して、C:¥DevProjects¥MyWebAppSln¥Source を参照します。
2. この場所の下で、**MyWebAppWeb** という名前の新しいフォルダを作成します。
3. [**MyWebAppWeb**] を右クリックし、[**共有とセキュリティ**] をクリックします。
4. [**Web 共有**] タブをクリックします。
5. [**このフォルダを共有する**] をクリックします。
6. [**エイリアス**] を [**MyWebAppWeb**] のまま変更しないで残し、既定の [**アクセス許可**] と [**アプリケーション許可**] はそのままにして、[**OK**] をクリックします。
7. [**OK**] をクリックしてから、もう一度 [**OK**] をクリックします。

Web サイトをソリューションに追加するには:

1. [**ソリューション エクスプローラ**] で、[**ソリューション MyWebAppSln**] を右クリックし、[**追加**] をポイントしてから、[**新しい Web サイト**] をクリックします。
2. [**新しい Web サイトの追加**] ダイアログ ボックスで、[**場所**] を [**HTTP**] に設定し、[**言語**] を [**Visual C#**] のままにします。

3. [**場所**] の URL を http://localhost/MyWebAppWeb に設定します。
4. [**OK**] をクリックして、[**新しい Web サイトの追加**] ダイアログ ボックスを閉じます。
Visual Studio によって、Default.aspx、Default.aspx.cs ファイルが C:¥DevProjects¥MyWebAppSln¥Source¥MyWebAppWeb フォルダに追加されて、**Bin** および **App_Data** 子フォルダが作成されます。

ステップ 4 - クラス ライブラリの追加 (オプション)

追加のクラス ライブラリが Web アプリケーションで必要な場合、次のようにして追加します。
1. [**ソリューション エクスプローラ**] で、[**MyWebAppSln**] ソリューションを右クリックし、[**追加**] をポイントしてから、[**新しいプロジェクト**] をクリックします。
2. [**Visual C#**] をプロジェクトの種類として選択し、[**クラス ライブラリ**] をテンプレートとして選択します。
3. 名前 **ClassLibrary** を入力し、[**場所**] を C:¥DevProjects¥MyWebAppSln¥Source に設定してから、[**OK**] をクリックします。
新しいプロジェクトはすべて、**Source** フォルダの下に追加されます。

ステップ 5 - ソリューション構造のチェック

ソリューション構造を検証するには、ソリューション エクスプローラを使用します。以下のようになっているはずです。

ステップ 6 - ローカル フォルダの構造のチェック

ローカル フォルダ構造を検証するには、Windows エクスプローラを使用します。以下のようになっているはずです。

ステップ 7 - ソース管理へのソリューションの追加

このステップでは、Team Foundation Server ソース管理にソリューションを追加します。
1. ソリューションを右クリックし、[**ソリューションをソース管理に追加**] をクリックします。
2. [**ソリューション MyWebAppSln をソース管理に追加**] ダイアログ ボックスで、チーム プロジェ

クトを選択します。

3. このダイアログ ボックスで、[**フォルダの新規作成**] をクリックし、新しいフォルダを **Main** と命名します。

4. 新しく作成した [**Main**] フォルダを選択し、[**フォルダの新規作成**] をクリックし、新しいフォルダを **Source** と命名します。

5. 新しく作成した [**Source**] フォルダを選択し、[**OK**] をクリックします。

6. [**チーム エクスプローラ**] 内の [**ソース管理**] をクリックして、[ソース管理] フォルダをチェックします。以下のようになっているはずです。

7. この時点で、保留中の変更を表示し、ソリューション ソース ファイルをサーバーにチェックインすることができます。そのためには、[**表示**] メニューから、[**その他のウィンドウ**] を指して、[**保留中の変更**] をクリックします。チェックインするプロジェクトとソース ファイルを選択し、チェックインのコメントを入力して、[**チェックイン**] をクリックします。

共有コードに関する考慮事項

共有ソース コードを参照する ASP.NET Web アプリケーションの場合、主に次のような 2 つの選択肢が考えられます。

- **共有場所のコードの参照**
- **共有コードの分岐**

共有場所のコードの参照

このアプローチをとる場合、たとえば別のチーム プロジェクトなどの共有場所のソースを、開発コンピュータ上のワークスペースに対応付けます。これにより、共有場所の共有ソースと、開発コンピュータ上のプロジェクト コードを統一する構成が作成されます。

