Разработка приложений для облака на платформе, 2-й выпуск

patterns & practices

Обзор: Это вторая книга в серии patterns & practices, посвящённой платформе Windows Azure. В книге рассматривают различные аспекты (выделенная или мультитенантная архитектура, использование CDN, модульное тестирование приложение и mock-объекты, слой доступа к данным, алгоритм MapReduce, интеграция LiveID и другими провайдерами с и т.п.) разработки облачных приложений и сервисов на платформе Windows Azure.

Категория: Справочник\руководство
Аудитория: Windows Azure
Источник: patterns & practices
Дата публикации электронной книги: Сентябрь 2012

Содержание

Введение ... 5
Общие сведения ... 6
Сообщество ... 8
Дополнительные руководства ... 8
Обратная связь и поддержка ... 8
Авторы и соавторы ... 9
Связанные материалы ... 9
Предисловие: Бхарат Шьям (Bharat Shyam) ... 9
Выражение благодарности ... 10
Глава 1. Перед началом работы ... 13
О Windows Azure ... 13
Службы и компоненты Windows Azure ... 13
Разработка приложений для Windows Azure ... 18
Обновление приложений Windows Azure ... 19
Управление, мониторинг и отладка приложений Windows Azure ... 20
Управление базами данных SQL Azure ... 21
Подписка на Windows Azure и модель выставления счетов ... 21
Оценка затрат ... 23
Дополнительные сведения ... 24
Глава 2. Сценарий Tailspin ... 25
Компания Tailspin ... 25
Архитектура приложения Surveys ... 28
Глава 3. Размещение мультитенантных приложений в Windows Azure ... 29
Выделенная (single-tenant) и мультитенантная (multi-tenant) архитектуры ... 30
Мультитенантная архитектура в Azure ... 32
Выбор выделенной или мультитенантной архитектуры ... 33
Глава 4. Доступ к приложению Surveys ... 42
DNS-имена, сертификаты и SSL в приложении Surveys ... 42
Географическое местоположение ... 47
Проверка подлинности и авторизация ... 49
Дополнительные сведения ... 62
Глава 5. Создание масштабируемого, мультитенантного приложения для Windows Azure ... 63
Партиционирование ... 63

Провиженинг пробной учетной записи и новых клиентов 67
Выставление счетов клиентам 71
Настройка пользовательского интерфейса 72
Масштабирование приложений с помощью рабочих ролей 73
Масштабирование приложения Surveys 81
Глава 6. Работа с данными в приложении Surveys 94
Тестирование хранилища Windows Azure 100
Сохранение данных ответов на опрос 104
Отображение данных 111
Использование SQL Azure 126

Разработка приложений для облака, 2-й выпуск

Введение

Каким образом компания может создать приложение, которое имеет действительно глобальный охват и способно быстро масштабироваться при внезапном росте количества пользователей? Раньше компаниям приходилось вкладывать средства в создание собственной инфраструктуры, способной поддерживать такое приложение, и, как правило, ресурсы для такого рискованного предприятия имелись в наличии только у крупных компаний. Создание такой инфраструктуры и управление ею недешево, особенно потому, что необходимо рассчитывать на пиковые нагрузки, а это часто означает длительные простои большей части мощностей. Облако изменило правила игры: теперь вы пользуетесь инфраструктурой по мере необходимости, платите за нее по мере использования и можете создавать масштабируемые, глобальные приложения, доступные как крупным, так и небольшим компаниям.

Облачная платформа обеспечивает доступ к ресурсам по запросу, сохранение работоспособности при сбоях, распределенные вычисления, центры обработки данных в разных частях мира и возможность интеграции с другими платформами. За поддержку всей инфраструктуры и управление ею отвечает кто-то другой (поставщик), а вы платите только за ресурсы, используемые за каждый расчетный период. Вы можете сосредоточиться на основной задаче — создании приложения и его развертывании в одном или нескольких центрах обработки данных, ближайших к пользователям этого приложения. Затем вы можете отслеживать свои приложения, увеличивая и уменьшая загрузку по мере необходимости.

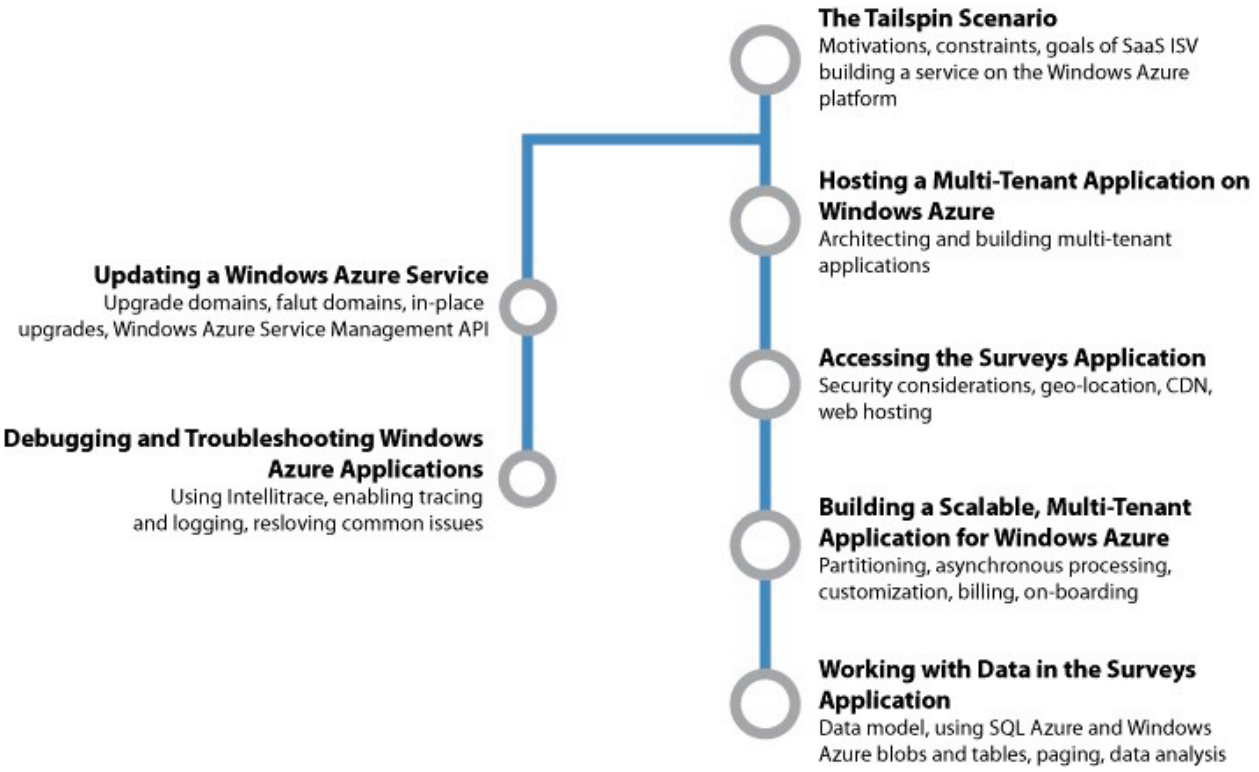
Действительно, если приложения переносятся в облако, вы теряете некоторую долю контроля и автономности, но выигрываете от снижения затрат, повышения гибкости и масштабируемости вычислительных мощностей и хранилищ. В этом руководстве рассказывается, как это сделать.

Загрузки	Пример кода
Учебное содержимое	Практическое занятие
Печатная и электронная книга	Эта книга скоро будет доступна в печатном и электронном виде.
Сообщество	http://wag.codeplex.com

Общие сведения

Эта книга является вторым томом в серии, посвященной технологической платформе Windows Azure. Том 1, Миграция приложений в Windows Azure, содержит общие сведения о платформе Windows Azure. В этом томе рассматривается модель стоимости и управление жизненным циклом облачных приложений, а также описан способ переноса существующего приложения ASP.NET в облако. В этой книге рассматривается создание с нуля многопользовательского приложения, работающего по модели «программного обеспечения как услуги» (SaaS), для эксплуатации в облаке с использованием для этого последних версий инструментов Windows Azure и новейших функций платформы Windows Azure. Эта книга предназначена для любого архитектора, разработчика или специалиста по информационным технологиям (ИТ), который осуществляет проектирование, создание и эксплуатацию приложений и служб, работающих в облаке или взаимодействующих с ним. Хотя для работы в Windows Azure приложение не обязательно должно основываться на операционной системе Microsoft® Windows®, эта книга предназначена для специалистов, работающих с операционными системами Windows. Необходимо иметь представление о платформе Microsoft.NET Framework, среде разработки Microsoft Visual Studio®, ASP.NET MVC и средстве разработки Microsoft Visual C#®.

Общие сценарии



«Сценарий Tailspin» знакомит вас с компанией Tailspin и приложением Surveys. Здесь представлены общие сведения об архитектуре приложения Surveys, а в следующих главах содержатся дополнительные сведения о том, как компания Tailspin разработала и реализовала приложение Surveys для облака. Изучение этой главы поможет понять бизнес-модель компании Tailspin, ее стратегию внедрения облачной платформы и некоторые связанные с этим проблемы.

В главе **«Размещение мультитенатного приложения в Windows Azure»** рассматриваются некоторые вопросы, связанные с архитектурой и созданием **мультитенатных** приложений для работы в Windows Azure. Описываются преимущества **мультитенатной** архитектуры и компромиссы, на которые приходится идти. В этой главе предоставляется концептуальная основа, которая поможет читателю уяснить некоторые вопросы, рассматриваемые более подробно в последующих главах.

Глава **«Доступ к приложению Surveys»** посвящена некоторым проблемам, с которыми столкнулись разработчики компании Tailspin при разработке и реализации некоторых компонентов приложения, взаимодействующих с клиентами. В разделы входят выбор URL-адресов для доступа к приложению Surveys, безопасность, размещение приложения в разных географических регионах и использование сети доставки содержимого (CDN) для кэширования содержимого.

В главе **«Создание масштабируемого, мультитенатного приложения для Windows Azure»** рассматривается, каким образом компания Tailspin обеспечила масштабируемость **мультитенатного** приложения Surveys. А также описывается, как секционируется приложение, как оно использует рабочие роли и поддерживает адаптацию, настройку и выставление счетов для клиентов приложения.

В главе **«Работа с данными в приложении Surveys»** показано как приложение использует данные. Эта глава начинается с описания того, как приложение Surveys хранит данные в таблицах Windows Azure и больших двоичных объектах (BLOB), а также каким образом разработчики компании Tailspin проектировали тестируемые классы хранилищ. В этой главе также рассматривается решение некоторых конкретных проблем, возникших перед компанией Tailspin, связанных с данными, включая партиционирование данных и реализацию состояния сеанса. И наконец, в этой главе показана роль, которую играет технологическая платформа SQL Azure в приложении Surveys.

В главе **«Обновление службы Windows Azure»** описываются варианты обновления приложения Windows Azure, в ней также рассказывается, как можно обновить приложения, не прерывая его работу.

В главе **«Отладка и приложений Windows Azure и устранение неполадок»** описываются некоторые, характерные для приложений Windows Azure методы, позволяющие обнаруживать и разрешать проблемы при построении, развертывании и использовании приложений Windows Azure. Кроме того, в ней описано использование средств диагностики Windows Azure и программы Microsoft IntelliTrace с приложениями, развернутыми в Windows Azure.

Требования к аудитории

Эта книга предназначена для любого архитектора, разработчика или специалиста по информационным технологиям (ИТ), который осуществляет проектирование, создание и эксплуатацию приложений и служб, работающих в облаке или взаимодействующих с ним. Хотя для работы в Windows Azure приложение не обязательно должно основываться на операционной системе Microsoft® Windows®, эта книга предназначена для специалистов, работающих с операционными системами Windows. Необходимо иметь представление о платформе Microsoft.NET Framework, среде разработки Microsoft Visual Studio®, ASP.NET MVC и средстве разработки Microsoft Visual C#.

Требования к системе

Требования к системе для запуска сценариев:

- Microsoft Windows 7 с пакетом обновления 1 (SP1) или Windows Server 2008 R2 с пакетом обновления 1 (SP1) (32-разрядная и 64-разрядная версии)
- Microsoft .NET Framework версии 4.0
- Microsoft Visual Studio 2010, выпуск Ultimate, Premium или Professional с пакетом обновления 1 (SP1)
- Пакет SDK для Windows Azure для .NET (включает средства Visual Studio для Windows Azure)
- Microsoft SQL Server или SQL Server Express 2008
- MVC 3 Framework
- Windows Identity Foundation. Требуется для авторизации на основе утверждений.
- WebAii. Поместите в папку примеров «Lib».
- Moq версии 4.0.10501.6. Откройте диалоговое окно «Свойства» и разблокируйте zip-файл после его загрузки и до извлечения содержимого. Поместите содержимое в папку примеров «Lib».
- Microsoft Anti-Cross Site Scripting Library V4. Поместите в папку примеров «Lib».
- Enterprise Library 5 (необходимые сборки включены в папки с исходным кодом)
- Unity 2.0 (необходимые сборки включены в папки с исходным кодом)

Сообщество

Данное руководство, подобно многим материалам Pattern&Practice, связано с сайтом сообщества. Здесь можно задать вопросы, высказать свое мнение, пообщаться с другими пользователями и обменяться идеями. Члены сообщества могут оказывать помощь в планировании и тестировании будущих руководств, а также загружать расширения, учебные пособия и другие дополнительные материалы.

Дополнительные руководства

Для этого сценария было разработано расширение для мобильных пользователей, использующих устройства Windows Phone 7. Дополнительные сведения см. на веб-сайте http://msdn.microsoft.com/ruru/library/gg490765.aspx.

В третьей части планируется рассмотреть сценарии интеграции и новые возможности. Дополнительные сведения см. на веб-сайте http://wag.codeplex.com.

Обратная связь и поддержка

Есть вопросы? Комментарии? Предложения? Если вы хотите оставить свои комментарии об этом руководстве или вам требуется помощь в решении какой-либо проблемы, посетите сайт сообщества Windows Azure. Доска сообщений на сайте сообщества — наилучший канал обратной связи и поддержки, поскольку с ее помощью можно обмениваться идеями, задавать вопросы и описывать решения для всего сообщества.
Это содержимое предоставляется только для примера, его можно повторно использовать, изменять и расширять. Это не продукт MicrosoftMicrosoft. Примеры кода поставляются «как есть» и без каких-либо гарантий. Клиенты могут за плату получить поддержку в службе технической поддержки Microsoft.

Авторы и соавторы

Это руководство было создано следующими авторами:

- Управление программами и продуктами: Масаси Нарумото (Masashi Narumoto)
- Архитектор: Эудженио Паче (Eugenio Pace)
- Эксперты по конкретным вопросам: Доминик Беттс (Dominic Betts, Content Master), Скотт Денсмор (Scott Densmore), Райан Данн (Ryan Dunn), Стив Маркс (Steve Marx) и Матиас Волоски (Matias Woloski, Southworks).
- Разработка: Скотт Денсмор (Scott Densmore), Федерико Боерр (Federico Boerr, Southworks), Скотт Денсмор (Scott Densmore) и Адриан Менегатти (Adrián Menegatti, Southworks).
- Группа тестирования: Ханц Занг (Hanz Zhang), Равиндра Махендраварман (Ravindra Mahendravarman, Infosys Ltd.) и Рати Велусами (Rathi Velusamy, Infosys Ltd.).
- Группа редактирования: Доминик Беттс (Dominic Betts, Content Master), Роан Корбисье (Roann Corbisier), Алекс Гомер (Alex Homer) и Тина Берден (Tina Burden, TinaTech Inc.).
- Дизайн и иллюстрации книги: Эллен Форни (Ellen Forney), Джон Хаббард (John Hubbard, eson), Кети Наймер (Katie Niemer) и Эудженио Паче (Eugenio Pace).
- Управление выпуском Ричард Берт (Richard Burte, ChannelCatalyst.com, Inc.).

Хотим поблагодарить клиентов, участников и членов сообщества, которые внимательно знакомились с предварительными выпусками. Среди них мы бы хотели отметить исключительный вклад Дэвида Айкена (David Aiken), Грэхема Астора (Graham Astor, Avanade), Эдварда Баккера (Edward Bakker, Inter Access), Вивека Бхатнагара (Vivek Bhatnagar, Microsoft), Патрика Батлера Монтерде (Patrick Butler Monterde, Microsoft), Шайя Коэна (Shy Cohen), Джеймса Конарда (James Conard), Брайана Дэвиса (Brian Davis, Longscale), Ашиша Дхэмхера (Aashish Dhamdhere, Windows Azure, Microsoft), Андреаса Эрбена (Andreas Erben, DAENET), Гайла Фрита (Giles Frith), Эрика Л. Голпа (Eric L. Golpe, Microsoft), Джонни Халифа (Johnny Halife, Southworks), Алекса Гомера (Alex Homer), Саймона Инка (Simon Ince), Джоши Джозефа (Joshy Joseph), Эндрю Кимбэла (Andrew Kimball), Милинду Котлавеле (Milinda Kotelawele, Longscale), Марка Коттке (Mark Kottke, Microsoft), Криса Лаундса (Chris Lowndes, Avanade), Диану О'Брайен (Dianne O'Brien, Windows Azure, Microsoft), Штеффена Ворейна (Steffen Vorein, Avanade), Брэда Уилсона (Brad Wilson, ASP.NET Team) и Майкла Вуда (Michael Wood, Strategic Data Systems).

Связанные материалы

- Миграция приложений в облако, 2-й выпуск
- Руководство по управлению доступом и удостоверениями на основе утверждений, 2-й выпуск

Предисловие: Бхарат Шьям (Bharat Shyam)

Windows Azure является масштабируемым, надежным и экономически выгодным решением для развертывания приложений и служб — для организаций и предприятий любого размера, работающих на рынках любых типов. Моя задача в корпорации Microsoft заключается в том, чтобы разработчики могли помогать этим организациям и предприятиям достигать своих целей, пользуясь мощью, гибкостью и доступностью служб Windows Azure.

Разработчикам, которые знакомы с платформой Microsoft, а также с такими инструментами и средами разработки, как .NET и Visual Studio, не составит труда начать создавать приложения на платформе Windows Azure. Однако, чтобы добиться больших успехов в этой привлекательной новой области, им необходимо вникнуть в такие концепции, как мультитенантная модель, партиционирование данных и надежные механизмы хранения, используемые в Windows Azure и SQL Azure.

Наша команда сделала все возможное для создания широкого спектра средств, с помощью которых разработчики могут создавать приложения и службы Windows Azure. Теперь же, с выходом этого руководства, группа Pattern&Practice расширяет опыт разработчиков, предоставляя им сведения, необходимые для создания высокопроизводительных, готовых к работе в реальных условиях, приложений на платформе Windows Azure.

Руководство «Разработка приложений для облака» является вторым в серии руководств Pattern&Practice, в которых рассматриваются варианты создания облачных приложений. В первом руководстве этой серии рассказывалось каким образом разработчики могут мигрировать свои приложения в облако. В этом руководстве рассматриваются вопросы разработки новых приложений для облака. В его основе опыт работы вымышленной организации под названием Tailspin, которая создает масштабируемое мультитенантная приложение для управления опросами. В руководстве рассматриваются основные понятия и принципы разработки приложений платформе Windows Azure.

Что касается других руководств этой серии, в них используется практический подход, на примере которого разъясняется определенная ситуация с приложением, вопросы проектирования и разработки. В этом руководстве основное внимание уделено принципам проектирования и реализации кода, интеграции со средой выполнения, а также использованию хранилища и фоновых рабочих задач. В нем приведены сведения об отладке и устранении неисправностей, а также пример полноценного приложения, который разработчики могут загрузить и использовать. На его примере можно лучше понять реализацию и узнать как применять полученный набор знаний при проектировании и построении собственных приложений.

Я горжусь проделанной нами работой, цель которой заключается в том, чтобы поделиться с разработчиками опытом использования Windows Azure. Мне очень нравится то, что группа Pattern&Practice делает, занимаясь созданием этой серии руководств. Вместе мы можем помочь разработчикам реализовать свой потенциал, позволить им быть продуктивными и успешными в работе с платформой Windows Azure.

Бхарат Шьям (Bharat Shyam), генеральный менеджер платформы разработчика Windows Azure, корпорация Microsoft.

Выражение благодарности

4 марта я увидел в папке «Входящие» письмо от нашего генерального директора Стива Болмера. Обычно я нечасто получаю от него электронные письма, поэтому уделил этому письму особое внимание. В строке темы значилось «Это касается всех». В письме подводились итоги стремления компании Microsoft® к переходу на облачные вычисления. Если мне и было нужно еще одно подтверждение того, что я уже знал о серьезности подхода компании Microsoft к облаку, это письмо стало таким подтверждением.
Мое первое знакомство с платформой, которая теперь носит название Windows Azure произошло три года назад. Я работал в подразделении Developer & Platform Evangelism (DPE), и мои обязанности заключались в изучении SaaS (Software as Service) направления. Возможно, кто-то из вас даже помнит мой очень ранний пример, разработанный в конце 2007 года, который назывался «Northwind Hosting». Пример демонстрировал многие возможности, которые платформа Windows Azure предлагает сегодня. Видеть, как программа, в которой я участвовал с первого дня, становится реальной, было огромным счастьем.

В феврале 2009 года я ушел из подразделения Developer & Platform Evangelism (DPE) и присоединился к подразделению Patterns & Practices. Я должен был возглавить «облачную программу» — группу проектов по изучению задач проектирования, связанных с созданием приложений для облака. Когда платформа Windows Azure была анонсирована, спрос на руководства в этой области резко возрос.

В ходе рассмотрения различных сценариев разработки стало очевидно, что необходимо решить проблемы управления идентификацией и доступом, прежде чем переходить к чему-то еще. Это особенно важно, если ваша компания вложила большие средства в собственную систему и вы хотите перенести некоторые из этих ресурсов в облако. Именно так и обстоят дела со многими нашими клиентами.

В декабре 2009 года мы выпустили *Руководство по управлению доступом и удостоверениями на основе утверждений*. Это было первым результатом работы подразделения Patterns & Practices и важной вехой в нашей облачной программе. За этим руководством последовало руководство Миграция приложений в облако, которое стало первым из трех руководств по разработке приложений в Windows Azure.

Windows Azure — это особая платформа во многих отношениях. Одна из ее характеристик — скорость нововведений. Различные группы, которые поставляют системы на основе этой платформы, доказали, что могут быстро предоставлять новые возможности конечным пользователям. Чтобы не отставать, мы должны были очень быстро разрабатывать содержимое. Мы решили выполнять наши проекты интенсивными двухмесячными этапами, на каждом из которых должны рассматриваться конкретные вопросы.