このアプローチの長所は、最新バージョンのソースをワークスペースに取り込むたびに、共有ソースに加えられた変更もすべて取り込まれる点にあります。たとえば、共有ソースを取り扱う Common という名前のチーム プロジェクトを考察してみましょう。この場所のコードを参照するには、両方のチーム プロジェクトを、開発コンピュータ上の同じ場所のパスに対応付けます。たとえば、次のようにします。

- C:¥DevProjects¥MyWebAppSln¥
- C:¥DevProjects¥SharedCommon¥

次のようなワークスペース マッピングを使用します。

ソース管理 フォルダ ローカル フォルダ

$/MyTeamProject1/Main/Source/MyWebAppApp C:¥DevProjects¥MyWebAppSln

$/MyTeamProject2/Main/Source/Common C:¥DevProjects¥Common

詳細は、「Working with multiple team projects in Team Build」(http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx) を参照してください。

共有コードの分岐

このアプローチをとる場合、共有のチーム プロジェクトからご自分のチーム プロジェクトに共有コードを分岐します。この場合も、共有場所のソースとプロジェクトを統一する構成が作成されます。

ただし、このアプローチの場合、共有ソースの変更は、2 つの分岐のマージ プロセスの一環として取り込まれる点が異なります。それに起因して、共有ソースにおける変更を取り込む決定をもっと明示的に下せるようになります。最新の変更を取り込むためのマージをいつ実行するかは、ご自分で決定します。

たとえば、共有ソースを取り扱う **Common** というチーム プロジェクトについて考察してみましょう。この場所からコードを分岐するには:

1. [**ソース管理エクスプローラ**] で、**Common** チーム プロジェクトのルート フォルダを右クリックします。
2. [**分岐...**] をクリックします。
3. [**分岐**] ダイアログ ボックスで、[**ターゲット:**] を $/MyTeamProject1/Main/Source/ チーム プロジェクトのルート フォルダに設定し、[**OK**] をクリックします。
4. 分岐操作が完了したら、分岐後のソース コードを忘れずにチェックインします。

参考資料

- 「Team Foundation Server Branching Guidance」(http://www.CodePlex.com/BranchingGuidance)
- ワークスペースの作成に関する詳細は、「方法: ワークスペースを作成する」(http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server 用に Windows アプリケーションを構造化する方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、Team Foundation Server 向けに Windows フォーム アプリケーションを編成して構造化する方法を示します。ここでは、Team Foundation Server ソース管理内で使用するのに適したソース ツリー構造について説明します。

目次

目的

• Team Foundation Server ソース管理での Win フォーム アプリケーション構造化の方法の習得。

概要

この [HOWTO] では、Windows フォーム アプリケーションに適したソース管理フォルダ構造を作成する方法を説明します。多くの場合、Windows フォーム プロジェクトには追加のクラス ライブラリおよびコントロール ライブラリが組み込まれているので、それらに対処する構造が必要になります。Windows フォーム プロジェクトを保守するフォルダは、最上位構造 **/Main/Source** の下に置かれます。そのおかげで、独立した開発やリリースの保守用として分岐を作成する必要が生じた場合に、追加の **Development** および **Maintenance** フォルダを簡単に使用することができます。この最上位フォルダ構造の作成に関する詳細は、「[How To]: ソース管理フォルダを構造化する方法」を参照してください。

ステップの概要

- ステップ 1 - Windows フォーム プロジェクト用のローカル フォルダの作成
- ステップ 2 - 空のソリューションの作成
- ステップ 3 - ソリューションへの Windows フォーム プロジェクトの追加
- ステップ 4 - コントロール ライブラリの追加 (オプション)
- ステップ 5 - クラス ライブラリの追加 (オプション)
- ステップ 6 - ソリューション構造のチェック
- ステップ 7 - ローカル フォルダの構造のチェック
- ステップ 8 - ソース管理へのソリューションの追加

ステップ 1 - Windows フォーム プロジェクト用のローカル フォルダの作成

このステップでは、開発コンピュータ上で、Windows フォーム プロジェクトに適したローカル フォルダ構造を作成します。チーム開発時に一貫したアプローチを堅持し、開発コンピュータ上でチーム プロジェクトを手際よく整理整頓しておくために、作業の対象であるすべてのチーム プロジェクトの開発ソース全体を 1 つのグループとして、たとえば C:¥DevProjects などの単一のルート フォルダの

下にまとめておく必要があります。

C:¥DevProjects などの最上位フォルダを作成します。

ステップ 2 - 空のソリューションの作成

Windows フォーム アプリケーションを作成するには、まず Visual Studio ソリューション (.sln) ファイルを明示的に作成してから、Windows フォーム プロジェクトや、クラス ライブラリまたはコントロール ライブラリなどの必要と考えられる補足的なプロジェクトを追加します。以下のステップでは、最上位の C:¥DevProjects の下にソリューションを作成します。

1. [**ファイル**] メニューで、[**新規作成**] をポイントして、[**プロジェクト**] をクリックします。
2. [**その他のプロジェクトの種類**] を展開し、[**Visual Studio ソリューション**] を選択します。
3. [**空のソリューション**] をクリックします。
4. ソリューションに **MyApp**という名前を付けます。
5. [**場所**] を C:¥DevProjects に設定し、[**OK**] をクリックします。

これで、C:¥DevProjects¥MyApp フォルダが作成されます。Visual Studio によって、ソリューション (.sln) ファイルとソリューション ユーザー オプション (.suo) ファイルが、このフォルダに追加されます。その後、.sln ファイルのみが、ステップ 8 でソース管理に追加されることに注意してください。

ステップ 3 - ソリューションへの Windows フォーム プロジェクトの追加

このステップでは、新しい Windows フォーム プロジェクトをソリューションに追加します。

1. [**ソリューション エクスプローラ**] で、[**ソリューション MyApp**] を右クリックし、[**追加**] をポイントしてから、[**新しいプロジェクト...**] をクリックします。
2. [**新しいプロジェクトの追加**] ダイアログ ボックスで、[**Visual C#**] をプロジェクトの種類として選択し、[**Windows アプリケーション**] をテンプレートとして選択します。
3. [**場所**] を C:¥DevProjects¥MyApp¥Source に設定します。
4. プロジェクトの名前を **MyWinFormApp** に設定します。
5. [**OK**] をクリックして、[**新しいプロジェクトの追加**] ダイアログ ボックスを閉じ、プロジェクトを追加します。

ステップ 4 - コントロール ライブラリの追加 (オプション)

追加のコントロール ライブラリが Windows フォーム アプリケーションで必要な場合、次のようにして追加します。

1. [**ソリューション エクスプローラ**] で、ソリューションを右クリックし、[**追加**] をポイントしてから、[**新しいプロジェクト...**] をクリックします。
2. [**新しいプロジェクトの追加**] ダイアログ ボックスで、[**Visual C#**] をプロジェクトの種類として選択し、[**Windows コントロール ライブラリ**] をテンプレートとして選択します。
3. [**場所**] を C:¥DevProjects¥MyApp¥Source に設定します。
4. Windows コントロール ライブラリの名前を **ControlLibrary** に設定します。
5. [**OK**] をクリックして、[**新しいプロジェクトの追加**] ダイアログ ボックスを閉じます。

ステップ 5 - クラス ライブラリの追加 (オプション)

追加のクラス ライブラリ プロジェクトが Windows フォーム アプリケーションで必要な場合、次のようにして追加します。

1. [**ソリューション エクスプローラ**] で、ソリューションを右クリックし、[**追加**] をポイントしてから、[**新しいプロジェクト...**] をクリックします。
2. [**新しいプロジェクトの追加**] ダイアログ ボックスで、[**Visual C#**] をプロジェクトの種類として選択し、[**クラス ライブラリ**] をテンプレートとして選択します。
3. [**場所**] を C:¥DevProjects¥MyApp¥Source に設定します。
4. Windows クラス ライブラリの名前を **ClassLibrary1** に設定します。
5. [**OK**] をクリックして、[**新しいプロジェクトの追加**] ダイアログ ボックスを閉じます。

ステップ 6 - ソリューション構造のチェック

ソリューション構造を検証するには、ソリューション エクスプローラを使用します。以下のようになっているはずです。

ステップ 7 - ローカル フォルダの構造のチェック

ローカル フォルダ構造を検証するには、Windows エクスプローラを使用します。以下のようになっているはずです。

ステップ 8 - ソース管理へのソリューションの追加

このステップでは、Windows フォーム プロジェクトおよびオプションのクラス ライブラリおよびコントロール ライブラリの入ったソリューションを Team Foundation Server ソース管理に追加します。