В этом руководстве рассматривается сценарий Greenfield: проектирование и разработка новых приложений для платформы Windows Azure. Это продолжение предыдущего руководства, посвященного миграции существующего приложения на платформу Windows Azure.

Прежде всего я хочу поблагодарить следующих специалистов и тех, кто участвовал в создании этого руководства. Это Доминик Беттс (Dominic Betts), Скотт Денсмор (Scott Densmore), Райан Данн (Ryan Dunn), Стив Маркс (Steve Marx) и Матиас Волоски (Matias Woloski). Доминик обладает необыкновенным знанием предмета в деталях и умеет объяснить это всем остальным точно, полно и понятно. Скотт хорошо знает, как создавать масштабируемые приложения Windows Azure. Именно этим он занимался до того, как присоединился к моей команде. Он также имеет многолетний опыт создания платформ и инструментов для разработчиков. Я имел честь работать с Райаном в предыдущих проектах и всегда удивлялся его острому восприятию, проницательности и опыту. Как специалист по стандартам Windows Azure, он смог объяснить нам, что потребуется клиентам с реальными требованиями. Стив — технический стратег по платформе Windows Azure. Он сыграл важную роль в создании этого руководства. Он демонстрирует нам не только сегодняшние возможности платформы, но и то, как она будет развиваться в дальнейшем. Это важно, потому что сейчас мы хотим дать рекомендации, соответствующие долгосрочным целям. И наконец, Матиас, последний по списку, но не по значимости, является опытным участником многих проектов наряду со мной. Он работал над Windows Azure с самого первого дня, и его вклад в создание этого руководства неоценим.

Как всегда в нашей документации, в большинстве глав есть примеры кода. Они иллюстрируют то, о чем говорится в руководстве. Большое спасибо группам разработки и тестирования проекта за разумное соотношение между техническим качеством, расставленными акцентами и простотой понимания кода. Это Масаси Нарумото (Masashi Narumoto), Скотт Денсмор (Scott Densmore), Федерико Боерр (Federico Boerr, Southworks), Адриан Менегатти (Adrián Menegatti, Southworks), Ханц Занг (Hanz Zhang), Равиндра Махендраварман (Ravindra Mahendravarman, Infosys Ltd.) и Рати Велусами (Rathi Velusamy, Infosys Ltd.).

Наши руководства должны быть не только технически точными, но занимательными и интересными для чтения. Это непростая задача, и я хочу поблагодарить за это Доминика Беттса (Dominic Betts), Роанн Корбизье (RoAnn Corbisier), Алекса Гомера (Alex Homer) и Тину Берден (Tina Burden) из группы создания и редактирования.

Концепция визуального представления этого руководства была первоначально разработана Робертой Лейбович (Roberta Leibovitz) и Колином Кэмпбелом (Colin Campbell) из компании Modeled Computation LLC *для Руководства по управлению доступом и удостоверениями на основе утверждений.* После блестящих отзывов мы решили использовать эту концепцию и для этой книги. Над оформлением книги работал Джон Хаббард (John Hubbard, eson). Мультипликационные персонажи нарисованы знаменитым карикатуристом Эллен Форни (Ellen Forney). Она проживает в г. Сиэтл, и ее работы отмечены наградами. Технические иллюстрации переработаны Робом Нэнсом (Rob Nance) и Кэти Нимер (Katie Niemer) из моих макетов, выполненных на планшетном ПК.

Все наши руководства рецензируются, комментируются, тщательно изучаются и критикуются многочисленными клиентами, участниками и коллегами. Мы также получили отзывы от многочисленного сообщества через наш веб-сайт CodePlex. Платформа Windows Azure обширна и охватывает множество отраслей. Нам очень повезло, что мы могли использовать интеллектуальный потенциал чрезвычайно разноплановой и опытной группы наших читателей.

Я также хочу поблагодарить всех тех, кто добровольно потратил свое время и знания на оценку наших черновиков и ранних вариантов содержимого. Среди них я хотел бы отметить исключительный вклад Дэвида Айкена (David Aiken), Грэхема Астора (Graham Astor, Avanade), Эдварда Баккера (Edward Bakker, Inter Access), Вивека Бхатнагара (Vivek Bhatnagar), Патрика Батлера Монтерде (Patrick Butler Monterde, Microsoft), Шайя Коэна (Shy Cohen), Джеймса Конарда (James Conard), Брайана Дэвиса (Brian Davis, Longscale), Ашиша Дхэмхера (Aashish Dhamdhere, Windows Azure, Microsoft), Андреаса Эрбена (Andreas Erben, DAENET), Гайла Фрита (Giles Frith), Эрика Л. Голпа (Eric L. Golpe, Microsoft), Джонни Халифа (Johnny Halife, Southworks), Алекса Гомера (Alex Homer), Саймона Инка (Simon Ince), Джоши Джозефа (Joshy Joseph), Эндрю Кимбэла (Andrew Kimball), Милинду Котлавеле (Milinda Kotelawele, Longscale), Марка Коттке (Mark Kottke, Microsoft), Криса Лаундса (Chris Lowndes, Avanade), Диану О'Брайен (Dianne O'Brien, Windows Azure, Microsoft), Штеффена Ворейна (Steffen Vorein, Avanade) и Майкла Вуда (Michael Wood, Strategic Data Systems). Надеюсь, это руководство окажется вам полезным!

Эухиньйо Пче (Eugenio Pace)

Старший руководитель проекта — *подразделение Patterns & Practices*

Корпорация Microsoft

Глава 1. Перед началом работы

В этой главе приведено краткое описание технологической платформы Microsoft Windows Azure, предоставляемых ей служб, которые она обеспечивает. Под *облаком* имеется в виду набор взаимосвязанных вычислительных ресурсов, размещенных в одном или нескольких центрах обработки данных. В этой главе также приведены ссылки, по которым можно найти дополнительные сведения о функциях Windows Azure, методах работы и технологиях, используемых в этой серии руководств, а также прилагаемые к ним примеры кода.

О Windows Azure

В облаке разработчики могут развертывать и выполнять приложения, а также хранить данные. При этом локальные приложения могут использовать облачные ресурсы. Например, приложение, размещенное на локальном сервере, полнофункциональный клиент, работающий на настольном компьютере или мобильном устройстве могут использовать хранилище, расположенное в облаке.

Платформа Windows Azure делает аппаратные ресурсы абстрактными, виртуализуя их. Каждое развернутое в Windows Azure приложение выполняется на одной или нескольких виртуальных машинах. Развернутые приложения работают так, как будто они размещены на выделенном компьютере, хотя могут совместно с другими виртуальными машинами на том же физическом узле использовать физические ресурсы — место на диске, сетевой ввод-вывод, ядра ЦП и виртуальные машины на одном физическом сервере. Ключевыми преимуществами уровня абстракции поверх физического оборудования являются портативность и масштабируемость. Виртуализация служб позволяет перемещать их на любое число узлов в центре обработки данных. Используя виртуальные технологии, неспециализированное оборудование, мультитенантность и агрегирование потребностей, Microsoft достигает экономии за счет масштаба. Это приводит к более полному использованию ресурсов центра обработки данных (и, соответственно, к большей выработке на доллар затрат на оборудование) и соответствующей экономии, которая отражается и на ваших затратах.

Комментарий Бхарата:

Windows Azure позволяет обеспечить возможность переноса и масштабируемости приложений, а также сократить затраты на эксплуатацию и общую стоимость владения.

Службы и компоненты Windows Azure

Ряд служб и компонентов Windows Azure, Windows Azure Building Blocks и платформа SQL Azure предъявляют к приложениям специфические требования. При подписке на Windows Azure можно выбрать нужные компоненты и платить только за их использование. Добавить компоненты в подписку и удалить их оттуда можно в любой момент. Механизм выставления счетов за каждую службу зависит от типа компонента, предоставляющего службу. Дополнительные сведения о модели выставления счетов см. ниже в этой главе в разделе «*Подписка на Windows Azure и модель выставления счетов*».

По мере развития платформы Windows Azure службы и компоненты продолжают изменяться. Данная серия руководств с прилагаемыми образцами кода и сопутствующими практическими занятиями показывает многие компоненты и службы, доступные в Windows Azure и SQL Azure. Следующие четыре раздела этой главы кратко описывают основные службы и компоненты, доступные на момент написания. Они разделены на четыре категории: среда исполнения, управление данными, сетевые службы и прочие службы.

Комментарий Бхарата:

Windows Azure содержит набор служб, позволяющих упростить разработку, повысить надежность и облегчить управление размещенными в облаке приложениями.

Дополнительные сведения о службах и компонентах Windows Azure см. в разделе «*Компоненты Windows Azure*» на портале Windows Azure по адресу http://www.microsoft.com/windowsazure/features/. Конкретные рекомендации по каждому компоненту и службе см. в источниках, ссылки на которые приведены в следующих разделах.

Примечание.

Для использования любого из этих компонентов и служб необходима подписка на Windows Azure. Для регистрации учетной записи Windows Azure необходим действующий идентификатор Windows Live ID. Дополнительные сведения см. в разделе «*Получение платформы Windows Azure*» по адресу http://www.windowsazure.com/ru-ru/pricing/free-trial/.

Среда выполнения

Среда выполнения Windows Azure состоит из платформы для приложений и служб, размещенных в одной или нескольких ролях. Типы ролей, которые могут быть реализованы в Windows Azure:

- Azure Compute (веб-роли и рабочие роли). Приложение Windows Azure состоит из одной или нескольких ролей, размещенных в центрах обработки данных Azure. Как правило, существует хотя бы одна веб-роль, предоставленная для доступа пользователям приложения. Приложение может содержать дополнительные роли, например рабочие, которые обычно используются для фоновой обработки и выполнения вспомогательных задач для веб-ролей. Дополнительные сведения см. в разделах «*Обзор создания размещенной службы для Windows Azure*» по адресу http://technet.microsoft.com/ru-ru/library/gg432976.aspx и «*Создание приложения для работы в размещенной службе*» по адресу http://technet.microsoft.com/ru-ru/library/hh180152.aspx.
- Виртуальная машина (роль виртуальной машины). Эта роль позволяет разместить собственный пользовательский экземпляр операционной системы Windows Server 2008 R2 Enterprise или Windows Server 2008 R2 Standard в центре обработки данных Windows Azure. Дополнительные сведения см. в разделе «*Создание приложений с помощью роли виртуальной машины в Windows Azure*» по адресу http://technet.microsoft.com/ru-ru/library/gg465398.aspx.

Большинство примеров в этом руководстве и связанном с ним руководстве «*Разработка приложений для облака*» (см. http://wag.codeplex.com/), а также примеров в Hands-on-Labs используют веб-роль для выполнения необходимой обработки.

Использование рабочей роли также описывается и показано во многих местах в руководствах и примерах. Сюда входит глава 5 этого руководства и соответствующий образец приложения, упражнение 3 из практических упражнений для этого руководства, глава 6 руководства «Миграция приложений в облако» (или http://wag.codeplex.com/) и соответствующий пример приложения, а также упражнение 4 практических упражнений для руководства «Миграция приложений в облако».

Управление данными

Windows Azure, SQL Azure и связанные службы позволяют хранить данные и управлять ими несколькими разными способами. Доступны следующие службы и функции управления данными:

- Хранилище Azure. Включает четыре основные службы для постоянного и длительного хранения данных в облаке. Службы поддерживают интерфейс REST, доступный из приложений, размещенных в Azure, а также локальных приложений. Сведения о REST API см. в «*Справочнике по интерфейсу REST API служб хранилища Windows Azure*» по адресу http://msdn.microsoft.com/ruru/library/dd179355.aspx. Существует четыре службы хранения:

    - Служба таблиц Azure обеспечивает механизм хранения данных в табличном формате, то есть в виде привычных строк и столбцов, и поддерживает запросы на управление данными. Эта служба предназначена в основном для тех сценариев, где необходимо хранить большие объемы данных с простым доступом и обновлением. Дополнительные сведения см. в разделах «*Концепции табличных служб*» по адресу http://msdn.microsoft.com/ru-ru/library/dd179463.aspx и «*API табличных служб*» по адресу http://msdn.microsoft.com/ru-ru/library/dd179423.aspx.

    - Служба больших двоичных объектов (BLOB) реализует последовательность контейнеров для хранения текста или двоичных данных. Обеспечиваются как блочные контейнеры больших двоичных объектов для потоковой передачи данных, так и страничные контейнеры больших двоичных объектов для выполнения произвольного чтения и записи. Дополнительные сведения см. в разделах «*Сведения о блочных и страничных больших двоичных объектах*» по адресу http://msdn.microsoft.com/ru-ru/library/ee691964.aspx и «*API службы больших двоичных объектов*» по адресу http://msdn.microsoft.com/ruru/library/dd135733.aspx.

    - Служба очередей реализует механизм для постоянного и надежного обмена сообщениями между экземплярами ролей, например между веб-ролью и рабочей ролью. Дополнительные сведения см. в разделах «*Концепции службы очередей*» по адресу http://msdn.microsoft.com/ru-ru/library/dd179353.aspx и «*API службы очередей*» по адресу http://msdn.microsoft.com/ru-ru/library/dd179363.aspx.

    - Диски Windows Azure реализуют механизм, который позволяет приложению подключить однотомный виртуальный жесткий диск NTFS в виде страничного большого двоичного объекта, а также выгружать и загружать виртуальные жесткие диски через такой объект. Дополнительные сведения см. в разделе «*Диск Windows Azure*» (PDF-файл) по адресу http://go.microsoft.com/?linkid=9710117.

- База данных SQL Azure. Это масштабируемая служба облачной базы данных с высокой степенью доступности, построенная на основе технологий SQL Server, поддерживает знакомую реляционную модель баз данных, совместимую с T-SQL. Она может использоваться с приложениями, размещенными в Windows Azure, и другими приложениями, размещенными локально у пользователей или на других площадках. Дополнительные сведения см. в разделе «*База данных SQL Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/ee336279.aspx.

- Синхронизация данных. Синхронизация данных SQL Azure — это облачная служба синхронизации данных, построенная на основе технологий Microsoft Sync Framework. Она обеспечивает двунаправленную синхронизацию данных и управление данными, что дает возможность работать с несколькими базами данных SQL Azure, а также совместно использовать данные в базе данных пользователя и базе данных SQL Azure. Дополнительные сведения см. в «*Центре разработки Microsoft Sync Framework*» по адресу http://msdn.microsoft.com/ru-ru/sync.

- Кэширование. Эта служба позволяет создавать высокопроизводительные приложения, использующие распределенный кэш данных в памяти с малой задержкой и высокой пропускной способностью. Эта служба не требует установки или управления и автоматически увеличивает и уменьшает размер кэша в динамическом режиме по мере необходимости. Она может использоваться для кэширования данных приложения и сведений о состоянии сеанса ASP.NET, а также для кэширования страничного вывода ASP.NET. Дополнительные сведения см. в разделе «*Служба кэширования (Windows Azure AppFabric)*» по адресу http://msdn.microsoft.com/ruru/library/gg278356.aspx.

В главах 3 и 5 этого руководства описываются основные понятия и реализация архитектур обслуживания одним экземпляром нескольких развертываний для хранения данных.

В главе 5 этого руководства описывается использование очередей.

В главе 6 этого руководства показано использование хранилища таблиц (включая постраничный просмотр данных) и хранилища больших двоичных объектов в приложениях.

В главе 6 этого руководства описывается использование SQL Azure для хранения данных.

В главе 6 этого руководства демонстрируется использование службы кэширования.

В главе 5 руководства «Миграция приложений в облако» показывается использование хранилища таблиц.

В главе 6 руководства «Миграция приложений в облако» показывается использование хранилища больших двоичных объектов и очередей.

Сетевые службы

Windows Azure включает несколько сетевых служб, которые обеспечивают максимальное повышение производительности, проверку подлинности и повышают управляемость размещенными приложениями. Это следующие службы:

- Сеть доставки содержимого (CDN). Сеть CDN позволяет кэшировать общедоступные статические данные для приложений в специальных точках, расположенных ближе (с точки зрения доставки по сети) к конечным пользователям. Сеть CDN использует ряд центров обработки данных по всему миру, где данные хранятся в хранилище больших двоичных объектов с анонимным доступом. То есть не обязательно там, где фактически работают приложения. Дополнительные сведения см. в разделе «*Доставка широкополосного содержимого по сети доставки содержимого (CDN) Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/ee795176.aspx.

- Служба Virtual Network Connect. Эта служба позволяет настроить роли приложения, работающего в Windows Azure, а также компьютеры в сети конечного пользователя таким образом, чтобы обеспечить видимость их нахождения в одной и той же сети. Эта служба с помощью программного агента, запускаемого на компьютере конечного пользователя, устанавливает соединение с ролями Windows Azure по протоколу IPsec. Поддерживается также возможность прямого администрирования, управления, мониторинга и отладки ролей. Дополнительные сведения см. в разделе «*Подключение локальных компьютеров к ролям Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/gg433122.aspx.

- Диспетчер трафика виртуальной сети. Эта служба позволяет задать перенаправление запросов и балансировку нагрузки с помощью трех разных методов. Как правило, диспетчер трафика служит для повышения производительности путем перенаправления пользовательских запросов к экземпляру в ближайшем центре обработки данных в режиме Performance (Производительность). Альтернативные режимы — Failover (Отработка отказа) и Round Robin (Циклический перебор). Дополнительные сведения см. в разделе «*Диспетчер трафика Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/WAZPlatformTrainingCourse_WindowsAzureTrafficManager.

- Управление доступом. Эта служба удостоверений и управления доступом построена на основе стандартов и использует целый ряд поставщиков удостоверений (IdPs) для проверки подлинности пользователей. Служба управления доступом действует как служба маркеров безопасности (STS), помогая использовать преимущества методов федеративной проверки подлинности, когда подлинность пользователя проверяется не в той области или домене, где находится приложение. В качестве примера можно привести управление доступом пользователей, когда удостоверение пользователя проверяется поставщиком удостоверений (Windows Live ID, Google и т. п.). Дополнительные сведения см. в разделе «*Access Control Service 2.0*» по адресу http://msdn.microsoft.com/ru-ru/library/gg429786.aspx и «*Руководстве по проверке подлинности на основе утверждений и управлению доступом*» в http://claimsid.codeplex.com/.

- Шина интеграции. Обеспечивает надежный обмен сообщениями и возможность потоковой передачи данных для распределенных и гибридных приложений (например, для обмена данными между приложениями, размещенными в Windows Azure, и локальными приложениями и службами конечных пользователей) без брандмауэров и инфраструктуры безопасности. Эта шина может использовать ряд протоколов связи и обмена сообщениями, а также шаблонов, обеспечивающих надежную доставку и обмен сообщениями. Она масштабируется в зависимости от нагрузки и может интегрироваться с артефактами собственного сервера BizTalk Server. Дополнительные сведения см. в разделе «*Шина обслуживания AppFabric*» по адресу http://msdn.microsoft.com/ruru/library/ee732537.aspx.

В главе 4 этого руководства показывается использование сети доставки содержимого (CDN).

Подробные инструкции по использованию службы управления доступом см. в соответствующем «*Руководстве по проверке подлинности на основе утверждений и управлению доступом*» (см. http://claimsid.codeplex.com/) и упражнении 3 Hands-on-Labs для этого руководства.

Другие службы

Windows Azure включает следующие дополнительные службы:

- Отчетность по бизнес-аналитике. Позволяет создавать оперативные рабочие отчеты на основе данных, которые хранятся в базе данных SQL Azure, и развертывать их в облаке. Служба построена на основе тех же технологий, что и службы SQL Server Server Reporting Services, и позволяет пользоваться знакомыми средствами создания отчетов. Доступ к отчетам осуществляется через портал управления Windows Azure, через веб-браузер, а также прямо из ваших приложений Windows Azure и собственных приложений конечных пользователей. Дополнительные сведения см. в разделе «*Отчетность SQL Azure*» по адресу http://msdn.microsoft.com/ruru/library/gg430130.aspx.
- Приложение Marketplace. Это портал, где разработчики могут искать, передавать, продавать и покупать компоненты систем, учебные материалы, шаблоны служб, наборы данных, а также готовые услуги и приложения, необходимые для создания приложений на платформе Windows Azure. Дополнительные сведения см. в разделах «*Windows Azure Marketplace DataMarket*» по адресу http://msdn.microsoft.com/ru-ru/library/gg315539.aspx и «*Windows Azure Marketplace*» (AppMarket) по адресу http://windowsazure.pinpoint.microsoft.com/ru-ru/Default.aspx.

Разработка приложений для Windows Azure

В этом разделе описаны средства и ресурсы разработчика, предназначенные для создания приложений Windows Azure и SQL Azure.

Комментарий Маркуса:

Разработчик может создавать и тестировать приложения Windows Azure на своем компьютере с помощью эмуляторов вычислений и хранилища данных.

Обычно на платформе Windows используется Visual Studio 2010 со средствами Windows Azure для Microsoft Visual Studio. Средства Windows Azure включают все возможности, необходимые для создания приложений Windows Azure, включая локальные эмуляторы вычислений и хранилища данных, запускаемые на компьютере разработчика. Это означает, что разработчик может на своем компьютере создавать, тестировать и отлаживать приложения, а затем развертывать их в облаке. Кроме того, сюда входят средства развертывания приложений в Windows Azure и управления этими приложениями после развертывания.

Средства Windows Azure для Microsoft Visual Studio и средства разработки для других платформ и языков (iOS, Eclipse, Java и PHP) можно загрузить на странице «*Средства Windows Azure*» по адресу http://www.microsoft.com/windowsazure/tools/.

Большой выбор полезных видеоматериалов, примеров краткого руководства и упражнений Hands-On-Lab по множеству тем, которые помогут научиться создавать приложения Windows Azure, см. в разделах «*Знакомство с Windows Azure и SQL Azure*» по адресу http://www.microsoft.com/windowsazure/tutorials/ и «*Проектирование, кодирование, масштабирование*» по адресу http://www.microsoft.com/windowsazure/getstarted/.

Раздел MSDN «*Разработка приложений для Windows Azure*» по адресу http://msdn.microsoft.com/ruru/library/gg433098.aspx содержит конкретные примеры и рекомендации для создания размещенных служб, где для пакетирования и развертывания приложений используется набор средств Windows Azure SDK Tools, а также полезный пример для быстрого начала работы.

Учебный комплект Windows Azure включает практические упражнения, которые помогут быстрого начать работу. Его можно загрузить на странице http://www.microsoft.com/downloads/details.aspx?FamilyID=413E88F8-5966-4A83-B30953B7B77EDF78&displaylang=ru.