1. ソリューション **MyApp** を右クリックし、[**ソリューションをソース管理に追加**] をクリックします。
2. [**ソリューション MyApp をソース管理に追加**] ダイアログ ボックスで、チーム プロジェクトを選択します。
3. このダイアログ ボックスで、[**フォルダの新規作成**] をクリックし、新しいフォルダを **Main** と命名します。
4. 新しく作成した [**Main**] フォルダを選択し、[**フォルダの新規作成**] をクリックし、新しいフォルダを **Source** と命名します。

5. 新しく作成した [**Source**] フォルダを選択し、[**OK**] をクリックします。
6. [**チーム エクスプローラ**] 内の [**ソース管理**] をダブルクリックして、[ソース管理] フォルダの構造をチェックします。これで、ソース管理エクスプローラが表示されます。構造は、以下のようになっているはずです。
7. この時点で、保留中の変更を表示し、ソース ファイルをサーバーにチェックインすることができます。そのためには、[**表示**] メニューから、[**その他のウィンドウ**] をポイントして、[**保留中の変更**] をクリックします。チェックインするプロジェクトとソース ファイルを選択し、チェックインのコメントを入力して、[**チェックイン**] をクリックします。

共有コードに関する考慮事項

共有ソース コードを参照する Windows フォーム アプリケーションの場合、主に次のような 2 つの選択肢が考えられます。

- **共有場所のコードの参照**

- **共有コードの分岐**

共有場所のコードの参照

このアプローチをとる場合、たとえば別のチーム プロジェクトなどの共有場所のソースを、開発コンピュータ上のワークスペースに対応付けます。これにより、共有場所の共有ソースと、開発コンピュータ上のプロジェクト コードを統一する構成が作成されます。

このアプローチの長所は、最新バージョンのソースをワークスペースに取り込むたびに、共有ソースに加えられた変更もすべて取り込まれる点にあります。たとえば、共有ソースを取り扱う Common という名前のチーム プロジェクトを考察してみましょう。この場所のコードを参照するには、両方のチーム プロジェクトを、開発コンピュータ上の同じ場所のパスに対応付けます。たとえば、次のようにします。

- C:¥MyProjects¥MyApp¥
- C:¥MyProjects¥SharedCommon¥

次のようなワークスペース マッピングを使用します。

ソース管理 フォルダ ローカル フォルダ

$/MyTeamProject1/Main/Source/MyApp C:¥DevProjects¥MyApp

$/MyTeamProject2/Main/Source/Common C:¥DevProjects¥Common

詳細は、「Working with multiple team projects in Team Build」(http://blogs.msdn.com/manishagarwal/archive/2005/12/22/506635.aspx) を参照してください。

共有コードの分岐

このアプローチをとる場合、共有のチーム プロジェクトからご自分のチーム プロジェクトに共有コードを分岐します。この場合も、共有場所のソースとプロジェクトを統一する構成が作成されます。

ただし、このアプローチの場合、共有ソースの変更は、2 つの分岐のマージ プロセスの一環として取り込まれる点が異なります。それに起因して、共有ソースにおける変更を取り込む決定をもっと明示的にくだせるようになります。最新の変更を取り込むためのマージをいつ実行するかは、ご自分で決定します。

たとえば、共有ソースを取り扱う **Common** というチーム プロジェクトについて考察してみましょう。この場所からコードを分岐するには:

1. [**ソース管理エクスプローラ**] で、**Common** チーム プロジェクトのルート フォルダを右クリックします。

2. [**分岐...**] をクリックします。

3. [**分岐**] ダイアログ ボックスで、[**ターゲット:**] を $/MyTeamProject1/Main/Source/ チーム プロジェクトのルート フォルダに設定し、[**OK**] をクリックします。

4. 分岐操作が完了したら、分岐後のソース コードを忘れずにチェックインします。

完全なフォルダ構造

単体テスト、ビルド出力、ドキュメンテーション、ビルド スクリプト、およびテスト用の追加のフォルダを備えたさらに完全なソース管理内のフォルダ構造は次のようになります。

/Main　複数のプロジェクトにまたがる .sln ファイルが入っています。

/Source

/**MyApp**　MyApp.sln ファイルが入っています。

/Source

/**MyWinFormApp**　MyWinFormApp.csproj およびフォームが入っています。

/**ControlLibrary**　ControlLibrary.csproj コントロール クラス ファイルなどが入っています。

/**ClassLibrary1**　ClassLibrary1.csproj およびその他のクラス ファイルが入っています。

/**UnitTests**

/**MyWinFormsAppTests**　MyWinFormsApp 用の単体テスト プロジェクトが入っています。

/**ControlsLibraryTests**　ControlsLibrary 用の単体テストが入っています。

/**ClassLibrary1Tests**　ClassLibrary1 用の単体テストが入っています。

/**Build**　ビルド出力 (バイナリ) が入っています。

/**Docs**　デザイン ドキュメントなどが入っています。

/**Script**　ビルド スクリプトやデータベース スクリプトなどが入っています。

/**Tests**　テスト用のコンテナ フォルダ

/FunctionalTests　機能テスト スクリプトおよびコードが入っています。

/PerformanceTests　パフォーマンス テスト スクリプトおよびコードが入っています。

/SecurityTests　セキュリティ テスト スクリプトおよびコードが入っています。

参考資料

- 「Team Foundation Server Branching Guidance」

(http://www.CodePlex.com/BranchingGuidance)

- ワークスペースの作成に関する詳細は、「方法: ワークスペースを作成する」

(http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx) を参照してください。

[HOWTO]: Visual Studio Team Foundation Server でソース管理フォルダを構造化する方法

適用対象

- Visual Studio Team Foundation Server

概要

この [HOWTO] では、実践的経験に基づいて Team Foundation Server でフォルダを構造化する方法を示します。ここに示したフォルダ構造は 1 つの出発点です。構造以上に重要なのは、その背後にある理論的解釈です。理論的解釈を参考にして、各自のシナリオに適したフォルダ構造およびフォルダ命名規則を評価します。

目次

目的

- ソース管理フォルダを構造化し命名する方法の習得。

概要

この [HOWTO] では、最も一般的なプロジェクトの種類での使用に適したソース管理フォルダ構造を作成する方法を説明します。この [HOWTO] で説明されているフォルダ構造では、次のような 3 つの最上位フォルダが使用されています。

• **Main**。これは、メインのルート フォルダです。これは、メイン ソース ツリーのほかに、設計ドキュメント、スクリプト、およびテスト ケースなどの、プロジェクトに付随した成果物を収容するコンテナ フォルダとして機能します。**Main** フォルダには、Visual Studio ソリューション (.sln) ファイルも収容されます。

• **Development**。これは、機能またはチームごとに独自のフォルダを作成する必要が生じた場合の、オプションのルート レベル フォルダです。このフォルダは、**Main** から分岐することで作成されます。

• **Maintenance**。これは、たとえば特定のリリース向けの継続保守のために分離する必要があるリリース ビルド用のオプションのルート レベル フォルダです。

この [HOWTO] で説明されているフォルダ構造を以下に要約してあります。

/Main　複数のプロジェクトにまたがるソリューション用のソリューション (.sln) ファイルを入れることができます。

/Source

/MyApp1　MyApp1.sln ファイルが入っています。

/Source　すべてのソースのフォルダが入っています。

/ClassLibrary1　ClassLibrary1.csproj が入っています。

/MyApp1Web　Default.aspx が入っています。

/Build　ビルド出力 (バイナリ) が入っています。

/Docs　製品ドキュメントなどが入っています。

/Scripts　ビルド スクリプトが入っています。

/Tests　テスト用のコンテナ

/Development

/FeatureBranch1

/Source

/MyApp1

/Source

/MyApp1Web

/ClassLibrary1

/FeatureBranch2

/**Maintenance**

/Release1

/Source

/MyApp1

/Source

/MyApp1Web

/ClassLibrary1

/Release 1.1

/Release 1.2

このフォルダ構造は、お勧めする一連のフォルダ名の 1 つの出発点であることに注意してください。構造および命名規則以上に重要なのは、その背後にある理論的解釈であり、さまざまなフォルダのそれぞれの用途です。この構造の理論的解釈の詳細は、第 5 章「Defining Your Branching and Merging Strategy」を参照してください。

ステップの概要

- ステップ 1 - ワークスペース マッピングの作成
- ステップ 2 - **Main** フォルダの作成
- ステップ 3 - プロジェクト成果物のフォルダの作成
- ステップ 4 - ソース ツリーへのソリューションの追加
- ステップ 5 - 分離のための分岐 **Development** フォルダの作成に関する考慮事項
- ステップ 6 - 分離に必要なリリース ビルド用の **Maintenance** フォルダの作成に関する考慮事項