Разобраться с работой ролей Windows Azure и жизненным циклом выполнения можно в разделе «*Реальный мир: жизненный цикл роли Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/hh127476.aspx. Список ресурсов, полезных для разработки и развертывания баз данных в SQL Azure, см. в разделе «*Разработка (база данных SQL Azure)*» по адресу http://msdn.microsoft.com/ru-ru/library/ee336225.aspx.

Перечень средств, которые помогут запланировать перенос приложения в Windows Azure, см. в разделе «*Оценка и планирование*» статьи «*Средства платформы Windows Azure*» по адресу http://www.microsoft.com/windowsazure/tools/#assessment.

Обновление приложений Windows Azure

После развертывания приложения в Windows Azure необходимо его обновление в случае изменения служб ролей в соответствии с новыми требованиями, а также по мере совершенствования кода и исправления ошибок. Можно просто развернуть службу заново. Для этого необходимо приостановить и удалить службу, а затем развернуть ее новую версию. При этом простоя приложений можно избежать, если использовать поэтапное развертывание (когда новый пакет загружается и замещает существующую производственную версию) или обновление на месте (когда новый пакет загружается и применяется к работающим экземплярам службы).

О том, как обновлять службы путем загрузки нового пакета и замены существующей производственной версии этим пакетом, см. в разделе «*Обновление службы методом замены (VIP)*» по адресу http://msdn.microsoft.com/ru-ru/library/ee517253.aspx.

О том, как обновлять службы на месте, в том числе о развертывании служб в обновляемом домене или при обработке отказа, а также о влиянии такого развертывания на варианты обновления, см. в разделе «*Выполнение обновления на месте*» по адресу http://msdn.microsoft.com/ru-ru/library/ee517255.aspx.

Примечание.

Если необходимо только изменить данные конфигурации для службы без развертывания нового кода, можно использовать веб-портал, или с помощью API управления изменить файл конфигурации службы, или загрузить новый файл конфигурации.

Управление, мониторинг и отладка приложений Windows Azure

В этом разделе описаны средства и ресурсы управления, которые удобно использовать для развертывания, мониторинга и отладки приложений, а также для управления приложениями в Windows Azure и SQL Azure.

Все подсистемы хранения данных и управления в Windows Azure используют интерфейсы REST. Они не зависят от какой-либо технологии .NET Framework или операционной системы Microsoft Windows ®.
Любая технология, способная создавать запросы HTTP или HTTPS, может обращаться к средствам Windows Azure.
Сведения о собственных и управляемых Windows Azure библиотечных API, а также REST API служб хранения данных см. в разделе «*Справочник по API для Windows Azure*» по адресу http://msdn.microsoft.com/ruru/library/ff800682.aspx.

API управления службами на основе REST может использоваться в качестве альтернативы веб-порталу управления Windows Azure. Этот API включает функции для работы с учетными записями хранилища, размещенными службами, сертификатами, группами привязки, местами расположения и информацией о подписках. Дополнительные сведения см. в «*Справочнике по REST API управления службами Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/ee460799.aspx.

Комментарий Маркуса:

Windows Azure включает компоненты для мониторинга и отладки служб, размещенных в облаке.

Помимо этих компонентов, платформа Windows Azure реализует службы диагностики для таких задач, как отслеживание за работоспособностью приложения. Пакет управления Windows Azure и Operations Manager 2007 R2 могут быть использованы для обнаружения приложений Windows Azure, получения данных о состоянии каждого экземпляра роли, сбора и отслеживания информации о производительности, сбора и отслеживания событий Windows Azure, а также для сбора и отслеживания сообщений трассировки .NET Framework от каждого экземпляра роли. Дополнительные сведения см. в разделе «*Мониторинг приложений Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/gg676009.aspx.

Сведения об использовании встроенных объектов трассировки Windows Azure для настройки диагностики и средств без использования Operations Manager, а также о загрузке результатов см. в разделе «*Сбор журнальных данных с помощью средств диагностики Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/gg433048.aspx.

Сведения об отладке приложений Windows Azure см. в разделах «*Устранение неполадок и отладка в Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/gg465380.aspx
и «*Отладка приложений в Windows Azure*» по адресу
http://msdn.microsoft.com/ruru/windowsazure/WAZPlatformTrainingCourse_WindowsAzureDebuggingLab.

Примечание.

Глава 7, «*Управление жизненным циклом приложений для приложений Windows Azure*», данного руководства содержит сведения по управлению приложениями Windows Azure.

Управление базами данных SQL Azure

Приложения производят доступ к базам данных SQL Azure точно так же, как и к локальным серверам SQL Server — через классы доступа к данным ADO.NET и встроенные технологии доступа к данным — ODBC, PHP или JDBC.

Управление базами данных SQL Azure осуществляется через веб-портал, среду SQL Server Management Studio, с помощью средств базы данных Visual Studio 2010 и целого ряда других средств для выполнения таких операций, как перемещение и перенос данных, а также с помощью средств командной строки для развертывания и администрирования.

Диспетчер базы данных также упростит работу с базами данных SQL Azure. Дополнительные сведения см., в разделе «*Диспетчер баз данных для SQL Azure*» по адресу http://msdn.microsoft.com/ruru/library/gg442309.aspx. Список других средств содержится в разделе «*Средства платформы Windows Azure*» по адресу http://www.microsoft.com/windowsazure/tools/#sqlazure.
SQL Azure поддерживает API управления, а также управление через веб-портал. Сведения об API управления SQL Azure см. в разделе «*Справочник по REST API (SQL Azure)*» по адресу http://msdn.microsoft.com/ruru/library/gg715283.aspx.

Подписка на Windows Azure и модель выставления счетов

В этом разделе описана модель выставления счетов за подписку на Windows Azure и SQL Azure и использование этих служб. Для использования Windows Azure необходимо сначала создать учетную запись выставления счетов, подписавшись на службу Microsoft Online Services на странице https://mocp.microsoftonline.com/ или через портал Windows Azure по адресу https://windows.azure.com/. Клиентский портал Microsoft Online Services управляет подписками на все службы Microsoft. Windows Azure является одной из таких служб. Есть и другие службы, например, платформа Office 365, Windows Office Live Meeting и Windows Intune.

Комментарий По:

У владельца учетной записи и администратора службы для подписки могут быть (и во многих случаях должны быть) разные Live ID.

Каждая учетная запись выставления счетов имеет одного владельца учетной записи, определяемого по идентификатору Windows Live. Владелец учетной записи может создавать подписки и управлять ими, просматривать данные для выставления счета и данные об использовании, а также указывать администратора службы для каждой подписки. Подписка Windows Azure является лишь одной из таких подписок.

Администраторы управляют отдельными размещенными службами для подписки Windows Azure с помощью портала Windows Azure на странице https://windows.azure.com/. В состав подписки Windows Azure может входить следующее.

- Размещенные службы, состоящие из ролей и экземпляров в каждой роли. Роли и экземпляры могут быть остановлены в производственном (production) режиме или в тестовом (staging) режиме.
- Учетные записи хранилищ, состоящие из экземпляров хранилищ таблиц, больших двоичных объектов и очередей.
- Экземпляры сети доставки содержимого.
- Базы данных SQL Azure.
- Экземпляры служб SQL Azure Reporting Services.
- Экземпляры служб управления доступом, шины интеграции и кэша.
- Экземпляры подключения виртуальной сети и диспетчера трафика.

На рисунке 1 показана конфигурация тарификации Windows Azure для стандартной подписки.

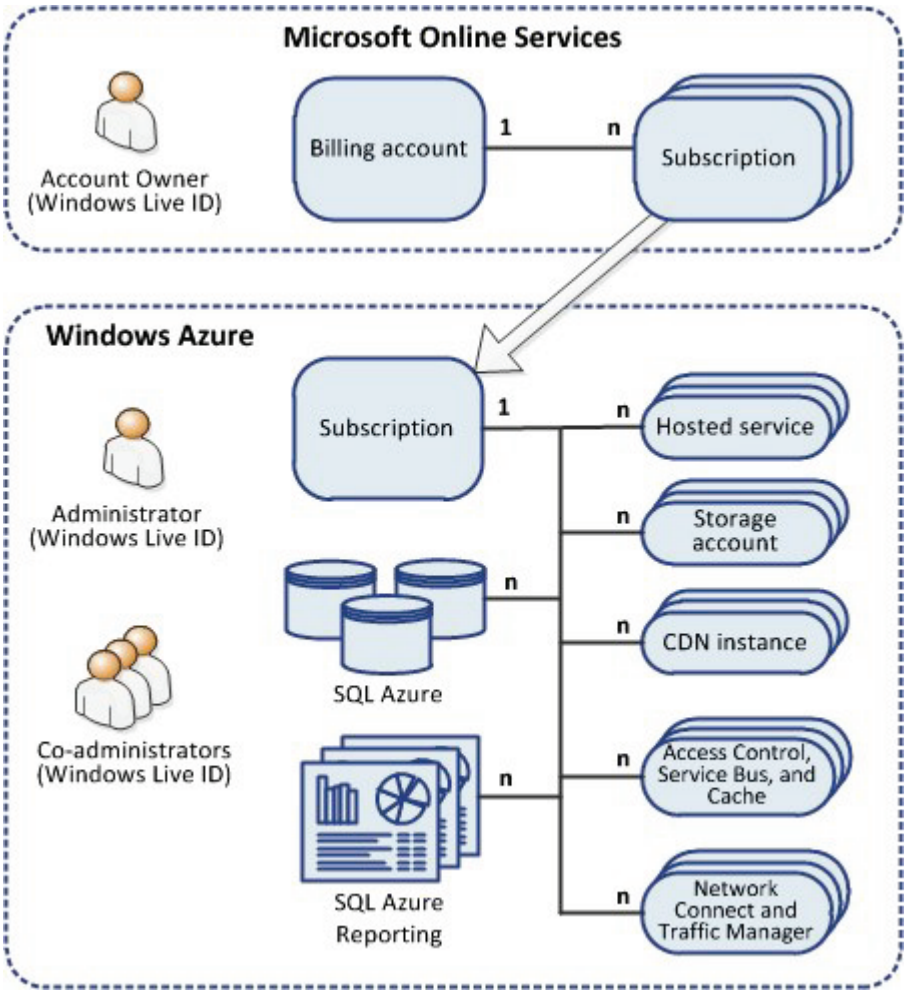


Рис. 1 **Конфигурация тарификации Windows Azure для стандартной подписки**

Дополнительные сведения о тарификации Windows Azure см. в разделах «*Основы тарификации Windows Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/ff959173.aspx и «*Учетные записи и тарификация в SQL Azure*» по адресу http://msdn.microsoft.com/ru-ru/library/ee621788.aspx.

Оценка затрат

Оплата в Windows Azure взимается за использование таких служб, как вычислительные ресурсы времени, хранилище, трафик и т.п. Оплата за вычисленные ресурсы рассчитывается на почасовой и зависит от выбранного размера экземпляра роли. Оплата хранилища определяется размеров хранимых и количеством транзакций. Цены на трафик различаются в зависимости от региона, тарифицируется только исходящий трафик из дата-центров, т.е. никакой трафик внутри дата-центров не тарифицируется.

Для оценки вероятных затрат на подписки Windows Azure можно использовать следующие ресурсы.

- Обзор подписки для различных моделей закупок, например для модели повременной оплаты и модели подписки, включая инструмент для измерения потребления, по адресу http://www.microsoft.com/windowsazure/pricing/.
- Калькулятор цен по адресу http://www.microsoft.com/windowsazure/pricing-calculator/.
- Калькулятор совокупной стоимости владения по адресу http://www.microsoft.com/windowsazure/offers/#tcoCompare-LB.

Комментарий По:

Оплата вычислительных ресурсов производится не только за запущенные экземпляры, но и за остановленные экземпляры, т.е. за любые развернутые экземпляры. Если вы не хотите оплачивать ту или иную службу, удалите связанные с ней развертывания

Примечание.

В главе 4 данного руководства приведены дополнительные сведения об оценке стоимости размещения приложений в Windows Azure.

Дополнительные сведения

О платформе Windows Azure имеется много сведений в документации, учебных видеофильмах и технических документах. Ниже приведены некоторые веб-сайты, где можно найти дополнительные сведения.

- Веб-сайт для этой серии руководств по адресу http://wag.codeplex.com/ содержит ссылки на интерактивные ресурсы, образцы кода, пособия Hands on-Labs, отзывы и многое другое.
- Портал сведений о Microsoft Windows Azure находится по адресу http://www.microsoft.com/windowsazure/. Там можно найти ссылки на технические документы и такие инструменты, как пакет Windows Azure SDK, а также многие другие ресурсы. Кроме того, здесь можно подписаться на учетную запись Windows Azure.
- Райан Данн (Ryan Dunn) и Стив Маркс (Steve Marx) записали на канале 9 ряд передач с обсуждением Azure. Эти материалы можно найти в Cloud Cover по адресу http://channel9.msdn.com/shows/Cloud+Cover/.
- Ответы на вопросы можно найти на форуме Windows Azure по адресу http://social.msdn.microsoft.com/Forums/ru-ru/windowsazure/threads.
- Стив Маркс — технический специалист по вопросам стратегии Windows Azure. Его блог находится по адресу http://blog.smarx.com/. Это интересный источник новостей и информации о Windows Azure.
- Райан Данн — технический специалист по Windows Azure. Его блог находится по адресу http://dunnry.com/blog.
- Юджении Пейс (Eugenio Pace), руководитель программы в группе архитектуры и рекомендаций корпорации Microsoft, создает серию руководств по Windows Azure, в которую входит эта документация. Для получения дополнительных сведений об этой серии просмотрите его блог по адресу http://blogs.msdn.com/eugeniop.
- Скотт Денсмор (Scott Densmore), ведущий разработчик группы архитектуры и рекомендаций корпорации Microsoft, пишет о разработке приложений для Windows Azure в своем блоге адресу http://scottdensmore.typepad.com/.
- Джим Накашима (Jim Nakashima) — руководитель программы в группе, которая создает инструменты для Windows Azure. Его блог содержит множество технических подробностей и советов. Адрес блога — http://blogs.msdn.com/jnak/.
- Код и документация для проекта руководства по архитектуре и рекомендациям для Windows Azure доступен на сайте руководства CodePlex Windows Azure http://wag.codeplex.com/.
- Полные руководства и примеры службы управления доступом к Windows Azure имеются в книге по архитектуре и рекомендациям «*Руководство по управлению доступом и удостоверениями на основе утверждений*», доступной также в Интернете по адресу http://claimsid.codeplex.com/.

Глава 2. Сценарий Tailspin

В этой главе мы знакомимся с вымышленной компанией Tailspin. В главе описываются планы компании по запуску новой службы в сети под названием Surveys, которая позволит другим компаниям и отдельным пользователям проводить собственные интерактивные опросы. Здесь также объясняется почему в Tailspin планируют разместить приложение на платформе Windows Azure. Как и в любой компании, рассматривающей этот процесс, существует множество вопросов и проблем, требующих решения, особенно в связи с тем, что компания Tailspin использует облачную технологию впервые. В следующих главах поэтапно демонстрируется разработка и проектирование компанией Tailspin приложения Surveys, которое должно работать в Windows Azure.

Компания Tailspin

Компания Tailspin, имеющая 20 сотрудников, является вымышленной компанией ISV, специализирующейся в разработке решений с использованием технологий Microsoft®. Разработчики Tailspin осведомлены о различных продуктах и технологиях Microsoft, включая .NET Framework, ASP.NET MVC, SQL Server® и среду разработки Microsoft Visual Studio®. Эти разработчики знакомы с Windows Azure, но еще не разрабатывали полных приложений для платформы.

Приложение Surveys является первой из нескольких инновационных услуг в сети, которые компания Tailspin хочет вывести на рынок. Компания Tailspin хочет разработать и запустить эти услуги с минимальными инвестициями в оборудование и ИТ-персонал. Разработчики Tailspin надеются, что некоторые из этих услуг будут развиваться быстрыми темпами, и компания хочет иметь возможность быстро реагировать на растущую потребность. Они прекрасно осознают, что некоторые из этих услуг не найдут применения, и не хотят иметь избыточное, простаивающее оборудование.

Стратегия Tailspin

Tailspin является инновационной и гибкой организацией, имеющей возможность использовать новейшие технологии и производственные возможности, предлагаемые облачными вычислениями.
На первоначальном этапе компания готова идти на риск и использовать новые технологии, если это дает возможность реализовать приложения. Tailspin планирует включить в свои разработки технологию облака и получить конкурентные преимущества первопроходца. В компании надеются получить определенный опыт, а затем быстро использовать его в своих разработках. Эта стратегия может быть описана как «пытаться, ошибаться на первоначальном этапе, учиться, а затем вновь повторять попытку». В Tailspin решено начать с приложения Surveys в качестве первого сервиса на основе облака.

Приложение Surveys

Приложение Surveys позволяет клиентам Tailspin проектировать опросы, публиковать их и собирать результаты опроса для последующего анализа. Опрос — это набор нескольких типов вопросов, предлагающих множественный выбор, числовой диапазон или произвольный текст. Клиенты начинают с создания подписки с помощью службы Surveys, которую они используют для управления их опросами и для применения брэндинга с помощью стилей и эмблем. Клиенты могут также выбрать географический регион для своих учетных записей, чтобы они могли размещать свои опросы как можно ближе к аудитории опросов.
Приложение Surveys позволяет пользователям бесплатно попробовать приложение, а также подписаться на один из нескольких различных пакетов.

На рис. 1 демонстрируется приложение Surveys и выделяются три различные группы пользователей, которые взаимодействуют с приложением.

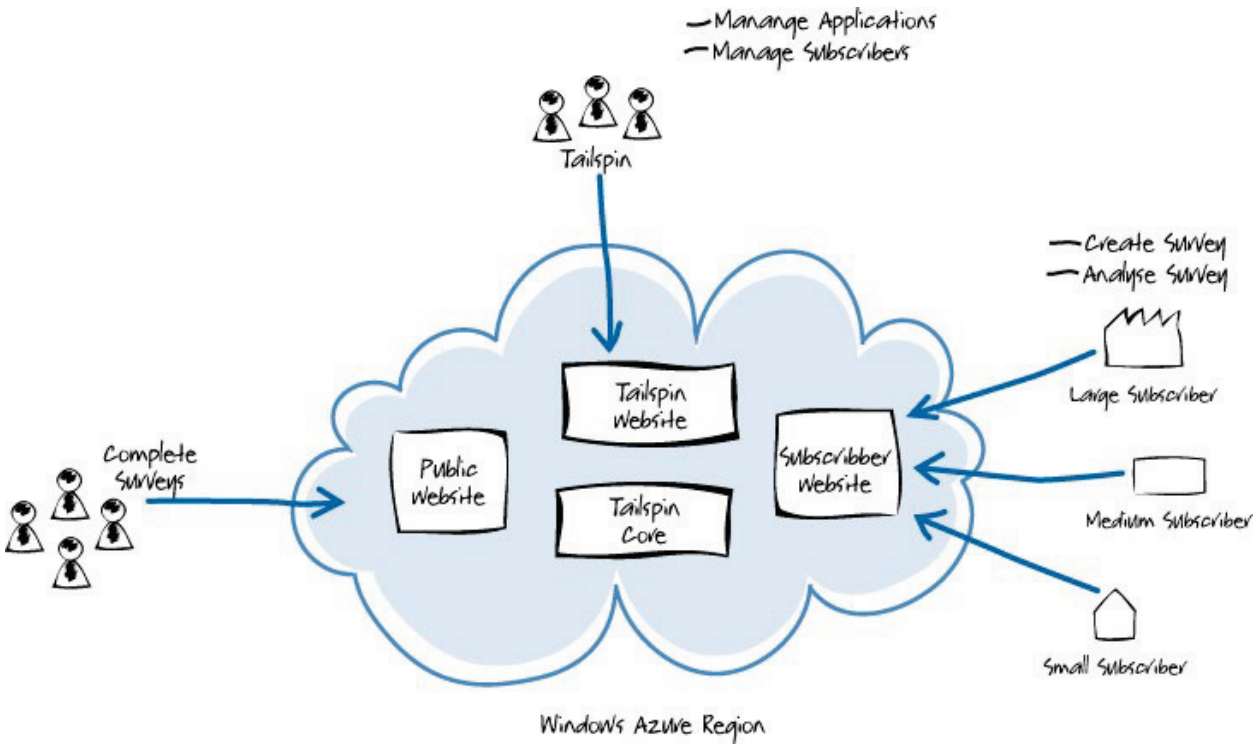


Рис. 1

Клиенты, подписанные на услугу Surveys (или использующие ее бесплатную пробную версию), имеют доступ к веб-сайту Subscribers, который позволяет им разрабатывать свои собственные опросы, применять брэндинг и кастомизация, собирать и анализировать результаты опросов. В зависимости от выбранного пакета они имеют доступ к различным уровням функциональных возможностей в рамках приложения Surveys. В Tailspin ожидают, что их услуги могут понадобиться клиентам как крупного, так и малого бизнеса по всему миру. Кроме того, клиенты могут выбирать географический регион для своих учетных записей и для опросов.

Комментарий Бхарата:

В сфере программного обеспечения как услуги (SaaS) подписчики обычно называются «клиентами». Такие приложения, как Surveys (Tailspin), часто называются мультитенантными приложениями.

В Tailspin стремятся разрабатывать сервис таким образом, чтобы большинство административных задач и задач по настройке клиенты могли выполнять самостоятельно с минимальным участием персонала Tailspin.

Общедоступный веб-сайт позволяет пользователям, участвующим в опросе, давать ответы на вопросы через веб интерфейс.

Примечание.

Сведения о создании клиентского приложения Windows Phone 7 для приложения Tailspin Surveys см. в книге *Руководство разработчика Windows Phone 7,* доступной в Интернете по адресу http://msdn.microsoft.com/ruRU/library/gg490765.aspx.

Веб-сайт Tailspin позволяет сотрудникам компании управлять приложением и учетными записями подписчиков. Все три веб-сайта взаимодействуют с основными службами, которые входят в состав приложения Surveys и обеспечивают доступ к хранилищу данных приложения.