ステップ 1 - ワークスペース マッピングの作成

このステップでは、TFS サーバー上のフォルダ構造と、クライアント上のフォルダ構造の間のマッピングを定義するワークスペース マッピングを作成します。これは、ソース ツリー構造の作成を可能にするために行う必要があります。まず、ソース ツリー構造がクライアント上のワークスペース内に作成された後、Team Foundation Server へのチェックインを実行する必要があります。
以下の 2 通りの方法のいずれかでワークスペース マッピングを作成することができます。

- **ワークスペース マッピングを明示的に設定する。**
- **チーム プロジェクトに対して取得操作を実行する。**

ワークスペース マッピングを明示的に設定するには:

1. Visual Studio で [**ファイル**] メニューから、[**ソース管理**] をポイントして、[**ワークスペース**] をクリックします。
2. [**ワークスペースの管理**] ダイアログ ボックスで、コンピュータ名を選択し、[**編集**] をクリックします。
3. [**ワークスペースの編集**] ダイアログ ボックスの [**作業フォルダ**] リストで、[**新しい作業フォルダに入力するにはここをクリック**] をクリックします。
4. 省略記号ボタンをクリックし、たとえば **MyTeamProject1** などのチーム プロジェクト名を選択して、[**OK**] をクリックします。
5. ローカル フォルダ セルをクリックして、別の省略記号ボタンを表示します。
6. [**ローカル フォルダ**] の下の省略記号ボタンをクリックし、チーム プロジェクトの収納先として、たとえば C:¥DevProjects¥MyTeamProject1 などの、開発コンピュータ上のローカル フォルダを参照して選択します。
7. [**OK**] をクリックしてから、もう一度 [**OK**] をクリックして、[**ワークスペースの編集**] ダイアログ ボックスを閉じます。
8. 1 つ以上の作業フォルダが変更されたことが Microsoft Visual Studio メッセージ ボックスで通知されたら、それに対する応答として、[**OK**] をクリックします。
9. [**閉じる**] をクリックして、[**ワークスペースの管理**] ダイアログ ボックスを閉じます。

チーム プロジェクトに対して取得操作を実行するには:

1. [**チーム エクスプローラ**] で、たとえば **MyTeamProject1** などのチーム プロジェクト ノードを展開します。

2. チーム プロジェクトの下の [**ソース管理**] をダブルクリックします。

3. [**ソース管理エクスプローラ**] で、ルート フォルダ **MyTeamProject1** を右クリックし、[**最新のバージョンを取得**] をクリックします。

4. [**フォルダの参照**] ダイアログ ボックスで、たとえば C:¥DevProjects¥MyTeamProject1 などのローカル パスを選択し、[**OK**] をクリックします。これで、Team Foundation Server 内のチーム プロジェクトのルート フォルダは、コンピュータ上のローカル パスにマップされます。

ステップ 2 - Main フォルダの作成

チーム プロジェクト内で、**Main** という名前の新しいルート フォルダを作成します。ソースおよびその他のプロジェクト成果物用のルート レベル コンテナとして、**Main** フォルダを使用します。また、**Main** フォルダを使用して、複数のプロジェクトにまたがる任意の Visual Studio ソリューション (.sln) ファイルを保管することもできます。ソリューション ファイルはまた、ツリー構造下部にある独自のアプリケーション フォルダ内でも保持されます。

/Main

Main フォルダを作成するには:

1. [チーム エクスプローラ] から、チーム プロジェクトを展開し、[**ソース管理**] をダブルクリックします。

2. [ソース管理エクスプローラ] 内で、チーム プロジェクトのルート フォルダを選択します。

3. 右側のペイン内を右クリックしてから、[**フォルダの新規作成**] をクリックします。

4. **Main** と入力し、ENTER を押します。

ステップ 3 - プロジェクト成果物のフォルダの作成

メイン ソース コードとそれに付随するビルド、ドキュメント、スクリプト ファイル、およびテストなどの成果物を入れておくフォルダを **Main** の下に作成します。

/Main

/Build

/Docs

/Scripts

/Source

/Tests

プロジェクト資産用のフォルダを作成するには:

1. チーム プロジェクトを展開し、[**ソース管理エクスプローラ**] の左側のペイン内の **Main** フォルダを選択します。
2. 右側のペイン内を右クリックし、[**フォルダの新規作成**] をクリックし、**Build** と入力してから、ENTER を押します。
3. ステップ 2 を繰り返して、残りのフォルダを **Main** の下に作成します。

ステップ 4 - ソース ツリーへのソリューションの追加

このステップでは、Visual Studio ソリューション (.sln)、プロジェクト (.vsproj、.vbproj)、およびソース ファイルを、Team Foundation ソース管理内のソース ツリーに追加します。

ソース ツリーにソリューションを追加するには:

1. Visual Studio を使用してソリューションを開きます。
2. [ソリューション エクスプローラ] 内のソリューションを右クリックし、[**ソリューションをソース管理に追加**] をクリックします。
3. ワークスペース マッピングを既に確立済みの場合、この時点でソリューションは追加されます。そうでなければ、チーム プロジェクトおよびターゲット フォルダをソース管理で選択するよう、プロンプトで指示されます。[**ソリューションをソース管理に追加**] ダイアログ ボックスで、[**チーム プロジェクト**] フォルダを展開し、[**Main**] を展開して、[**Source**] を選択します。
4. 正しい**ローカル ワークスペース**が選択されていることを確認します。これは、Team Foundation ソース管理内でフォルダ構造を作成したときに使用したワークスペースと同じでなければなりません。
5. [**OK**] をクリックし、ソース管理にソリューションを追加します。

[ソース管理] フォルダは、以下のようになっているはずです。

/Main

/Source

/MyApp1

/Source

/MyApp1Web

/ClassLibrary1

/Build

/Docs

/Scripts

/Tests

適切なクライアント側フォルダをセットアップする方法の詳細は、以下を参照してください。

- 「[HOWTO]: Team Foundation Server 用に Windows アプリケーションを構造化する方法」
- 「[HOWTO]: Team Foundation Server 用に ASP.NET アプリケーションを構造化する方法」

単体テストの使用

プロジェクトが単体テストを含む場合、メイン ソース フォルダから分離して単体テストを保管するための、次のようなフォルダ構造を検討してみてください。

/Main

/Source

/MyApp1

/Source

/MyApp1Web

/ClassLibrary1

/UnitTests

/Build

/Docs

/Scripts

/Tests

ステップ 5 - 分離のための分岐 Development フォルダの作成に関する考慮事項

機能またはチームをそれぞれ分離したい場合、**Main** フォルダから分岐した **Development** フォルダの作成を検討してみます。分岐を行う前に、保留中の変更をすべてチェックインする必要があることに注意してください。

Development フォルダを分岐するには:

1. ルート レベルの **Development** フォルダ (**Main** フォルダの兄弟として) を作成します。
2. **FeatureBranch1** という名前のサブフォルダを作成します。
3. $/TeamProject/Main/Source フォルダを選択して右クリックし、[**分岐**] をクリックします。

注意: [**分岐**] が淡色表示になっている場合、保留中の変更をすべてチェックインしたかどうかを確認してください。

4. [**分岐**] ダイアログ ボックスで、[**参照**] をクリックして MyTeamProject/Development/FeatureBranch1 を選択してから、[**OK**] をクリックします。
5. [**OK**] をクリックし、分岐を作成します。
6. 保留中の変更をチェックインして、サーバーにプッシュします。

[ソース管理] 内の新しいフォルダ構造は、以下に示したサンプルのようになるはずです。

/Development

/FeatureBranch1

/Source

/MyApp1

/Source

/MyApp1Web

/ClassLibrary1

/FeatureBranch2

/Main

/Build

/Docs

/Scripts

/Source

/Tests

ステップ 6 - 分離の必要なリリース ビルド用の Maintenance フォルダの作成に関する考慮事項

開発の継続中にビルドの保守を実行する必要がある場合、**Maintenance** フォルダの作成を検討します。このフォルダの下に、リリースごとに 1 つずつ、複数のサブ フォルダを作成することができます。

Maintenance フォルダを分岐するには:

1. ルート レベルの **Maintenance** フォルダ (**Main** フォルダおよび **Development** フォルダの兄弟として) を作成します。
2. **Release1** という名前のサブフォルダを作成します。
3. $/TeamProject/Development/Source フォルダを選択して右クリックし、[**分岐**] をクリックします。

注意: [**分岐**] が淡色表示になっている場合、保留中の変更をすべてチェックインしたかどうかを確認してください。

4. [**分岐**] ダイアログ ボックスで、[**参照**] をクリックして $/TeamProject/Maintenance/Release1 を選択してから、[**OK**] をクリックします。
5. [**OK**] をクリックし、分岐を作成します。