Цели и задачи компании Tailspin

Компании Tailspin необходимо решить несколько задач, как организационных, так и связанных с приложением Surveys. Во-первых, клиентам, возможно, потребуется создавать опросы, связанные с запуском продукта и с маркетинговой кампанией. Кроме того, опросы могут быть сезонными, возможно связанными с отпускным периодом. Часто клиенты, которые используют приложение опроса, хотят создавать эти опросы за очень короткий срок. Опросы обычно проводятся в определенный короткий период времени, но количество респондентов может быть значительным. Это означает, что приложение Surveys будет использоваться, как правило, короткий период времени, и в Tailspin практически невозможно определить заранее эти периоды. Компания Tailspin хочет предлагать приложение Surveys клиентам по всему миру, и из-за того, что потребность в приложении Surveys может возникать внезапно, в компании хотят иметь возможность быстро увеличивать или сокращать свою инфраструктуру в различных географических регионах. В компании не хотят приобретать оборудование и управлять им и поддерживать достаточные мощности для максимально возможного количества посетителей.

Комментарий Бхарата:

Основными свойствами платформы Windows Azure являются эластичность (увеличение или уменьшение используемых ресурсов по требования, т.е. в зависимости от текущей нагрузки) и возможность работы в различных географических регионах.

Tailspin хочет сохранить свое конкурентное преимущество, быстро улучшая функции существующих сервисов, и получить конкурентное преимущество первой компании, вышедшей на рынок с новыми продуктами и услугами.

С приложением Surveys компания Tailspin хочет предложить своим клиентам надежный, настраиваемый и гибкий сервис создания и управления интерактивными веб-опросами.

В Tailspin хотят иметь возможность предлагать клиентам различные пакеты (по разным ценам), основанные на конкретных требованиях заказчика. В Tailspin хотят предлагать наиболее крупным клиентам возможность интеграции приложения Surveys в их собственную инфраструктуру. Например, интеграция с собственной инфраструктурой идентификации клиента может обеспечить единый вход (SSO) или позволит нескольким пользователям управлять опросами или обращаться к данным тарификации. Интеграция с собственными системами бизнес-аналитики (BI) клиента могла бы обеспечить более сложный анализ результатов опросов. Для небольших клиентов, у которых нет необходимости или возможности использовать сложные функции интеграции, основной пакет может включать систему проверки подлинности. Диапазон доступных пакетов должен также включать бесплатную пробную версию, чтобы клиенты имели возможность пробного использования приложения Surveys перед его приобретением.

У портала подписчиков и общедоступных веб-сайтов опросов также имеются различные требования масштабируемости. Вполне вероятно, что заполнять опрос могут тысячи пользователей, но лишь небольшое число пользователей каждого подписчика будут изменять существующие опросы или создавать новые опросы. Компания Tailspin хочет оптимизировать ресурсы для каждого из этих сценариев.

Бизнес-модель Tailspin предусматривает взимание с подписчиков ежемесячной платы за службу, например за приложение Surveys, и в условиях работы на мировом рынке необходимо, чтобы цены были конкурентоспособными. Далее, компания Tailspin должна оплачивать фактические расходы на обеспечение работы приложения, поэтому для поддержания рентабельности своих продаж Tailspin должна строго управлять текущими затратами на использование служб, предлагаемых клиентам.

Примечание.

В этом сценарии клиенты компании (подписчики) *не являются* клиентами Windows Azure. Подписчики платят компании Tailspin, которая в свою очередь платит Microsoft за использование компонентов платформы Windows Azure.

В Tailspin хотят обеспечить безопасность данных клиентов. Например, данные клиента должны быть доступными исключительно этому клиенту, должно быть несколько физических копий данных опроса, и нужно исключить потерю данных в результате случайного удаления опроса клиентом. Кроме того, когда компания Tailspin обновляет приложение, все данные опроса должны сохраняться.

Наконец, для построения приложения Surveys в Tailspin хотели бы использовать существующие навыки разработчиков и свести к минимуму любое повторное обучение.

Архитектура приложения Surveys

Для достижения вышеописанных целей в компании Tailspin решили реализовать приложение Surveys на облачной платформе Windows Azure. На рис. 2 показано высокоуровневое представление этой архитектуры.

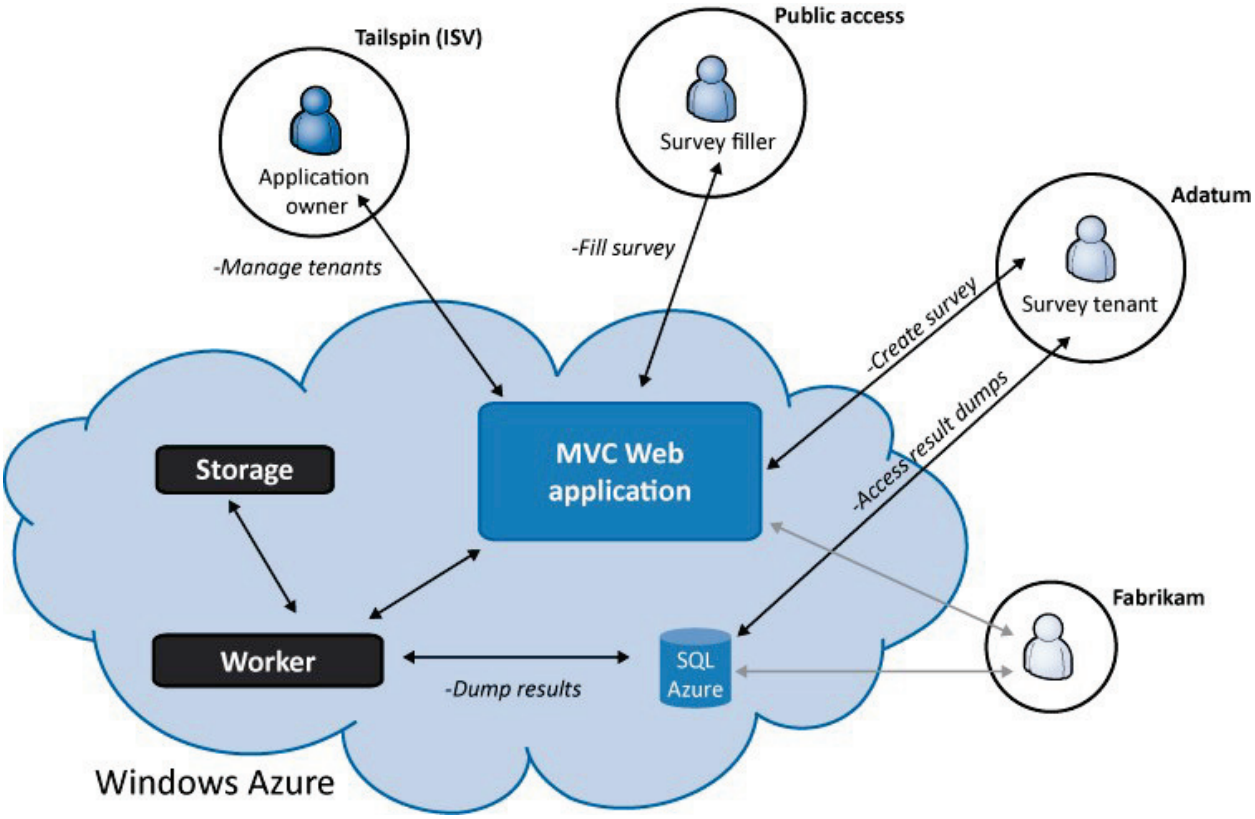


Рис. 2 Архитектура приложения Surveys

Архитектура приложения Surveys

Приложение Surveys имеет простую архитектуру, которую используют многие другие приложения Windows Azure. Ядро приложения использует веб-роли, рабочие роли и хранилище Windows Azure. На приведенном далее рисунке показаны три группы пользователей, обращающихся к приложению: владелец приложения, пользователи и подписчики услуги Surveys (в данном примере Adatum и Fabrikam).

На этом рисунке также показано, как приложение использует технологическую платформу SQL Azure для обеспечения механизма вывода подписчиками результатов их опроса в реляционную базу данных для последующего детального анализа. В этом руководстве подробно рассматриваются архитектура приложения и ее реализация, описываются различные веб-роли и рабочие роли приложения Surveys.

Некоторые из конкретных вопросов, рассматриваемых в этом руководстве, касаются способа реализации компанией Tailspin приложения Surveys, которое является мультитенантным приложением в Windows Azure, а также подходы к тестированию. В этом руководстве описывается как Tailspin реализует интеграцию механизма проверки подлинности приложения с инфраструктурой безопасности клиента с использованием федеративных удостоверений. В руководстве описываются причины выбора модели хранения данных, которая сочетает хранилище на базе Windows Azure и SQL Azure. Рассматриваются также другие разделы, в которых рассматривается механизм кэширования для повышения производительности веб-сайта, автоматизации создания рабочей области новой подписчики (тенанта) и механизма выставления счетов, используемой компанией Tailspin для приложения Surveys.

Комментарий Яны:

Tailspin может использовать CDN Windows Azure для кэширования содержимого.

Tailspin в качестве основным платформ: Visual Studio 2010, ASP.NET MVC 3.0 и .NET Framework 4.

Глава 3. Размещение мультитенантных приложений в Windows Azure

В этой главе рассматриваются некоторые вопросы, касающиеся проектирования и разработки мультитенантных приложений в Windows Azure. Облачная платформа с высокой степенью масштабируемости предлагает набор функций для построения сервисов, которые предоставляются конечным пользователям по модели SaaS. Мультитенантная архитектура, в которой несколько подписчиков совместно используют приложение, обеспечивает эффективную экономию, поскольку клиенты совместно используют ресурсы, но при этом приложение с технической точки зрения является более сложным, т.к. необходимо управлять различными подписками независимо друг от друга.

Мультитенантность

Multitenancy (англ. Tenancy — аренда, Multitenancy — множественная аренда) — элемент архитектуры программного обеспечения, где единый экземпляр объекта приложения, запущенного на сервере, обслуживает множество клиентов организаций (арендаторы). Multitenancy сопоставляется с архитектурой из множественных экземпляров (multiinstance), где разделённые программные экземпляры (или аппаратная часть системы) настроены для различных клиентов организаций. С архитектурой множество-арендаторов (multitenant) программные приложения предназначаются виртуальным разделам со своими данными и конфигурациями и каждый клиента организации, работает с экземпляром настроенного виртуального приложения. [wiki]

В этой главе не рассматривается конкретно компания Tailspin или приложение Surveys, но в ней рассматриваются факторы, влияющие на итоговое решение об использовании стандартной архитектуры или мультитенантной.

Эта глава является концептуальной основой для понимания некоторых тем, которые более подробно рассматриваются в последующих главах данного руководства.

Выделенная (single-tenant) и мультитенантная (multi-tenant) архитектуры

Одним из первых архитектурных решений, принятых в Tailspin по приложению Surveys, был выбор между выделенной или мультитенантной архитектурой (последняя позволяет поддерживать несколько клиентов). На рис. 1 показана разница между этими подходами на высоком уровне. Выделенная модель имеет отдельный, логический экземпляр приложения для каждого клиента, в то время как мультитенантная модель имеет одно **логическое экземпляр**[1] приложения, совместно используемого многими клиентами. Важно отметить, что мультитенантная модель по-прежнему предлагает пользователям разграниченный доступ к данных приложения. В приложение Surveys клиент B не должен видеть или изменять опросы или данные клиента A. Компания Tailspin, будучи владельцем приложения, имеет полный доступ ко всем данным, хранящимся в ее приложении.

1 Под логическим экземпляром мультитенатного приложение подразумевается одно развертывание или инсталляция приложения, а не количество запущенных экземпляром веб или рабочей ролей.

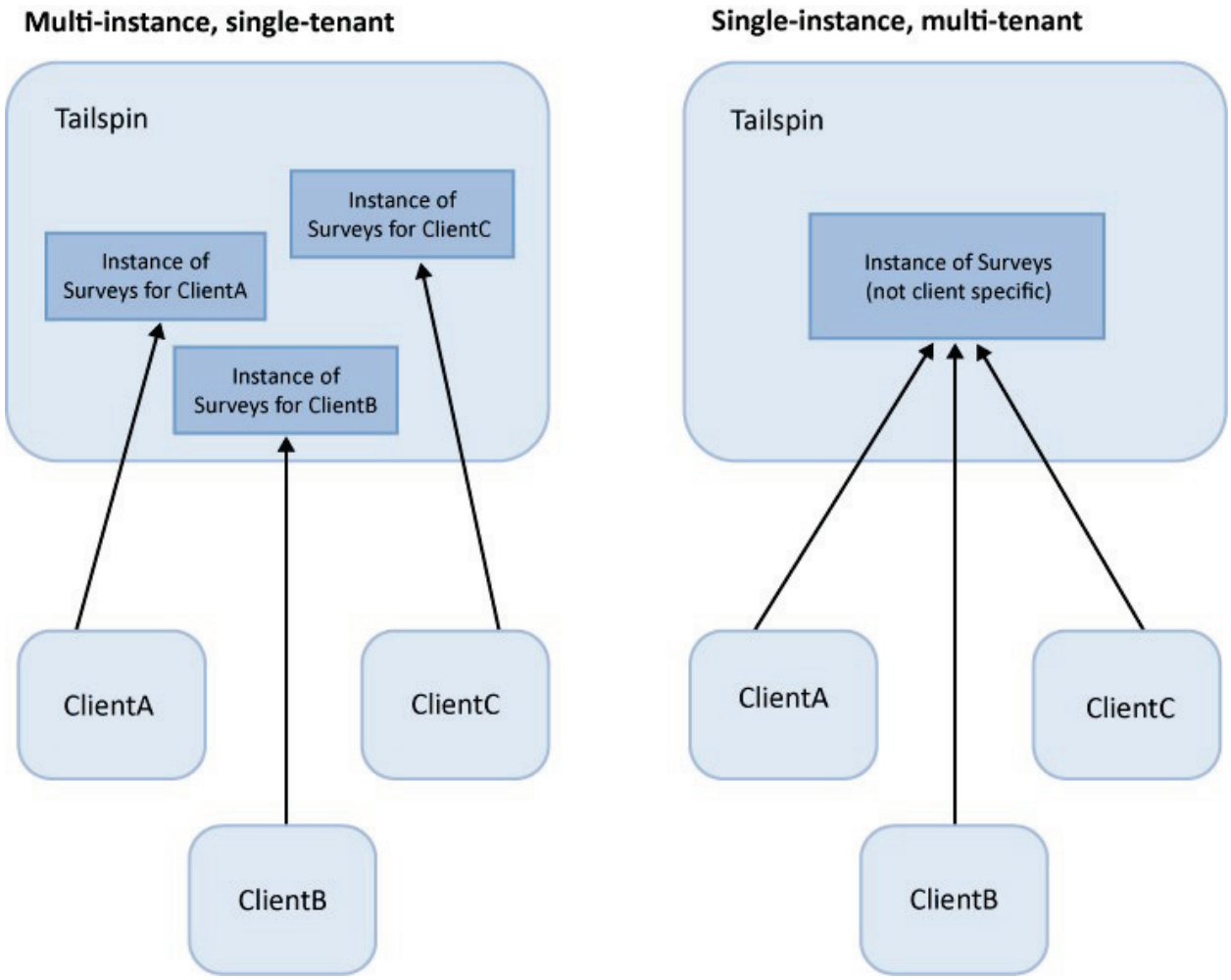


Рис. 1 **Логическое представление выделенной и мультитенантной архитектуры**

Комментарий Бхарата:

На этой диаграмме показаны логические экземпляры приложения Surveys. На практике вы можете реализовать каждый логический экземпляр как несколько физических экземпляров, чтобы обеспечить масштабирование приложения.

Мультитенантная архитектура в Azure

В Windows Azure различие между выделенной и мультитенантной моделями не так очевидно, как для модели на рис. 1, поскольку приложение в Windows Azure может выключить несколько компонентов, каждый из которых может быть выделенным или мультитенантным. Например, если приложение имеет компонент пользовательского интерфейса (UI), компонент веб-служб и компонент хранилища, возможная архитектура может выглядеть так, как это показано на рис. 2.

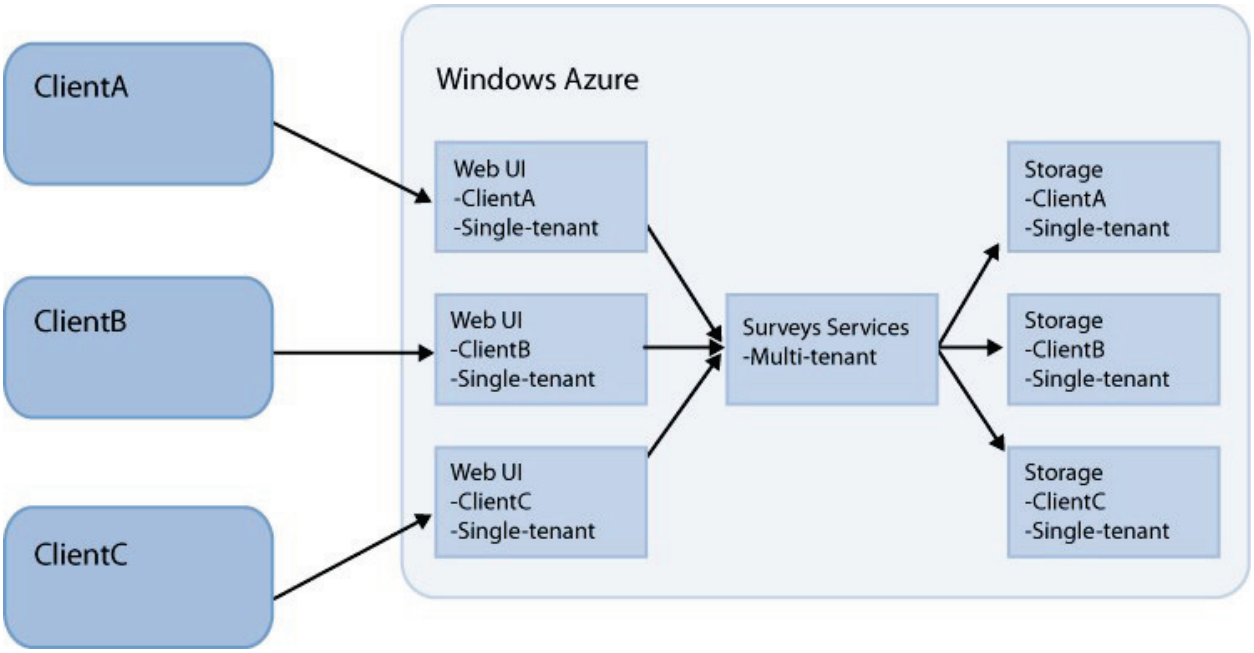


Рис. 2 **Пример архитектуры для Windows Azure**

Это не единственная возможная архитектура, но она демонстрирует, что для каждого компонента независимо друг от друга может быть выбрана собственная модель (выделенный или мультитенантная).

Каким образом следует разрабатывать приложение Windows Azure: как выделенное или мультитенантное? Нет правильного или неправильного ответа, но, как будет видно из следующего раздела, есть ряд факторов, которые могут повлиять на ваш выбор.

В мультитенантном приложение вы всегда можете эмулировать выделенный режим (один подписчик), тогда как возможности расширить выделенное приложение до нескольких независимых подписок нет.

Выбор выделенной или мультитенантной архитектуры

В данном разделе представлены некоторые критерии, которые следует учитывать при выборе модели. В последующих главах книги мы рассмотрим применение этих применительно к компании Tailspin и приложению Surveys.

Архитектура

Требования к архитектуре приложения будут влиять на выбор выделенной или мультитенантной архитектуры.

Отказоустойчивость

Мультитенантное приложение более уязвимо к сбою экземпляра, чем выделенное приложение. Сбой выделенного экземпляра влияет только на клиента, который использует данный экземпляр, тогда как сбой мультитенатного экземпляра затронет всех клиентов. Однако Windows Azure может уменьшить этот риск, позволяя выполнить развертывание нескольких идентичных копий приложения за счет использования нескольких экземпляров ролей Windows Azure (такое подход как раз в полной мере реализует мультитенантную модель с несколькими экземплярами). Windows Azure автоматически распределяет нагрузку по обработке запросов между данными экземплярами ролей, следовательно, приложение должно быть разработано таким образом, чтобы оно корректно функционировало при развертывании нескольких экземпляров. Например, если приложение использует состояние сеанса, необходимо убедиться, что каждый экземпляр веб-роли может получить доступ к состоянию. Windows Azure выполняет постоянный мониторинг всех экземпляров ролей и автоматически перезапускает экземпляры при возникновении сбоя.

Повышение устойчивости приложения

Масштабируемость приложения, выполняющегося в Windows Azure, в значительной степени зависит от возможности развернуть несколько экземпляров веб-роли и рабочей роли на вычислительных узлах и возможности получать доступ к одним и тем же данным с этих узлов. Данный принцип может быть использован как выделенным, так и мультитенантным приложением в Windows Azure. Windows Azure предлагает также различные размеры вычислительных ресурсов (Extra Small, Small, Medium, Large, Extra Large), что позволяет гибко масштабировать экземпляры приложения.

Комментарий Бхарата:

В Windows Azure рекомендуемым способом масштабирования приложения является добавлением дополнительных экземпляров приложения, а не увеличение ресурсов конкретного экземпляра (например, с Medium до Large).

Для некоторых приложений может быть не целесообразным совместное использование всеми клиентами одного мультитенантного экземпляра. Например, может понадобиться сгруппировать клиентов на основе функциональности приложения, которую они используют, а затем оптимизировать каждый экземпляр для каждой из созданных групп. В этом случае вам может потребоваться две или более копии мультитенатного приложения, развернутого даже в различных подписка Windows Azure.

Так же для повышения масштабируемости приложения может понадобиться рассмотреть вопрос об использовании Windows Azure AppFabric Caching Service и диспетчера трафика Windows Azure (Traffic Manager). Служба кэша Windows Azure (Caching Service) обеспечивает не только кэширование выходной разметки и данных, но так же включает масштабируемый поставщик сеансов (session provider) для ASP.NET. Диспетчер трафика позволяет распределять трафик и входные запросы пользователей между несколькими экземплярах Windows Azure, даже если эти экземпляры расположены в разных дата-центрах Windows Azure.

Дополнительные сведения о том, как использовать Windows Azure AppFabric Caching Service, см. в главе 6, «Работа с данными в приложение Surveys».

Дополнительные сведения о службе диспетчера трафика Windows Azure см. в разделе «*Диспетчер трафика Windows Azure*» по адресу http://msdn.microsoft.com/ru-RU/WAZPlatformTrainingCourse_WindowsAzureTrafficManager.

Соглашение об уровне обслуживания (SLA)

Вы можете предлагать различные соглашения об уровне обслуживания (SLA) для различных уровней подписки на сервис. Если подписчики с разными соглашениями об уровне обслуживания совместно используют один мультитенантный экземпляр, следует обеспечить самый высокий SLA.
Но если имеется ограниченное число различных SLA, рекомендуется разместить всех клиентов с одинаковым SLA в один мультитенантный экземпляр и убедиться, что экземпляр обладает достаточными ресурсами для удовлетворения требований SLA.

Правовая и нормативная среда

Для некоторых приложений необходимо учитывать конкретные нормативные или юридические вопросы. Для этого могут потребоваться различия в функциональных возможностях, отображение в пользовательском интерфейсе юридических соглашений, обеспечение отдельных баз данных или областей хранилища для каждого конкретного подписчика, расположение сервиса и хранилища в конкретном географическом регионе. При таких условиях может потребоваться реализовать отдельные мультитенантные экземпляры для различных групп пользователей, либо строго выделенные режим для каждого подписки на сервис.

Проверка подлинности и авторизации

Для приложения в облаке может использовать своя собственная система проверки подлинности и авторизации, в этом случае подписчикам приложения потребуется создавать учетные данные для всех пользователей приложения. С другой стороны клиенты могут предпочесть использовать существующую у них систему проверки подлинности и избежать необходимости создавать новые учетные данные для приложения. Второй вариант для мультитенантного приложения означает необходимость поддержки нескольких поставщиков проверки подлинности, а так же может потребовать сопоставление схемы авторизации приложения в облаке со схемой локальной инфраструктуры авторизации. Например, кто-то, являющийся «администратором» в службе каталогов Microsoft Active Directory в компании Adatum, может быть сопоставлен с «администратором» в приложении Surveys (Tailspin) в Adatum.

Примечание.

Дополнительные сведения об удостоверении на основе утверждений, см. в книге *Руководство по идентификаторам и управлению доступом на основе утверждений*. Копию этой книги в формате PDF можно загрузить по адресу http://msdn.microsoft.com/ru-RU/library/ff423674.aspx.

Управление жизненным циклом приложения

Выбор выделенной или мультитенантной архитектуры будет влиять на то, насколько легко выполняется разработка, развертывание, поддержка и управление приложением.

Ведение базы кода

Для компаний-разработчиков ведение отдельных баз кода для различных клиентов сопряжено с увеличением расходов на поддержку и обслуживание, а так же с проблемами при отслеживании того, какие клиенты какую версию используют. Это может привести к появлению ошибок, которые приведут к дополнительным затратам. Мультитенантная система с одним логическим экземпляром гарантирует одну базу кода для всего приложения. Вы можете по-прежнему поддерживать одну базу кода с выделенной моделью с несколькими экземплярами, но может появиться мимолетное искушение (с долгосрочными последствиями) разветвить код для различных клиентов. В некоторых сценариях, где существует потребность в тонкой настройке приложений, несколько баз кода могут быть приемлемым вариантом, но, прежде чем выбрать этот путь, необходимо изучить возможность применения пользовательских конфигураций или пользовательских бизнес-правил на уровне стандартной настройки системы (например, через интерфейс). Если должно быть несколько баз кода, то приложение должно быть структурировано таким образом, чтобы пользовательский код ограничивался только минимально необходимым числом пользовательских компонентов.

Обновления приложения

Мультитенантное приложение позволяет одновременно распространить обновления на всех клиентов. Этот подход означает, что обновляется только один логический экземпляр, что позволяет уменьшить усилия по обслуживанию. Кроме того, у вас есть уверенность, что все клиенты используют последнюю версию приложения, что упрощает поддержку. Домены обновления (Update Domain) Windows Azure упрощают этот процесс, позволяя распространять обновления на несколько экземпляров роли без остановки приложения. Если у клиента есть операционные процедуры или программное обеспечение, привязанные к конкретной версии приложения, то любые обновления должны быть скоординированы с клиентом.

Для сокращения рисков, связанных с обновлением приложения, вы можете реализовать процедуру последовательного обновления: на первой фазе происходит обновления приложения для небольшой группы пользователей, на второй фазе, если с новой версией не возникло никаких сложностей, обновления распространяются на оставшихся пользователей.

Примечание.

Сведения об обновлении на месте (in-place upgrade) , включая сведения о том, каким образом службы развертываются в домене обновления (upgrade domain)и домене ошибок (fault domain) и каким образом это влияет на параметры обновления, см. в разделе «*Выполнение обновления на месте*» по адресу http://msdn.microsoft.com/ruRU/library/ee517255.aspx.
Сведения о том, каким образом обновлять службы путем загрузки нового пакета и замены существующей рабочей версии, см. в разделе «*Как заменить виртуальные IP-адреса служб*» по адресу http://msdn.microsoft.com/ruRU/library/ee517253.aspx.

Мониторинг приложения

Мониторинг одного экземпляра приложения проще, чем мониторинг нескольких экземпляров. В выделенной модели с несколькими экземплярами для любого нового экземпляра роли необходимо устанавливать настройки среды мониторинга, в результате чего увеличивается сложность всего процесса. Если будет решено использовать последовательное обновление, мониторинг будет также более сложным, поскольку необходимо одновременно отслеживать две версии приложения и использовать данные мониторинга для оценки стабильности версии приложения.

Использование поставщиков .NET и сторонних компонентов

Если будет решено использование мультитенантную архитектуру, необходимо оценить насколько хорошо в таком режиме будут работать сторонние компоненты. Что убедиться, что сторонний компонент может работать в мультитенантной архитектуре, возможно, потребуется выполнить некоторые дополнительные действия. Если при выделенном развертывании с несколькими экземплярами необходима возможность масштабирования для крупных клиентов, необходимо также проверить, что сторонние компоненты могут работать в подобном режиме (несколько экземпляров).

Провиженинг тестового доступа и новых клиентов

Провиженинг области для нового клиента или инициализация бесплатного пробного использования службы в значительной степени упрощается, если данный процесс предполагает только изменение конфигурации. Для выделенной модели с несколькими экземплярами требуется развернуть новый экземпляр приложения для каждого клиента, включая тех, которые используют бесплатную пробную версию. Хотя этот процесс вы можете автоматизировать, это будет значительно сложнее, чем изменение или создание данных конфигурации в мультитенантном приложении с одним экземпляром.

Настройка приложения

Вне зависимости от того какая модель используется (выделенная или мультитенантная) клиенты должны иметь возможность настройки и кастомизации облачного приложения.

URL-адреса для доступа к приложению

По умолчанию Windows Azure предоставляет каждому приложению DNS-имя следующего вида <*имяприложения*>.cloudapp.net. Для сопоставления пользовательского DNS-имени с приложением вы можете использовать запись DNS CNAME. Например, если приложение Tailspin Surveys имеет имя tailspin.cloudapp.net, Tailspin может использовать запись CNAME для сопоставления с URL-адреса https://surveys.tailspin.com с приложением. Если каждый клиент имеет свой собственный, отдельный выделенный экземпляр приложения, выполняющегося с помощью отдельной учетной записи Windows Azure, то обычно пользовательское DNS-имя можно сопоставить с конкретным экземпляром приложения. Например, https://surveys.adatum.com и https://surveys.fabrikam.com можно сопоставить с отдельными экземплярами.

Так как службы IIS могут иметь только один SSL-сертификат, связанный с портом, мультитенантное приложение может использовать только одно доменное имя на 443 порту по умолчанию. Поэтому в мультитенантном приложение вы можете использовать схему адресации следующего вида: https://<*azureaccount*>.cloudapp.net/<*app*>/<*tenant*>. Компания Adatum, подписчик приложения Surveys (Tailspin), будет иметь доступ к приложению Surveys по адресу https://services.tailspin.com/surveys/adatum.

Комментарий Бхарата:

Если применять SSL не требуется, то можно использовать пользовательские доменные имена для каждого клиента в мультитенантном приложении. Каждый клиент может создать запись CNAME для сопоставления его доменного имени с приложением Windows Azure.

Настройка приложения клиентом

Скорее всего, клиентам потребуется возможность кастомизировать интерфейс приложения под собственный дизайн. Необходимо определить уровень кастомизации, который будет доступен. Самый простой вариант: клиентам может быть предоставлена возможность настройки внешнего вида приложения, позволяющая заменять каскадные таблицы стилей и файлы изображений. Более сложный вариант: клиентам можно позволить разработку собственных страниц, которые программно взаимодействуют со службам приложения через предоставляемые API функции.

Для некоторых приложений может потребоваться предоставить клиентам возможность включать или отключать определенные функции. Например, в приложении Surveys клиенты могут выбирать интеграцию инфраструктуры идентификации приложения с собственной инфраструктурой, а также выбирать географическое расположение создаваемых опросов. Настройки подобного типа удобно размещать в хранилище таблиц Windows Azure (Table Storage).

Для других приложений может потребоваться возможность позволить пользователям в некоторой степени настраивать бизнес-процессы. В этом случае возможны следующие варианты: реализация подключаемых модулей (plugin), так чтобы клиенты могли загружать их собственный код, или использование механизма правил, который позволяет выполнять настройки через интерфейс.

Комментарий Маркуса:

В этом, конечно, нет ничего нового. Microsoft Dynamics CRM является прекрасным примером приложения, в котором доступны подобные уровни настройки.

Можно также предоставить клиентам способы расширения приложения без использования пользовательского кода. Пользователям приложения Survey может понадобиться получить дополнительные сведения о респонденте опроса, которые стандартное приложение не собирает. Это означает, что пользователи должны иметь механизм настройки пользовательского интерфейса для сбора данных и способ расширения схемы хранения данных для включения новых данных.

Комментарий Бхарата:

Разрешение клиентам загружать собственный код увеличивает риск сбоя приложения, потому что снижается уровень контроля выполняемого приложением кода. Поэтому многие SaaS поставщики накладывают ограничение на использование пользовательского. В большинстве случаев это просто запрещено. Разрешение клиентам загружать код или скрипты также увеличивает риски, связанные с безопасностью приложения.

Архитектура слоя данных для мультитенантного приложения

Архитектура приложения должна обеспечить конфиденциальность данных клиента по отношению к другим клиентам. В приложение может также потребоваться поддержка пользовательских хранилищ данных.

Защита данных от других тенантов

Риск случайного или злонамеренного раскрытия данных в мультитенантной модели выше. Трудно убедить клиента, что его личные данные в безопасности, если клиент знает, что приложение физически используется совместно с другими клиентами. Однако качественная архитектура приложения, которая логически изолирует данные каждого клиента, должна обеспечить требуемый уровень защиты. Подобная архитектура может использовать следующие подходы:

- схемы баз данных, где таблицы каждого клиента находятся в отдельной схеме;
- средства безопасности базы данных, которые позволяют использовать механизмы контроля доступа на уровне базы данных;
- партиционирование для изоляции данных клиента;
- сочетание этих подходов.

Описанные выше подходы применяются как к хранилищу Windows Azure, так и к SQL Azure. Однако они сопряжены с различными моделями тарификации. Для хранилища Windows Azure счет выставляется по объему хранимых данных и по количеству транзакций к хранилищу, так что затраты не зависят от количества отдельных учетных записей или контейнеров хранилища. Счет за использование SQL Azure зависит от количества имеющихся баз данных, поэтому с точки зрения затрат оптимальным является использование одной базы для максимально возможного количества клиентов (тенантов).

Расширяемость архитектуры данных

Есть несколько способов построения хранилища данных таким образом, чтобы клиентам было разрешено расширять модель данных для включения собственных пользовательских атрибутов. Существует несколько подходов:

- подход, при котором каждый клиент имеет отдельную схему;
- подход, при котором предоставляется набор предопределенных пользовательских столбцов;
- подход с более гибкими схемами, которые позволяют клиенту добавлять в таблицу произвольное число пользовательских полей.

При использовании SQL Azure основной сложностью является необходимость работать в рамках фиксированной схемы данных. При использовании табличного хранилища Windows Azure сложность, наоборот, обусловлена нефиксированной схемой хранилища. Табличное хранилище Windows Azure позволяет записям в одной таблице иметь совершенно разную структуру, что обеспечивает большую гибкость за счет большей сложности кода.

Комментарий Яны:

Microsoft SharePoint® является примером приложения с базой данных с фиксированной схемой, что обеспечивает чрезвычайную гибкость представления данных.

Вносимые пользователями изменения в модель данных (пользовательские расширения) не должны требовать изменения кода приложения. Для того чтобы уровень бизнес-логики и уровень представления, имели возможность взаимодействовать с расширенной моделью данных, необходимо создать набор файлов конфигурации, которые описывают расширения или код, способный динамически отслеживать внесенные в схему изменения.

Масштабируемость архитектуры данных

Горизонтальное партиционирование позволяет масштабировать хранилище данных. Для достижения масштабируемости SQL Azure можно перенести данные отдельных клиентов в новый экземпляр SQL Azure.

Примечание.

Горизонтальное партиционирование данных, известное также как шардирование (sharding), предполагает перенос некоторых записей таблицы в новую таблицу. Вертикальное партиционирование данных подразумевает помещение некоторых полей из каждой строки в другую таблицу. Обсуждение стратегий партиционирование в SQL Azure можно найти в блоге *Построение масштабируемых решений базы данных с использованием SQL Azure* — **http://tinyurl.com/247zm33**.

Ключевым фактором, определяющим масштабируемость табличного хранилища Windows Azure, является выбор ключа партиции. Чтобы избежать просмотра (сканирования) нескольких партиций, все запросы должны включать ключ партиции. Дополнительные сведения см. в разделе «Этап 2. Автоматизация развертывания и использование хранилища Windows Azure» *руководства по архитектуре Windows Azure, часть 1, «*Миграция приложений в облако*».*

Финансовые вопросы

Модель тарификации и стоимости может повлиять на выбор между выделенной или мультитенантной моделью.

Выставление счетов клиентам

Для приложения, развертываемого в Windows Azure, выставляется ежемесячный счет для оплаты, использованных ресурсов (трафик, хранилища, транзакции и т. д.). В свою очередь вы так же выставляете счет за услуги своим клиентам.

Одним из подходов к тарификации является использование плана оплаты по фактическому использованию. При таком подходе необходимо отслеживать ресурсы, затраченные на каждого клиента, рассчитывать стоимость этих ресурсов и добавлять маржу для получения прибыли. Для выделенного приложения можно создать отдельную учетную запись Windows Azure и определить итоговую сумму для каждого клиента.

Однако для выделенного экземпляра, выполняющегося в отдельной учетной записи Windows Azure, некоторые затраты будут зафиксированы, т.е. их нельзя будет избежать. Например, оплата за использование вычислительных ресурсов 24x7 и базы SQL Azure может сделать начальную стоимость вашего сервиса слишком высокой для небольших компаний. С мультитенантной архитектурой фиксированные затраты можно разделить между клиентами, но затраты на одного пользователя рассчитать не так просто, и для определения объемов ресурсов, потребляемых каждым клиентов, потребуется в приложение добавить дополнительный код. Более того, клиенты захотят каким-то образом отслеживать свои затраты, поэтому вычисление затрат необходимо сделать максимально прозрачным, а так же реализовать механизм доступа клиентов к этим данным.

Трудно заранее точно определить нагрузку и ресурсы, которые потребуются для конкретного подписчика. Для приложения Surveys компания Tailspin не может спрогнозировать, какое количество опросов создаст подписчик или какое количество респондентов будет у того или иного опроса. Если Tailspin принимает модель тарификации, которая предлагает службу Surveys за фиксированную месячную плату, то прибыльность будет варьироваться между подписчиками (и даже может быть в некоторых случаях отрицательной). Сделав Surveys мультитенантным приложением, Tailspin может сгладить различия в шаблонах использования разных подписчиков, что дает возможность легче прогнозировать расходы и доходы и уменьшить риск убытков. Чем больше клиентов, тем легче на основе статистических данных определить шаблоны использования службы.

С точки зрения клиента начисление фиксированной платы за услугу означает, что клиент заранее точно знает, какие будут расходы в следующем платежном периоде. Это также означает, что система тарификации становится намного проще. Некоторые расходы, такие как расходы, связанные с хранилищем и транзакциями, будут переменными и будут завесить от количества имеющихся клиентов и от того, как они используют службу. Другие расходы, такие как расходы на вычисленные ресурсы или стоимость экземпляра SQL Azure, будут четко зафиксированы. Для того, чтобы получить итоговую прибыль, необходимо продать достаточное число подписок, которые покроют фиксированные и переменные затраты.

Если ваша клиентская база включает как крупные, так и небольшие компании, то стандартная абонентская плата может быть слишком высокой для привлечения некрупных клиентов. В этом случае потребуется видоизменение второго подхода и предложение ряда пакетов для разных уровней использования. Например, в приложении Surveys компания Tailspin может предложить ограниченный пакет услуг с более низкими ежемесячными ценами, чем стандартный пакет. В таком пакете услуг может быть ограничено количество опросов, которые могут быть созданы в течение определённого промежутка времени, либо количество респондентов, которые могут пройти опрос.

Если клиенты имеют возможность комбинировать различные функции и/или выбирать квоты, то продукт должен изначально проектироваться с учетом этих возможностей, т.к. подобное требования влияет на все уровни приложения: представление, бизнес-логика и данные. Также необходимо провести анализ рынка и определить ожидаемый спрос на различные пакеты по разным ценам, это поможет оценить потенциальную клиентскую базу и ожидаемые доходы и расходы.

Управление затратами приложения

Для приложения Windows Azure текущие затраты можно разделить на фиксированные и переменные. Например, если тариф для вычислительных ресурсов равен 0,12 доллара в час (Small экземпляр), то стоимость круглосуточного выполнения двух вычислительных узлов (для избыточности и отказоустойчивости) за один месяц — это фиксированная сумма, примерно равная 180 долларам. Если это мультитенантное приложение, то все клиенты разделяют между собой эти расходы. Чтобы уменьшить стоимость на одного клиента, попытайтесь набрать максимальное возможное количество пользователей для совместного использования приложения без отрицательного воздействия на производительность приложения. Необходимо также проанализировать характеристики производительности приложения, чтобы определить наилучший подход при возрастании нагрузки на приложение:

- повышать текущие мощности, используя более производительные вычислительные ресурсы;
- или применить масштабирование путем добавления дополнительных экземпляров.

Переменные расходы будут зависеть от количества имеющихся клиентов или от того, как эти клиенты используют приложение. В приложение Tailspin Surveys количество опросов и количество респондентов на каждый опрос будет в значительной степени определять ежемесячные затраты на хранение и транзакции. От выделенной или мультитенантной архитектуры приложения стоимость на одного клиента не зависит. Независимо от модели для конкретного клиента требуется одинаковый объем хранилища и использование одинакового количества вычислительных циклов. Для управления этими затратами необходимо обеспечить максимально эффективное использование этих ресурсов приложением.

Примечание.

Дополнительные сведения об оценке затрат Windows Azure см. в главе 4, «Сколько это будет стоить», в книге *Руководство по архитектуре Windows Azure», часть 1, «*Миграция приложений в облако*».*
Дополнительные сведения о стоимости хранения см. в блоге группы разработчиков хранилища Windows Azure: **http://blogs.msdn.com/b/windowsazurestorage/archive/2010/07/09/understanding-windows-azure-storage-billingbandwidth-transactions-and-capacity.aspx**.

Глава 4. Доступ к приложению Surveys

В этой главе рассматриваются задачи, с которыми столкнулась компания Tailspin при разработке и реализации некоторых компонентов пользовательского интерфейса для приложения Surveys. В главе в основном рассматриваются способы взаимодействия клиентов с приложением. Глава начинается с описания выбора URL-адресов для доступа к приложению и использования в приложении протокола SSL.

В Tailspin планируют предложить подписки на приложение Surveys самым разным пользователям: от крупных предприятий до частных лиц. Эти подписчики могут находиться в любой точке мира, причем опросы им, возможно, потребуется проводить в других регионах. В этой главе описывается как компания Tailspin проектировала приложения Surveys с учетом географического положения. В главе также рассматриваются способы проверки подлинности и авторизации подписчиков и возможные способы использования в приложении сети доставки содержимого (CDN) Windows Azure для улучшения работы пользователей.

DNS-имена, сертификаты и SSL в приложении Surveys

В главе 2 «Сценарий Tailspin» были выделены три основные группы пользователей приложения Surveys. В этом разделе описывается способ использования компанией Tailspin записей службы доменных имен (DNS) для управления URL-адресами, которые каждая группа может использовать для доступа к службе, а также как Tailspin планирует использовать протокол SSL для защиты приложения Surveys.

Веб-роли в приложении Surveys

Компания Tailspin использует веб-роли для реализации пользовательского интерфейса приложения Surveys. В этом разделе описываются разработка и реализация этих веб-ролей.

Цели и требования

Три различные группы пользователей, обращающихся к приложению Surveys:

- администраторы в Tailspin, которые будут управлять приложением;
- подписчики, которые будут создавать свои собственные пользовательские опросы и анализировать результаты;
- и пользователи, которые будут отвечать на опросы.

В данное время первые две группы составляют очень небольшую часть от общего числа пользователей. Большинство пользователей — это люди, которые заполняют опросы. Потенциально опрос может проводиться среди сотен тысяч респондентов, тогда как подписчик может создавать новый опрос лишь один раз в две недели. Кроме того, число респондентов может внезапно вырасти на очень короткий период, когда подписчики начинают новые опросы. В дополнение к различным требованиям масштабируемости, которые являются результатом двух профилей использования, могут также появляться другие требования, например безопасность.

Есть три различные группы пользователей, которые будут использовать приложение Surveys.

К подписчикам и администраторам применяются элементы управления проверки подлинности и авторизации, описанные далее в этой главе. Ключевым требованием для приложения является возможность защиты доступа к опросу и результатам опросов от несанкционированного доступа, и для решения этой задачи приложение будет использовать проверку подлинности на основе утверждений. Хотя некоторые опросы могут быть предназначены для закрытой группы пользователей, для которых требуется проверка подлинности в той или иной форме, многие опросы могут быть открыты для широкой публики и доступны без какой-либо формы входа. Кроме того, при доступе к приложению подписчиков и администраторов будет использоваться протокол HTTPS (для защиты данных, передаваемых между приложением и клиентом). Для общедоступных опросов не требуется протокол HTTPS, и это позволяет применять настраиваемые URL-адреса для доступа к этим опросам с использованием пользовательских записей DNS CNAME.

Подписчики и респонденты опроса могут находиться в различных географических точках. Например, подписчик может быть в США, а маркетинговое исследование может проводиться в Европе. Компания Tailspin может свести к минимуму задержку для респондентов опроса, позволяя подписчикам размещать опросы в центре обработки данных, который находится в соответствующем географическом регионе. Однако анализировать результаты таких опросов подписчики, возможно, предпочтут в своем регионе.

Технологическая платформа Windows Azure позволяет развертывать экземпляры роли в центрах обработки данных в разных географических точках.