[ソース管理] 内の新しいフォルダ構造は、以下に示したサンプルのようになるはずです。

/Maintenance

/Release1

/Source

/MyApp1

/Source

/MyApp1Web

/ClassLibrary1

/Release 1.1

/Release 1.2

その他の考慮事項

- 必要がない限り、分岐を行わないでください。後で分離が必要になったときに、リリースにラベルを付けて分岐することができます。
- 上記のフォルダ構造は、すべてのプロジェクト ファイルを収めた 1 つの Visual Studio ソリューション (.sln) ファイルでプロジェクトが構成されているシナリオに対して理想的に適合します。このシナリオでは、**Main** フォルダにその .sln ファイルが含まれていて、各プロジェクト ファイル (.vsproj、.vbproj) ごとにサブフォルダが 1 つずつあります。上記のサンプルでは、プロジェクト フ

ォルダは MyApp1 や MyApp2 などと表されています。また、上記のアプローチを単一プロジェクトファイルに対して使用することもできます。たとえば、ソリューション ファイルのない単一のプロジェクトを使用する場合などがそれに該当します。

- 複数ソリューションのシナリオの場合、複数の .sln ファイルを **Main** フォルダに入れておくことができます。

参考資料

- 「Team Foundation Server Branching Guidance」(http://www.CodePlex.com/BranchingGuidance)
- ワークスペースの作成に関する詳細は、「方法: ワークスペースを作成する」(http://msdn2.microsoft.com/ja-jp/library/ms181384(VS.80).aspx) を参照してください。

Team Foundation Server のリソース

索引

patterns & practices

patterns and practices の詳細については、http://msdn.microsoft.com/practices/ の Microsoft patterns & practices のホームページを参照してください。

CodePlex サイト

- Team Foundation Branching Guidance (http://www.codeplex.com/BranchingGuidance)
- Guidance Explorer (http://www.codeplex.com/guidanceExplorer)
- Team Foundation Server Guide (http://www.codeplex.com/TFSGuide)
- Team Foundation Server Guidance (http://www.codeplex.com/VSTSGuidance)

Team Foundation についての Web サイト

- Team Foundation Server Team Center (http://msdn2.microsoft.com/en-us/teamsystem/aa718934.aspx)
- Team Foundation Server FAQ (http://msdn2.microsoft.com/en-us/teamsystem/aa718916.aspx)
- Team System Videos and Presentations

(http://msdn2.microsoft.com/en-us/teamsystem/aa718837.aspx)

- Team Foundation Server MSDN Documentation
  (http://msdn2.microsoft.com/ja-jp/library/ms181232(vs.80).aspx)
- Team Foundation Server Installation Guide のダウンロード
  (http://go.microsoft.com/fwlink/?linkid=40042)

パートナおよびサービス プロバイダ

- パートナ カタログ (http://catalog.vsipmembers.com/catalog/)
- Teamprise のクロス プラットフォーム サポート (http://www.teamprise.com/)
- Conchango の Scrum サポート
  (http://www.conchango.com/Web/Public/Content/Home.aspx)
- ComponentSoftware のソース管理の移行
  (http://www.componentsoftware.com/Products/converter/index.htm)
- Borland の要件管理 (http://www.borland.com/)
- RavenFlow のビジネス プロセス モデリング (http://www.n8systems.com/)

フォーラム

MSDN フォーラムの一覧は、次の URL を参照してください。

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

フォーラム	アドレス
Team Foundation Server – 一般	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=22&SiteID=1
Team Foundation Server – セットアップ	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=68&SiteID=1
Team Foundation Server – 管理	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=477&SiteID=1
Team Foundation Server – ビルド	http://forums.microsoft.com/MSDN/ShowForum.a

の自動化	spx?ForumID=481&SiteID=1
Team Foundation Server – Power Toy	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=930&SiteID=1
Team Foundation Server – プロセス テンプレート	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=482&SiteID=1
Team Foundation Server – レポーティング & ウェアハウス	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=480&SiteID=1
Team Foundation Server – Team System の Web アクセス	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=1466&SiteID=1
Team Foundation Server - バージョン管理	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=478&SiteID=1
Team Foundation Server – 作業項目の追跡	http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=479&SiteID=1

Microsoft Team Foundation のブログ

- 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 ブログ (http://blogs.msdn.com/team_foundation)

サービス パック

- 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 for 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)

トレーニング

トレーニング カンパニーの一覧は次の URL を参照してください。

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)