Обзор решения

Чтобы облегчить соответствие приложения Surveys указанным выше требованиям, разработчики компании Tailspin решили использовать отдельные веб-роли. Одна веб-роль будет содержать функции подписчика и административные функции, тогда как отдельная веб-роль будет размещать сами опросы. С точки зрения масштабирования Tailspin сможет настроить каждую веб-роль независимо от другой.

Комментарий Яны:

Tailspin может размещать веб-роли подписчика и опроса в различных географических точках. Дополнительные сведения об этом представлены в разделе «Географическое местоположение» далее в этой главе.

Наличие нескольких веб-ролей в одной размещенной службе влияет на выбор URL-адресов, которые могут быть использованы для доступа к приложению. Windows Azure назначает одно DNS-имя (например, tailspin.cloudapp.net) размещенной службе; это означает, что различные веб-сайты в рамках этой размещенной службы должны иметь разные порты. Например, два веб-сайта в размещенной службе компании Tailspin могут иметь адреса, перечисленные в следующей таблице.

Сайт A	Сайт B
http://tailspin.cloudapp.net:80	http://tailspin.cloudapp.net:81

Комментарий По:

Помните, что вы можете использовать записи DNS **CNAME** для сопоставления пользовательских доменных имен с DNS-именами, предоставляемыми Windows Azure по умолчанию. Записи DNS **A** вы можете также использовать для сопоставления пользовательского имени домена со службой, но неизменность IP адреса гарантируется только для того же развертывания, т.е. IP адрес может измениться, если приложение будет развернуто заново или обновлено. Поэтому вам может потребоваться так же изменить A запись.

Из-за определенных соображений безопасности приложения Surveys в компании Tailspin решили использовать следующие URL-адреса:

- https://tailspin.cloudapp.net
- http://tailspin.cloudapp.net

https://tailspin.cloudapp.net

Адрес используется для веб-роли, которая размещает административные функции для подписчиков и компании Tailspin, этот HTTPS-адрес использует порт по умолчанию 443. Поскольку для этого сайта применяется SSL-сертификат, то можно сопоставить только одно пользовательское DNS-имя. Для доступа к этому сайту Tailspin планирует использовать такой адрес, как https://surveys.tailspin.com.

http://tailspin.cloudapp.net

Адрес используется для доступа к веб-роли, размещающей опросы, этот HTTP-адрес использует порт по умолчанию - 80. Поскольку SSL-сертификат не используется, то с этим сайтом можно сопоставить несколько DNS-имен. Для доступа к опросам Tailspin настроит DNS-имя по умолчанию, такое как http://surveys.tailspin.com, но отдельные клиенты для сопоставления с сайтом могут создавать свои собственные записи CNAME, например http://surveys.adatum.com, http://surveys.tenant2.org или http://survey.tenant3.co.de.

Можно также создать отдельные размещенные службы для отдельных клиентов, что позволит подписчикам для их опросов применять пользовательские URL-адреса. Однако этот подход может осложнить процесс провиженинга, и для небольших подписчиков это будет экономически невыгодно. Tailspin планирует масштабировать приложение путем добавления дополнительных экземпляров роли в размещенную службу.

Комментарий По:

Компании Tailspin необходимо опубликовать руководства для подписчиков, где будет описана процедура создания записей CNAME в параметрах DNS.

Реализация

Для реализации двух различных веб-сайтов в одной размещенной службе разработчики Tailspin определили в решении две веб-роли. Первый веб-сайт TailSpin.Web представляет собой проект MVC 3.0, который выполняет административные задачи. Для этого веб-сайта требуется проверка подлинности и авторизация, и пользователи обращаются к нему с помощью протокола HTTPS. Второй веб-сайт, Tailspin.Web.Survey.Public, является проектом MVC 3.0 для работы с пользователями, заполняющих опросные листы. Это открытый вебсайт, и пользователи обращаются к нему по протоколу HTTP.

В следующем примере коде показано содержимого файла ServiceDefinition.csdef и определение двух вебролей:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="TailSpin.Cloud" xmlns=…>
  <WebRole name="TailSpin.Web" enableNativeCodeExecution="true">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="HttpsIn" endpointName="HttpsIn" />
          <Binding name="HttpIn" endpointName="HttpIn" />
</Bindings>
      </Site>
    </Sites>
    <ConfigurationSettings>
      <Setting name="DataConnectionString" />
      <Setting name="PublicSurveyWebsiteUrl" />
    </ConfigurationSettings>
    <Certificates>
      <Certificate name="tailspinweb" storeLocation="LocalMachine"
storeName="My" />     </Certificates>
    <Endpoints>
      <InputEndpoint name="HttpsIn" protocol="https"
port="443" certificate="tailspinweb" />
      <InputEndpoint name="HttpIn" protocol="http"
port="80" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
    <LocalResources>
    </LocalResources>
  </WebRole>
  <WebRole name="TailSpin.Web.Survey.Public">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="HttpIn" endpointName="HttpIn" />
          <Binding name="HttpsIn" endpointName="HttpsIn" />
</Bindings>
      </Site>
    </Sites>
    <ConfigurationSettings>
      <Setting name="DataConnectionString" />
    </ConfigurationSettings>
    <Certificates>
      <Certificate name="tailspinpublicweb"
        storeLocation="LocalMachine" storeName="My" />
```

```
    </Certificates>
    <Endpoints>
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
      <InputEndpoint name="HttpsIn" protocol="https"
port="443" certificate="tailspinpublicweb" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
  </WebRole>
</ServiceDefinition>
```

Примечание.

Этот лишь пример файла ServiceDefinition.csdef; он не полностью соответствует файлу в загружаемом решении, где используются другие имена для сертификатов.

Хотя подписчики могут обращаться к веб-сайту подписчиков (определенному в веб-роли TailSpin.Web) только с помощью протокола HTTPS, компания Tailspin также определила конечную точку HTTP. Модуль перезаписи URL-адресов для служб IIS будет выполнять переадресацию трафика к конечной точке HTTPS на порт 443. Определив конечную точку HTTP сейчас, в будущем компания Tailspin сможет добавлять на веб-сайт содержимое не HTTPS, не удаляя приложение Surveys. Общедоступный веб-сайт также использует модуль перезаписи URL-адресов для пересылки трафика HTTPS в конечную точку HTTP на порт 80 по тем же причинам.

Комментарий По:

Используйте модуль перезаписи URL-адресов для перенаправления трафика с неиспользуемых конечных точек. Это подготовит ваши приложения к требованиям завтрашнего дня, если позже вы решите использовать эти конечные точки.

Примечание.

Помните, что вы можете использовать другие SSL-сертификаты, если тестируете приложения с помощью локального *эмулятора вычислений*. Прежде чем публиковать приложение на Windows Azure, необходимо убедиться в том, что в файлах конфигурации указаны правильные сертификаты.

Дополнительные сведения об управлении развертыванием содержатся в главе 7, «Управление жизненным циклом приложений», книги *«Руководство по архитектуре Windows Azure», часть 1, «*Миграция приложений в облако*»*.

Помимо двух проектов веб-ролей решение содержит проект рабочей роли и библиотеку TailSpin.Web.Survey.Shared с общим кодом для веб-ролей и рабочей роли. Этот общий код включает классы модели и код доступа к данным.

Географическое местоположение

Windows Azure позволяет выбрать географическое местоположение для размещаемых в Windows Azure служб. Это позволяет разместить приложение ближе к пользователям. В этом разделе показан, как компания Tailspin использует эту функцию в приложении Surveys.

Цели и требования

Компания Tailspin дает возможность подписчикам на услугу Surveys указать в каком географическом регионе они хотели бы использовать экземпляр приложения Surveys. Например, клиент, расположенный в США, может выбрать услугу на территории США, а европейские клиенты — на территории Европы. Однако подписчик может захотеть провести опрос не в том географическом регионе, где он сам находится. На рисунке 1 показано, как подписчик, расположенный на территории США, может захотеть провести опрос в Европе:

Приложение Surveys — это услуга, учитывающая географическое расположение.

Комментарий По:

Проверить текущее состояние любого центра обработки данных Windows Azure вы можете здесь: http://www.microsoft.com/windowsazure/support/status/servicedashboard.aspx.

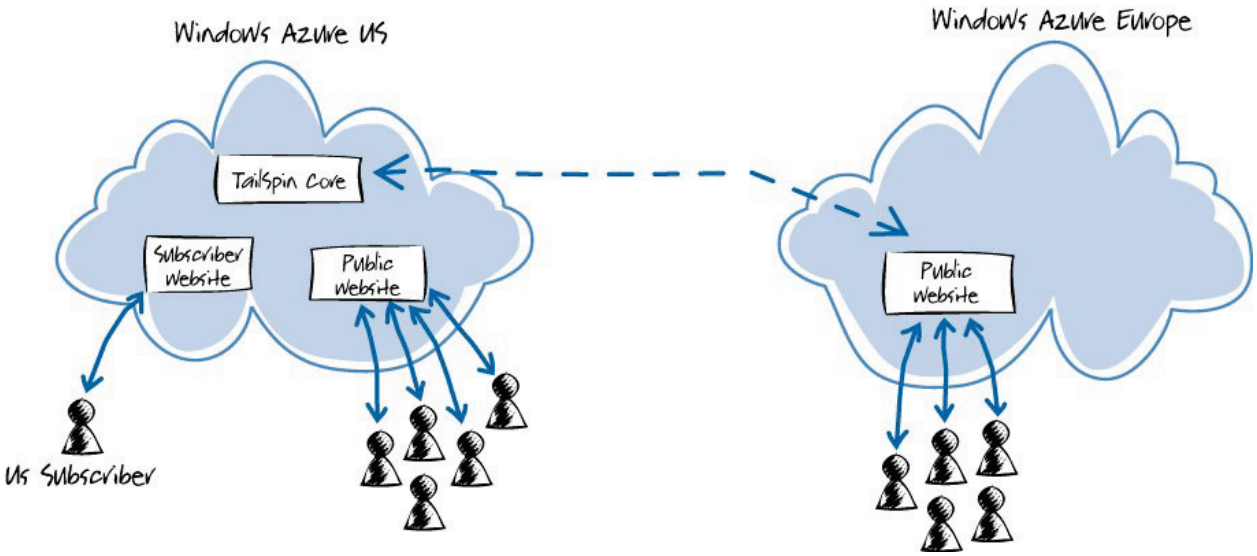


Рис. 1 Подписчик, расположенный в США, проводит опрос в Европе

Обзор решения

Размещение опроса в веб-роли в другом регионе не означает само по себе, что время отклика веб-сайта будет меньше. Для вывода опросного листа приложение должно извлечь параметры опроса из хранилища. Приложение также должно сохранить результаты опроса в хранилище. Если в примере, показанном на рисунке 1, хранилище приложения находится в центре обработки данных в США, это не дает преимуществ европейским клиентам, обращающимся к веб-сайту, размещенному в европейском центре обработки данных. На рисунке 2 показано выбранное компанией Tailspin решение указанной проблемы.

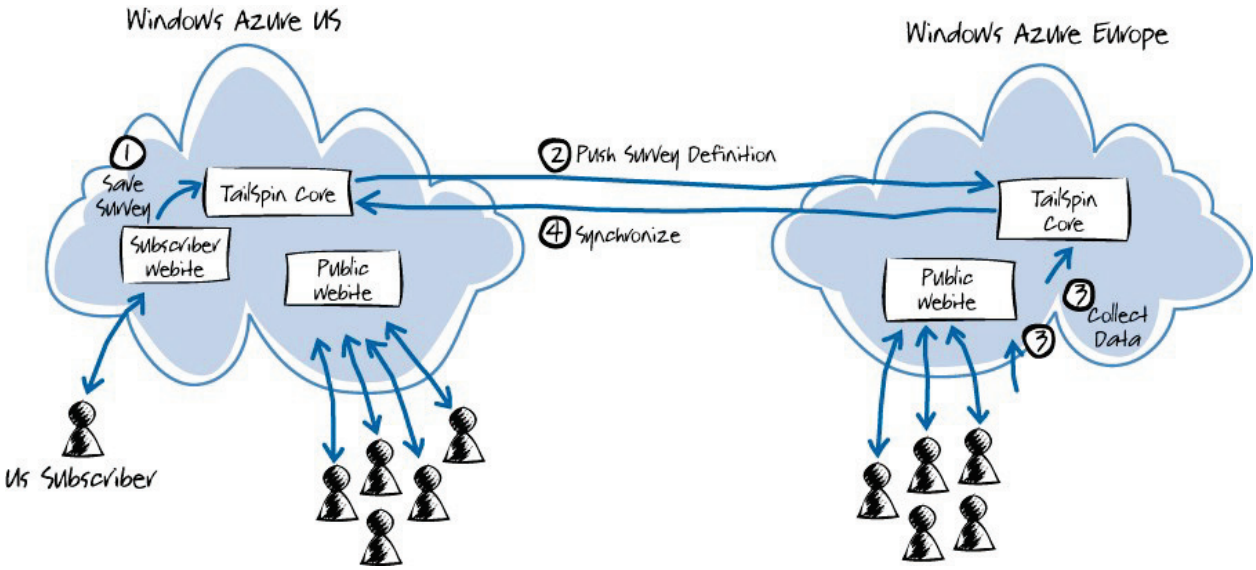


Рис. 2

Размещение опроса в другом географическом регионе

Ниже описаны шаги, показанные на рисунке 2.

1. Подписчик разрабатывает опрос, и приложение сохраняет опрос в хранилище, расположенном в центре обработки данных в США.
2. Приложение Surveys принудительно отправляет параметры опроса другому экземпляру приложения в европейском центре обработки данных. Это нужно сделать только один раз.
3. Респонденты в Европе заполняют опросный лист, и приложение сохраняет данные в хранилище в европейском центре обработки данных.
4. Приложение передает результаты опроса на хранение в центр обработки данных в США, где эти результаты доступны подписчику для анализа.

В некоторых сценариях может быть разумно предварительно обработать или суммировать данные в регионе, где они собраны, и передать назад только сводные данные для снижения затрат на передачу.
Для приложения Surveys компания Tailspin решила передавать все данные в регион, где находится подписчик. Это упрощает внедрение, помогает оптимизировать функции разбиения на страницы и гарантирует, что каждый ответ будет передаваться между центрами обработки данных только один раз.

Комментарий Яны:

Подробнее о том, как приложение перемещает данные опроса между центрами обработки данных, мы поговорим в главе 6, «Работа с данными в приложении Surveys.»

Примечание.

При развертывании приложения Windows Azure можно выбрать конкретный регион (центр обработки данных), где вы хотите разместить приложение. Вы можете также задать *группы привязки* и использовать их для создания групп взаимозависимых приложений Windows Azure и совместного хранения учетных записей, что повысит производительность и снизит затраты. Производительность улучшается, потому что Windows Azure размещает членов группы привязки в одном и том же центре обработки данных. Это сокращает затраты, потому что передача данных в одном и том же центре данных не требует затрат (внутренний трафик не тарифицируется). Группы привязки дают только небольшое преимущество над простым выбором региона для размещения служб, так как в Windows Azure автоматически старается оптимизировать расположения служб.

Если вы планируете разместить одну и ту же услугу в разных географических регионах, следует рассмотреть возможность использования диспетчера трафика Windows Azure: с помощью этой функции вы можете регулировать направление трафика Windows Azure к определенным центрам обработки данных, а также задать правила балансировки нагрузки и политики отработки отказов.

Проверка подлинности и авторизация

В этом разделе показано как компания Tailspin реализовала проверку подлинности и авторизацию в приложении Surveys.

Примечание.

Дополнительные сведения об этом сценарии см. в главе 6, «Федеративное удостоверение для нескольких партнеров», книги *Руководство по удостоверению, основанному на утверждениях, и управлению доступом*. Эта книга доступна для загрузки по адресу http://msdn.microsoft.com/ru-ru/library/ff423674.aspx.

Цели и требования

Приложение Tailspin Surveys предназначено для широкого круга клиентов от крупных предприятий до частных лиц. Всем клиентам приложения Surveys потребуется возможность проверки подлинности и авторизации в приложении, но для различных клиентов данная возможность должна быть реализована по-разному. Например, крупное предприятие, скорее всего, будет заинтересовано в интеграции с их существующей инфраструктурой проверки подлинности, менее крупные клиенты могут быть не в состоянии интегрировать свои системы, и им понадобится базовая система безопасности в в самом приложении Surveys, а частное лицо может захотеть повторно использовать существующий идентификатор Windows Live ID или OpenID.

Обзор решения

Компания Tailspin выявила три сценария идентификации, которые должны поддерживаться приложением Surveys.

- Организации могут захотеть интегрировать свою существующую инфраструктуру идентификации и иметь возможность самостоятельно управлять доступом к приложению Surveys, чтобы включить приложение Surveys как часть единого входа (SSO) для своих сотрудников.
- Организации меньшего размера могут потребовать от компании Tailspin встроенную систему идентификации, так как не в состоянии интегрировать свои существующие системы с Tailspin.
- Частные лица и небольшие организации могут пожелать повторно использовать свой существующий идентификатор, например Windows Live ID или OpenID.

Комментарий Бхарата:

Tailspin использует проверку подлинности на основе утверждений, это обеспечивает необходимую гибкость для разнообразной клиентской базы.

Компания Tailspin использует протокол WS-Federation для реализации сценария федерации утверждений. Хотя пример решения Tailspin Surveys использует Windows Identity Foundation (WIF) как поставщика федерации, совместимого с WS-Federation (см. проект TailSpin.SimulatedIssuer в решении), рабочий сценарий развертывания будет использовать «реального» поставщика федерации, например службы управления доступом (Access Control Services, ACS). Дополнительные сведения об использовании служб управления доступом (ACS) в сценарии этого типа см. В «*Руководстве по удостоверению, основанному на утверждениях, и управлению доступом*» по адресу http://msdn.microsoft.com/ru-ru/library/ff423674.aspx.

Следующие три диаграммы иллюстрируют процесс проверки подлинности и авторизации для каждого из этих трех сценариев.

Примечание.

Все три сценария основаны на утверждениях и используют одну и ту же базовую инфраструктуру идентификации. Единственное отличие состоит в источнике начальных утверждений.

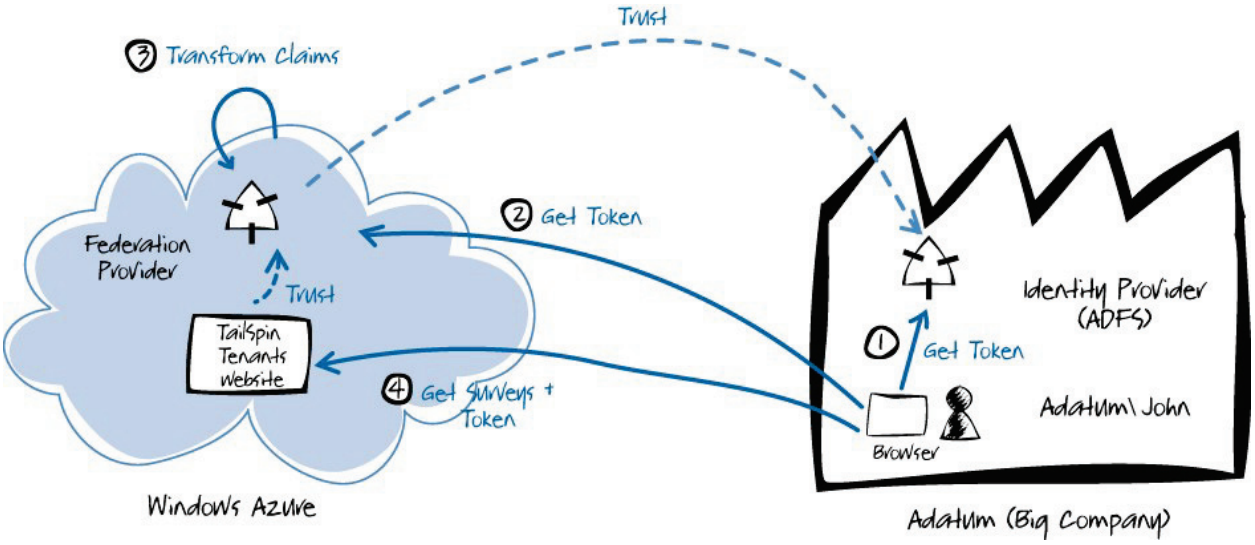


Рис. 3

В сценарии, показанном на рисунке 3, пользователи Adatum, крупного предприятия-подписчика, проходят проверку подлинности у собственного поставщика удостоверений Adatum (шаг 1), в данном случае у службы Active Directory Federation Services (ADFS). После успешной проверки подлинности пользователя Adatum служба ADFS выпускает маркер. Поставщик федерации Tailspin доверяет маркерам, выданным службой ADFS компании Adatum (шаг 2), и при необходимости, прежде чем вернуть маркер пользователю, может преобразовать утверждения, содержащиеся в маркере, в утверждения, признаваемые приложением Tailspin Surveys (шаг 3), Приложение Tailspin Surveys доверяет маркера, выданным поставщиком федерации Tailspin, и использует эти утверждения в маркере для применения правил авторизации (шаг 4). Пользователям в компании Adatum не нужно запоминать специальные учетные данные для доступа к приложению Surveys, а администратор в компании Adatum будет иметь возможность разграничить через ADFS доступ к приложению Surveys.

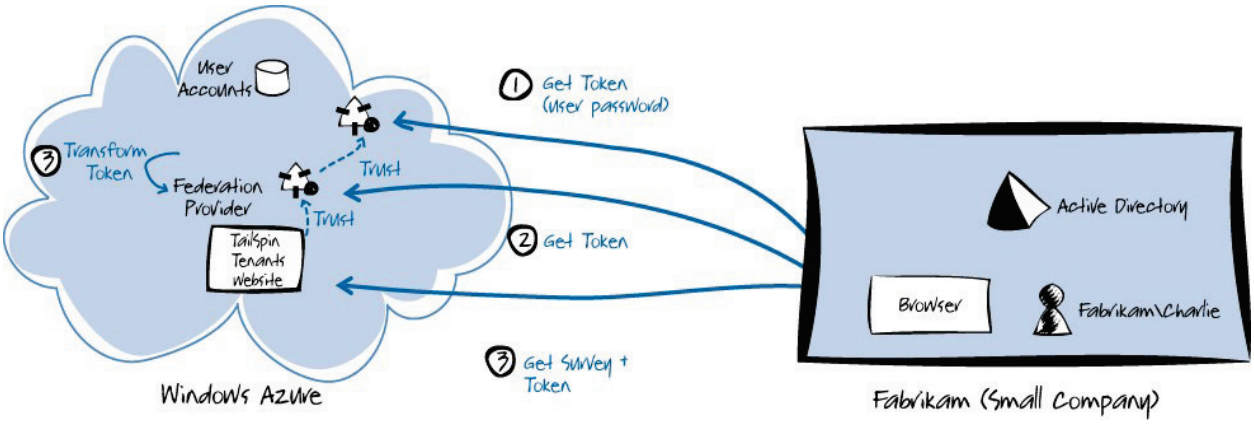


Рис. 4

В сценарии, показанном на рисунке 4, пользователи в небольшой компании Fabrikam проходят проверку подлинности у поставщика удостоверений Tailspin (шаг 1), так как их собственный Active Directory не может выдавать маркеры, понятные поставщику федерации Tailspin. За исключением выбора поставщика удостоверений, используется тот же самый подход, что и для компании Adatum. Недостатком этого подхода для пользователей Fabrikam является то, что они должны помнить отдельный пароль для доступа к приложению Surveys. Компания Tailspin планирует реализовать этот сценарий с помощью провайдера ASP.NET (ASP.NET Membership Provider) и использовать службу маркеров безопасности (Security Token Service, STS), интегрированную с поставщиком ASP.NET.

Примечание.

Чтобы понять, как реализовать этот сценарий, посмотрите проект Starter STS по адресу http://startersts.codeplex.com.

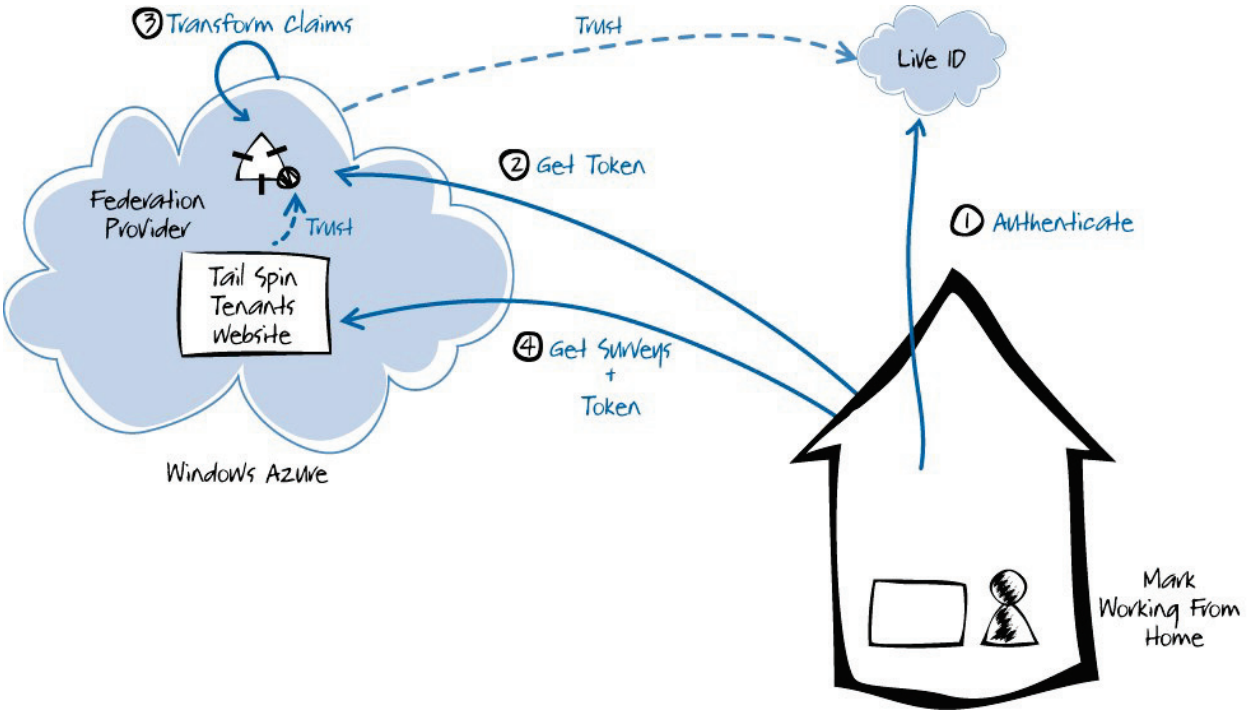


Рис. 5

Процесс для частных пользователей также очень похож на предыдущие. В сценарии, показанном на рисунке 5, поставщик федерации Tailspin настроен так, что доверяет маркеру, выданным сторонними поставщиками, например Windows Live ID или OpenID (шаг 1). Когда пользователь пытается получить доступ к своим опросам, приложение перенаправляет их для проверки подлинности к внешнему поставщику удостоверений.

Компания Tailspin планирует создать службу перевода протоколов STS для преобразования различных протоколов, используемых сторонними поставщиками, в протокол, используемый приложением Surveys.

Примечание.

Чтобы понять, как реализовать этот сценарий, посмотрите проект protocol-bridge-claims-provider по адресу http://github.com/southworks/protocol-bridge-claims-provider.

Реализация

Теперь самое время изучить более подробно код проверки подлинности и авторизации. В ходе изучения этого раздела может потребоваться среда разработки Microsoft Visual Studio и исходные коды приложения Tailspin Surveys со страницы http://wag.codeplex.com/. Диаграмма на рисунке 6 поможет понять весь процесс целиком.

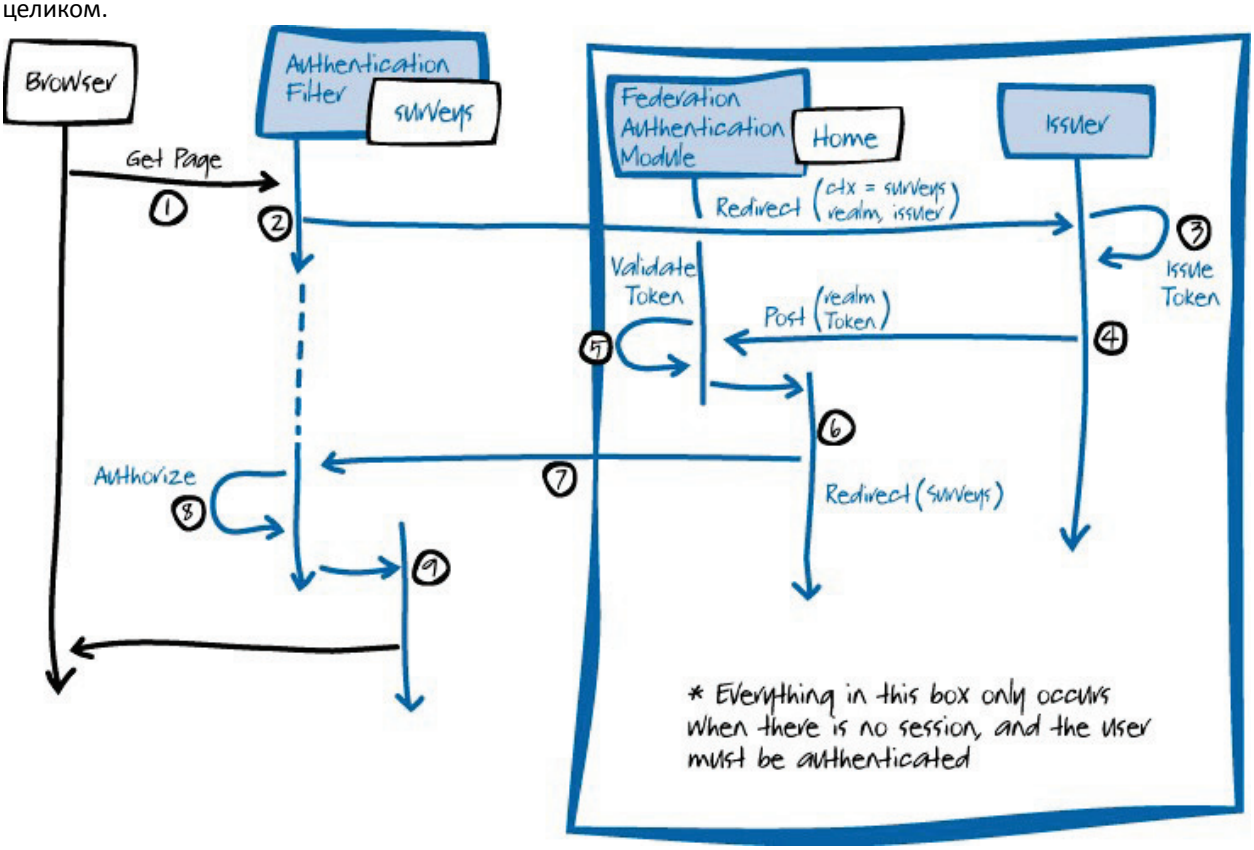


Рис. 6

Схема последовательностей федерации с несколькими участниками

Комментарий Яны:

Последовательность, показанная на этой диаграмме, относится ко всем трем сценариям. В контексте этой диаграммы поставщиком маркера является поставщик федерации Tailspin, поэтому шаг 3 включает перенаправление к другому поставщику для проверки подлинности.

Для большей ясности на рисунке 6 показана «логическая» последовательность, не «физическая». Если на диаграмме есть стрелка с меткой **Redirect**, то обратно в браузер фактически посылается ответ **redirect** и браузер посылает запрос по адресу, указанному в сообщении **redirect**.

Ниже описаны шаги, показанные на рисунке 6.

1. Этот процесс запускается, когда пользователь, не прошедший проверку подлинности, посылает запрос на защищенный ресурс, например страницу adatum/surveys. Это инициирует вызовов метода в классе **SurveysController**.
2. Атрибут **AuthenticateAndAuthorizeAttribute**, который реализует интерфейс MVC **IAuthorizationFilter**, применяется к этому классу контроллера. Так как подлинность пользователя еще не проверена, пользователь будет перенаправлен к поставщику федерации Tailspin по адресу https://localhost/TailSpin.SimulatedIssuer со следующими значениями параметров строки запроса:

wa. Wsignin1.0 *wtrealm*. https://tailspin.com *wctx*. https://127.0.0.1:444/survey/adatum *whr*. http://adatum/trust *wreply*. https://127.0.0.1:444/federationresult

В следующем примере кода показан метод **AuthenticateUser** класса **AuthenticateAndAuthorizeAttribute**, который создает строку запроса.

C#

```
private static void AuthenticateUser(
                        AuthorizationContext context)
{
    var tenantName =
        (string) context.RouteData.Values["tenant"];

    if (!string.IsNullOrEmpty(tenantName))
    {
        var returnUrl =
            GetReturnUrl(context.RequestContext);

// Пользователь не прошел проверку подлинности и входит
// впервые.
        var fam =
FederatedAuthentication             .WSFederationAuthenticationM
odule;
var signIn = new SignInRequestMessage
(new Uri(fam.Issuer), fam.Realm)
        {
            Context = returnUrl.ToString(),
            HomeRealm =
                RetrieveHomeRealmForTenant(tenantName)
        };

// В среде Windows Azure
// создать параметр wreply для запроса SignIn,
// который отражает реальный адрес
// приложения.
        HttpRequest request = HttpContext.Current.Request;
Uri requestUrl = request.Url;

        StringBuilder wreply = new StringBuilder();
        wreply.Append(requestUrl.Scheme);
// HTTP или HTTPS         wreply.Append("://");
wreply.Append(request.Headers["Host"] ??
requestUrl.Authority);
        wreply.Append(request.ApplicationPath);

        if (!request.ApplicationPath.EndsWith("/"))
        {
            wreply.Append("/");
        }

        wreply.Append("FederationResult");

        signIn.Reply = wreply.ToString();

        context.Result = new
            RedirectResult(signIn.WriteQueryString());
    }
}
```

3. Поставщик маркера, в данном случае эмулируемый поставщик маркера Tailspin, проверяет подлинность пользователя и создает маркер с представленными утверждениями. В этом сценарии компании Tailspin поставщик федерации Tailspin использует значение параметра *whr*, чтобы делегировать проверку подлинности другому поставщику маркера (в этом примере — поставщику маркера Adatum). При необходимости поставщик федерации Tailspin может преобразовать утверждения, которые он получает от поставщика маркера, в утверждения, понятные приложению Tailspin Surveys. Следующий код из класса **FederationSecurityTokenService** показывает, как эмулируемые поставщик маркеров Tailspin преобразует утверждения **Group** в маркере от поставщика Adatum.

C#

```
switch (issuer.ToUpperInvariant())
{
    case "ADATUM":
        var adatumClaimTypesToCopy = new[]
        {
            WSIdentityConstants.ClaimTypes.Name
        };
        CopyClaims(input, adatumClaimTypesToCopy,
output);
        TransformClaims(input,
            AllOrganizations.ClaimTypes.Group,
            Adatum.Groups.MarketingManagers,
            ClaimTypes.Role,
            TailSpin.Roles.SurveyAdministrator, output);
output.Claims.Add(
            new Claim(TailSpin.ClaimTypes.Tenant,
            Adatum.OrganizationName));         break;
case "FABRIKAM":
        …     default:
        throw new InvalidOperationException(
            "Issuer not trusted."); }
```

Примечание.

Образец кода эмулирующего поставщика маркера для Tailspin содержит некоторые жестко заданные имена (например, Adatum и Fabrikam) и некоторые жестко заданные типы утверждений. В реальном поставщике маркера эти значения были бы извлечены из файла конфигурации или хранилища.

4. Поставщик федерации Tailspin затем отправляет маркер и значение параметра *wctx* (https://127.0.0.1:444/survey/adatum) обратно по адресу, указанному в параметре *wreply* (https://127.0.0.1:444/federationresult). Этот адрес является еще одним контроллером MVC (у которого нет атрибута **AuthenticateAndAuthorizeAttribute**). В следующем примере кода показан метод **FederationResult** контроллера **ClaimsAuthenticationController**.

C#

```
 [RequireHttps]
public class ClaimsAuthenticationController: Controller
{
    [ValidateInput(false)]
    [HttpPost]
    public ActionResult FederationResult()
    {
        var fam =
FederatedAuthentication             .WSFederationAuthenticationM
odule;         if
(fam.CanReadSignInResponse( System.Web.HttpContext.Current.Reque
st, true))
        {
            string returnUrl = GetReturnUrlFromCtx();

            return this.Redirect(returnUrl);
        }
        return this.RedirectToAction(
            "Index", "OnBoarding");
    }
```

5. **WSFederationAuthenticationModule** проверяет маркер, вызывая метод **CanReadSignInResponse**.
6. Контроллер **ClaimsAuthenticationController** извлекает значение исходного параметра *wctx* и инициирует перенаправление на этот адрес.
7. Теперь запрос на страницу adatum/surveys идет через фильтр **AuthenticateAndAuthorizeAttribute**, т.к. подлинность пользователя уже была установлена. В следующем примере кода показано как фильтр проверяет подтверждена ли подлинность пользователя.

C#

```
public void OnAuthorization(
    AuthorizationContext filterContext)
{
    if (!filterContext.HttpContext.User
        .Identity.IsAuthenticated)
    {
        AuthenticateUser(filterContext);
    }
    else
    {
        this.AuthorizeUser(filterContext);
    }
}
```

8. Затем фильтр **AuthenticateAndAuthorizeAttribute** применяет все правила авторизации. В приложении Tailspin Surveys метод **AuthorizeUser** проверяет, что пользователь является членом одной из указанных ролей, где атрибут **AuthenticateAndAuthorize** применяется к контроллеру MVC, как показано в следующем примере кода.

C#

```
[AuthenticateAndAuthorize(Roles = "Survey
Administrator")]
[RequireHttps]
public class SurveysController: TenantController
{
   …
}
```

9. Теперь выполняется метод контроллера.

Защита маркеров сеансов в Windows Azure

По умолчанию Windows Identity Foundation (WIF), используемая для управления инфраструктурой идентификации, шифрует содержимое куки-файлов, которые она отправляет клиенту, с помощью Windows Data Protection API (DPAPI). Использование DPAPI для шифрования куки-файлов неприемлемо для приложения, которое имеет несколько экземпляров роли, потому что каждый экземпляр роли будет иметь свой ключ и подсистема балансировки нагрузки Windows Azure может направить запрос любому экземпляру. Вы должны использовать механизм шифрования типа RSA, где используются общие ключи. В следующем примере кода показано как приложение Surveys настраивает обработчик маркера безопасности таким образом, чтобы использовалось шифрование RSA.

Комментарий Яны:

Веб-приложению ASP.NET, работающему в веб-ферме, также пришлось бы использовать шифрование с общим ключом вместо DPAPI.

Дополнительные сведения об использовании механизмов DPAPI и шифрования с общим ключом для шифрования параметров конфигурации см. в разделе:

«Как шифровать разделы конфигурации в ASP.NET 2.0 с помощью DPAPI» библиотеки MSDN (http://msdn.microsoft.com/ru-ru/library/ff647398.aspx).

Дополнительные сведения о DPAPI см. в разделе «Защита данных Windows» библиотеки MSDN (http://msdn.microsoft.com/ru-ru/library/ms995355.aspx)».

C#

```
private void OnServiceConfigurationCreated(object sender,
ServiceConfigurationCreatedEventArgs e)
{
    var sessionTransforms = new List<КукиTransform>
(new КукиTransform[]
            {
                new DeflateКукиTransform(),
                  new RsaEncryptionКукиTransform(
                  e.ServiceConfiguration.ServiceCertificate),
                 new RsaSignatureКукиTransform(
                  e.ServiceConfiguration.ServiceCertificate)
            });
    var sessionHandler = new

SessionSecurityTokenHandler(sessionTransforms.AsReadOnly());
     e.ServiceConfiguration.SecurityTokenHandlers.AddOrReplace(
         sessionHandler);
}
```

Метод **Application_Start** в файле Global.asax.cs связывает этот обработчик событий с модулем **FederatedAuthentication**.

Сеть доставки содержимого (CDN)

Сеть доставки содержимого (Content Delivery Network, CDN) Windows Azure позволяет кэшировать большие двоичные объекты (BLOB) в специальных точках по всему миру, что позволяет сделать содержимое доступным для пользователей по каналам с максимальной пропускной способностью и свести к минимуму любые сетевые задержки. Сеть CDN предназначена для кэширования относительно статичных BLOB объектов. Для приложения Surveys разработчики компании Tailspin выявили два сценария, где они могли бы использовать сеть CDN:

- Компания Tailspin планирует создать ряд учебных видеофильмов («Приступая к работе с приложением Surveys», «Проектирование опросов» и «Анализ результатов опроса»).
- Размещение пользовательских изображений и таблиц стилей, передаваемых подписчиками.

В обоих этих сценариях пользователи могут обращаться к содержимому многократно, при этом содержимое обновляется достаточно. Скорее всего, учебное видео будет обновляться только в случае существенного обновления приложения, и компания Tailspin ожидает, что подписчики будут передавать стандартные корпоративные логотипы и таблицы стилей, отражающие их фирменный стиль. Но оба эти сценария будут занимать значительную часть полосы пропускания, отведенной приложению. Для хорошего качества воспроизведения в сети видеофильмы потребуют достаточной полосы пропускания, и каждый запрос на заполнение опроса будет сопровождаться запросом на пользовательский логотип и таблицы стилей. Одним из условий использования сети CDN является то, что содержимое должно находиться в контейнере больших двоичных объектов, настроенном для общего анонимного доступа. Кроме того, в обоих сценариях содержимое доступно для неограниченного доступа.

Сведения о текущих расценках сети CDN доступны на веб-сайте Windows Azure Platform («Платформа Windows Azure») по адресу http://www.microsoft.com/windowsazure/offers/.

Примечание.

Для данных, кэшированных в сети CDN, с вас будет взиматься плата за исходящий трафик. С вас также взимается плата по стандартным ставкам BLOB-хранилища Windows Azure за трафик для передачи данных из BLOB-хранилища в сеть CDN. Следовательно, использовать сеть CDN лучше для относительно статического содержимого. С высоко динамическим содержимым вы могли бы, фактически, платить двойную плату потому, что каждый запрос данных из сети CDN инициировал бы запрос обновленных данных из хранилища больших двоичных объектов. И потребовалось для динамического содержимого специальным образом настраивать времени жизни содержимого (TTL) и заголовки.

Решение

Чтобы использовать сеть CDN с приложением Surveys, компании Tailspin придется внести ряд изменений в приложение. Эти изменения описаны в следующих разделах. В этом разделе описывается планируемое решение; текущая версия приложения Surveys не включает использование сети CDN.

Настройка управления доступом для контейнеров больших двоичных объектов

Любые данные больших двоичных объектов, которые вы хотите разместить в сети CDN, должны располагаться в контейнере с разрешениями, обеспечивающими полный общий доступ на чтение. Этот параметр вы можете использовать, если вы создаете контейнер путем вызова метода **BeginCreate** класса **CloudBlobContainer** или вызова метода **SetPermissions** для существующего контейнера. В следующем коде показан пример того, как задать разрешения для контейнера.

C#

```
protected void SetContainerPermissions(String containerName)
{
    CloudStorageAccount cloudStorageAccount =
        CloudStorageAccount.FromConfigurationSetting(
        "DataConnectionString");
    CloudBlobClient cloudBlobClient =
        cloudStorageAccount.CreateCloudBlobClient();

    CloudBlobContainer cloudBlobContainer =
        new CloudBlobContainer(containerName, cloudBlobClient);

    BlobContainerPermissions blobContainerPermissions =
        new BlobContainerPermissions();
blobContainerPermissions.PublicAccess =
BlobContainerPublicAccessType.Container;
    cloudBlobContainer.SetPermissions(blobContainerPermissions);
}
```

Обратите внимание, что для настройки полного общего доступа должен использоваться тип разрешения **BlobContainerPublicAccessType.Container**.

Настройка сети CDN и хранение содержимого

Сеть CDN настраивается на уровне учетной записи хранилища Windows Azure через портал разработчиков Windows Azure. После включения доставки по сети CDN для учетной записи хранилища любые данные, содержащиеся в общих BLOB-контейнерах, доступны для доставки по сети CDN.

Приложение должно поместить все содержимое, которое должно храниться в CDN, в соответствующие контейнеры больших двоичных объектов. В приложение Surveys мультимедийные файлы, пользовательские изображения и таблицы стилей могут храниться в этих объектах BLOB. Например, если учебное видео объединено в пакет с приложением проигрывателя в виде HTML-файлов и скриптов, все эти взаимосвязанные файлы могут храниться как большие двоичные объекты в одном и том же контейнере.

Примечание.

Будьте осторожны, если скрипты или HTML-файлы содержат относительные пути к другим файлам в том же BLOB-контейнере, потому что в именах путей будет учитываться регистр. Это объясняется тем, что в контейнере больших двоичных объектов нет никакой реальной структуры папок и любые «имена папок» являются лишь частью имени файла в едином, плоском пространстве имен.

Настройка URL-адресов для доступа к содержимому

Windows Azure выделяет URL-адреса для доступа к данным BLOB в зависимости от имени учетной записи и имени контейнера. Например, если бы компания Tailspin создала общедоступный контейнер с именем «video» для размещения учебного видео, можно было бы получить доступ к видео «Приступая к работе с приложением Surveys» прямо в хранилище больших двоичных объектов Windows Azure по адресу http://tailspin.blob.core.windows.net/video/gettingstarted.html. Это предполагает, что страница gettingstarted.html является проигрывателем для мультимедийного содержимого. Сеть CDN обеспечивает доступ к размещенному содержимому с помощью URL-адреса в форме http://<uid>.vo.msecnd.net/, поэтому учебное видео по приложению Surveys будет доступно в CDN по адресу http://<uid>.vo.msecnd.net/video/gettingstarted.html.

На рисунке 7 показана эта взаимосвязь между сетью CDN и хранилищем больших двоичных объектов.

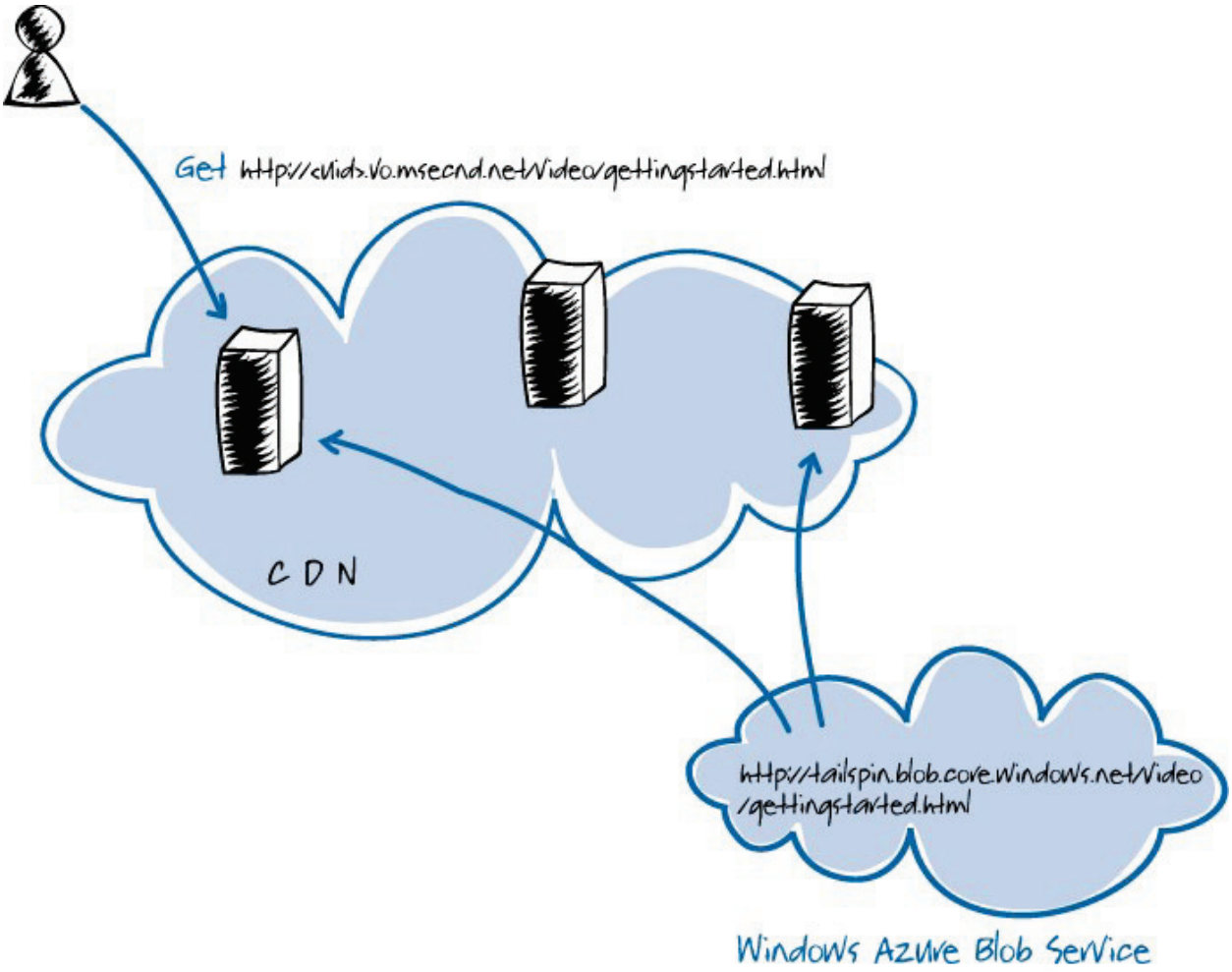


Рис. 7 **Сеть доставки содержимого**

Вы можете настроить запись CNAME в DNS, чтобы связать пользовательский URL-адрес с URL-адресом CDN. Например, Tailspin может создать запись CNAME, чтобы адрес http://files.tailspin.com/video/gettingstarted.html указывал на видео, размещенное в сети CDN. Необходимо убедиться в том, что настройки вашего поставщика DNS обеспечивают эффективное разрешение DNS-имен. Преимуществ в производительности от использования сети CDN может и не быть, если разрешение имени вашего пользовательского URL-адреса предполагает несколько обращений в центр DNS в другом географическом регионе.

Примечание.

Помимо создания записи CNAME для пользовательского имени домена в средстве, используемом для управления записями DNS, необходимо задать пользовательское имя домена в параметрах учетной записи хранилища. Для этого следует использовать раздел «Пользовательские домены» на странице «Summary» на портале разработчиков Windows Azure.

Если пользователь запрашивает содержимое из сети CDN, Windows Azure автоматически перенаправляет этот запрос в ближайшую доступную конечную точку CDN. Если данные больших двоичных объектов обнаружены в этой конечной точке, они возвращаются пользователю. Если данные больших двоичных объектов не найдены в конечной точке, они автоматически извлекаются из хранилища больших двоичных объектов для передачи пользователю и кэшируются в конечной точке для будущих запросов.

Комментарий Яны:

Если данные больших двоичных объектов не найдены в конечной точке, с вас взимается плата по тарифам хранилища Windows Azure, когда сеть CDN извлекает данные из BLOB-хранилища.

Настройка политики кэширования

Все большие двоичные объекты, кэшируемые сетью CDN, имеют срок жизни (TTL), который определяет как долго данные будут оставаться в кэше, пока CDN не обратиться в хранилище больших двоичных объектов за обновлением данных. В политике кэширования, используемой по умолчанию сетью CDN, TTL для кэшируемого содержимого вычисляется по дате последнего изменения содержимого в хранилище больших двоичных объектов. Чем дольше содержимое остается без изменения в хранилище больших двоичных объектов, тем больше срок жизни (TTL). Максимальный срок жизни — 72 часа.

Примечание.

CDN извлекает содержимое из хранилища больших двоичных объектов только в случае, если содержимое не найдено в кэше конечной точки или если оно изменилось в хранилище.

Вы можете также задать TTL с помощью свойства **CacheControl** класса **BlobProperties**. В следующем примере кода показано, как задать TTL, равный двум часам.

```
blob.Properties.CacheControl = "max-age=7200";
```

Дополнительные сведения

Дополнительные сведения об использовании записей CNAME в DNS доступны в публикации «Custom Domain Names in Windows Azure» («Имена пользовательских доменов в Windows Azure») в блоге Стива Маркса: http://blog.smarx.com/posts/custom-domain-names-in-windows-azure.

Дополнительные сведения о проверке подлинности на основе утверждений и о модели авторизации, используемых в приложении Surveys, содержатся в главе 6, «Федеративное удостоверение с несколькими партнерами» книги *Руководство по удостоверению, основанному на утверждениях, и управлению доступом*. Эта книга доступна для загрузки из MSDN по адресу: http://msdn.microsoft.com/ru-ru/library/ff423674.aspx.

Пошаговое руководство о том, как получить сайт ASP.NET с WIF в Windows Azure, см. в разделе «Упражнение: включение федеративной проверки подлинности для приложений ASP.NET в Windows Azure» на Channel9: http://channel9.msdn.com/learn/courses/Azure/IdentityAzure/WIFonWAZLab/Exercise-1-Enabling-FederatedAuthentication-for-ASPNET-applications-in-Windows-Azure/.

Дополнительные сведения о сети CDN см. в разделе «Доставка широкополосного содержимого с помощью сети доставки содержимого (CDN) Windows Azure» в MSDN:
http://msdn.microsoft.com/ru-ru/library/ee795176.aspx

Сведения о приложении, использующем сеть CDN, см. В публикации «EmailTheInternet.com: отправка и получение электронной почты в Windows Azure» в блоге Стива Маркса:
http://blog.smarx.com/posts/emailtheinternet-com-sending-and-receiving-email-in-windows-azure

Эпизод Cloud Cover («Облачный покров»), охватывающий сеть CDN, см. на канале 9:
http://channel9.msdn.com/shows/Cloud+Cover/Cloud-Cover-Episode-4-CDN/.

Глава 5. Создание масштабируемого, мультитенантного приложения для Windows Azure

В этой главе рассматриваются особенности архитектуры и реализации приложения Surveys с точки зрения построения мультитенантного приложения. Вопросы о способе партиционировании приложения и о выставлении клиентам счетов за его использование непосредственно связаны с мультитенантной архитектурой. Вообще говоря, вопросы масштабирования приложения и реализации процесса заведения новых подписок связаны и с выделенной, и с мультитенантной моделями, но в последней необходимо учитывать некоторые особенности.

В этой главе описывается, каким образом компанией Tailspin были решены эти вопросы при разработке приложения Surveys. Для других приложений могут лучше подойти иные варианты.

Партиционирование

Глава 6 «Работа с данными в приложении Surveys» описывает, как в модели данных приложения Surveys данные партиционируются по подписчикам. В этом разделе описывается, каким образом в приложении Surveys используются таблицы и таблицы маршрутизации MVC для обеспечения подписчику доступа лишь к его собственным данным.

Решение

Разработчики в компании Tailspin решили определить подписчика, работающего с приложением, через URLадрес. Для доступа к веб-сайту подписчиков пользователи должны пройти проверку подлинности, прежде чем они получат доступ к приложению, а для общедоступного веб-сайта Surveys приложение не требует проверки подлинности.

URL-адрес определяет функциональную область приложения, подписчика и действие.

Ниже приведены три образца путей на веб-сайте подписчиков:

- /survey/adatum/newsurvey
- /survey/adatum/newquestion
- /survey/adatum

Ниже приведены два примера путей на общедоступном веб-сайте Surveys:

- /survey/adatum/launch-event-feedback
- /survey/adatum/launch-event-feedback/thankyou

Первый элемент пути служит для определения различных функциональных областей приложения. Все предыдущие примеры относятся к опросам, но другие области связаны с безопасностью. Второй элемент указывает имя подписчика (в приведенных примерах это «Adatum»), а последний элемент указывает выполняемое действие — например, создание нового опроса или добавление в опрос нового вопроса.

Проектировать структуру путей для приложения нужно внимательно, чтобы исключить возможность возникновения конфликтов имен, которые могут появиться при вводе значения подписчиком. В приложение Surveys при создании подписчиком опроса под названием «newsurvey», путь к этому опросу совпадет с путем, используемым подписчиками страницы для создания новых опросов. Но в данном конкретном случае конфликт имен не возникнет, так как опросы в приложении размещаются в конечной точке HTTP, а страница создания опросов — в конечной точке HTTPS.

Примечание.

Третий элемент примеров путей на общедоступном веб-сайте Surveys, «launch-event-feedback», — это «склеенная» версия заголовка опроса, изначально выглядевшего как «Launch Event Feedback», что позволяет использовать такую строку в URL-адресе.

Комментарий Маркуса:

Склеенное имя — это строка, в которой все пробелы и недопустимые символы заменены знаком дефиса (-). Этот термин не имеет ничего общего с конторским клеем!

Реализация

Сейчас подходящий момент, чтобы подробно рассмотреть код, обрабатывающий маршрутизацию запросов в приложение. В ходе изучения этого раздела, возможно, потребуется загрузить среду разработки Microsoft Visual Studio и приложение Tailspin Surveys (http://wag.codeplex.com/).

В данной реализации используется сочетание таблиц маршрутизации ASP.NET и областей MVC для идентификации подписчика и сопоставления запросов с соответствующим функционалом в приложение.

В следующем примере кода показано, как общедоступный веб-сайт Surveys использует таблицы маршрутизации для определения отображаемого опроса по URL-адресу.

C#

```
using System.Web.Mvc;
using System.Web.Routing;

public static class AppRoutes
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.MapRoute("Home",string.Empty,
            new { controller = "Surveys", action = "Index" });
         routes.MapRoute(
            "ViewSurvey",
            "survey/{tenant}/{surveySlug}",
            new { controller = "Surveys", action = "Display" });
         routes.MapRoute(
            "ThankYouForFillingTheSurvey",
            "survey/{tenant}/{surveySlug}/thankyou",
             new { controller = "Surveys", action =
"ThankYou" });
    }
}
```

Код извлекает из URL-адреса имя клиента и имя опроса и передает их соответствующему методу действия в классе **SurveysController**. В следующем примере кода показан метод действия **Display**, который обрабатывает HTTP-запросы GET.

C#

```
 [HttpGet]
public ActionResult Display(string tenant, string surveySlug)
{
    var surveyAnswer = CallGetSurveyAndCreateSurveyAnswer(
         this. surveyStore, tenant, surveySlug);
     var model = new
        TenantPageViewData<SurveyAnswer>(surveyAnswer);
    model.Title = surveyAnswer.Title;     return this.View(model);
}
```

Если пользователь запрашивает опрос по URL-адресу со значением пути /survey/adatum/launch-eventfeedback, то параметр *tenant* будет иметь значение «adatum», а параметр *surveySlug* будет иметь значение «launch-event-feedback». Этот метод действия по значениям параметров извлекает опрос из хранилища, заполняет модель этими данными и передает модель в представлению для подготовки к отображению в браузере.

Комментарий Маркуса:

Также имеется действие Display для обработки HTTP-запросов POST. Это действие контроллера отвечает за сохранение данных заполненного опроса.

Веб-сайт подписчиков более сложен, так как он должен проводить проверку подлинности и заведение новых подписчиков, помимо предоставления подписчикам возможности разработки новых опросов и анализа результатов опросов. В связи с такой сложностью в нем используются и области MVC, и таблица маршрутизации. Следующий код из класса **AppRoutes** в проекте TailSpin.Web показывает, как приложение сопоставляет запросы верхнего уровня с классами контроллера, обрабатывающими адаптацию и проверку подлинности.

C#

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.MapRoute( "OnBoarding", string.Empty,
        new { controller = "OnBoarding", action = "Index" });
     routes.MapRoute(
        "FederationResultProcessing",
        "FederationResult",
        new { controller = "ClaimsAuthentication",
              action = "FederationResult" });
     routes.MapRoute(
        "FederatedSignout",
        "Signout",
        new { controller = "ClaimsAuthentication",
              action = "Signout" });
    }
    …
}
```

Приложение также определяет область MVC для основной функциональности опросов. Приложения MVC регистрируют области, вызывая метод **RegisterAllAreas**. В проекте TailSpin.Web этот вызов находится в методе **Application_Start**, в файле Global.asax.cs. Метод **RegisterAllAreas** выполняет поиск приложения для классов, расширяющих класс **AreaRegistration**, а затем вызывает метод **RegisterArea**. В следующем примере кода показана часть этого метода из класса **SurveyAreaRegistration**.

Комментарий Маркуса:

Области MVC позволяют сгруппировать в приложении несколько контроллеров, что упрощает работу с большими проектами MVC. Каждая область обычно представляет отдельную функцию приложения.

C#

```
public override void RegisterArea(
                        AreaRegistrationContext context)
{
    context.MapRoute("MySurveys", "survey/{tenant}",
        new { controller = "Surveys", action = "Index" });

    context.MapRoute(
        "NewSurvey",
        "survey/{tenant}/newsurvey",
        new { controller = "Surveys", action = "New" });

    context.MapRoute(
        "NewQuestion",
        "survey/{tenant}/newquestion",
        new { controller = "Surveys", action = "NewQuestion" });

    context.MapRoute(
        "AddQuestion",
        "survey/{tenant}/newquestion/add",
        new { controller = "Surveys", action = "AddQuestion" });
     …
}
```

Обратите внимание, что все маршруты в этой таблице маршрутизации содержат имя клиента, передаваемое MVC в методы действий контроллера в качестве параметра.

Провиженинг пробной учетной записи и новых клиентов

При каждой регистрации нового подписчика в службе Surveys приложение должно выполнить определенные настройки и подготовительные действия. Компания Tailspin хочет автоматизировать как можно большую часть этого процесса, чтобы упростить процесс получения пробного доступа и свести к минимуму затраты, связанные с созданием новых подписчиков. Процесс провиженинга затрагивает многие компоненты приложения Surveys. В этом разделе описывается, как именно процесс провиженинга влияет на эти компоненты.

Основные сведения о подписках

В следующей таблице описаны основные сведения, указываемые каждым из подписчиков при регистрации в службе Surveys.

Информация	Пример	Примечание
Имя подписчика	Adatum Ltd	Полное название организации подписчика. Приложение использует его для изменения страниц подписчика на веб-сайтах Surveys. Подписчик также может задать корпоративный логотип.

Алиас подпичиска adatum Уникальный псевдоним, используемый в приложении для идентификации подписчика. Например, он служит составной частью URL-адресов веб-страниц подписчика. Приложение создает значение на основе имени подписчика, но позволяет подписчику переопределить предложенное значение.

Тип подписки	TypeTrial, Individual, Standard, Premium	Тип подписки определяет набор доступных подписчику функций и может влиять на список дополнительной информации, требуемой от подписчика для адаптации.
Способ оплаты		Данные кредитной карты Сведения о кредитной карте
		Кроме пробных подписок, все другие виды подписок являются платными. Это приложение использует стороннее решение для обработки платежей по кредитным картам.

Помимо сведений о кредитной карте, вся эта информация размещается в хранилище Windows Azure, информация используется в процессе провиженинга и в течение всего срока действия подписки.

Сведения о проверке подлинности и авторизации

В разделе «Проверка подлинности и авторизации» главы 4 «Доступ к приложению Surveys» этой книги описано три варианта управления доступом к приложению. Каждый из этих вариантов связан с определенным типом подписки и требует от подписчика разных данных в процессе адаптации. Например, тип индивидуальной подписки использует для проверки подлинности Windows Live ID или Google ID, а тип расширенной подписки — собственный поставщик удостоверения подписчика.

Провиженинг отношений доверия с поставщиком удостоверения подписчика

Одной из возможностью расширенного типа подписки является интеграция с поставщиком удостоверения подписчика. В процессе провиженинга собирается информация, необходимая для настройки отношений доверия между службой маркеров безопасности (STS) подписчика и поставщиком федерации Tailspin (FP) STS. Эти сведения описаны в следующей таблице.

Информация	Пример	Примечание
Subscriber Federation Metadata URL	https://login.adatum.net/FederationMetadata/2007-06/FederationMetadata.xml	Это должна быть общедоступная конечная точка. Так же можно разрешить подписчику передавать эти данные вручную (просто загрузить файл метаданных).
Данные администратора (например, email)	john@adatum.com	Приложение Surveys создает правило в своем FP (провайдер федерации) для сопоставления этого идентификатора с ролью администратора в приложении Surveys.
Тип утверждения идентификатора пользователя	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	Это тип утверждения, которое будет выдаваться службой STS подписчика для идентификации пользователя.
Открытый ключ подписчика	adatum.cer	Подписчик может предоставить сертификат, если требуется шифрование передаваемых маркеров.
Правила преобразования утверждений	Группа: Пользователи домена => Роль: Создатель опроса	Эти правила сопоставляют типы утверждений подписчика с типами утверждений, поддерживаемыми приложением Surveys.

Приложение Surveys будет использовать эти данные для добавления соответствующих сведений о конфигурации в поставщик федерации (ФП) службы Tailspin STS. В процессе адаптации также предоставляется доступ к метаданным федерации ПФ Tailspin подписчику, так как они могут ему понадобиться для настройки отношений доверия в его собственной службе STS.

Комментарий Яны:

Приложение еще не реализует эту функциональность. Компания Tailspin могла выбрать в качестве своего поставщика федерации ADFS, ACS или пользовательскую службу STS. В рамках процесса провиженинга приложение Surveys должно будет программным образом создавать отношения доверия между поставщиком федерации службы STS Tailspin и поставщиком удостоверений клиента, а также программно добавлять любые правила преобразования утверждений в службу STS Tailspin.

Примечание.

Дополнительные сведения см. в разделе «Установка и физическое развертывание» на странице 97 книги *Руководство по управлению удостоверениями и доступом на основе утверждений*. Версию этой книги в формате PDF вы можете загрузить по адресу http://msdn.microsoft.com/ru-ru/library/ff423674.aspx.

Провиженинг проверки подлинности и авторизации для базовых подписчиков

Подписчики со стандартным типом подписки не могут интегрировать приложение Surveys со своей собственной службой STS. Вместо этого они могут создавать своих пользователей в приложении Surveys. Во время процесса регистрации они указывают сведения об учетной записи администратора, которой будет предоставлен полный доступ ко всему, связанному с их счетом, включая сведения о выставлении счетов. В дальнейшем они могут определить дополнительных пользователей, входящих в роль «Создатель опросов», которые смогут только создавать опросы и анализировать результаты.

Провиженинг проверки подлинности и авторизации для индивидуальных подписчиков

Индивидуальные подписчики используют сторонние системы идентификации, например Windows Live ID, OpenID или Google ID, для проверки подлинности в приложении Surveys. Во время процесса регистрации они должны указать сведения об используемом в дальнейшем удостоверении. Только это удостоверение будет иметь права администратора на приложение.

Информация о географическом местоположении

В ходе процесса регистрации подписчик выбирает географическое местоположение, в котором приложение Surveys разместит его учетную запись. Список доступных расположений — это список мест, где в настоящее время находятся центры обработки данных Windows Azure. Географическое расположение определяет местонахождение экземпляра веб-сайта, которым будет пользоваться подписчик, а также место хранения всех данных приложения, связанных с учетной записью. В этом расположении также по умолчанию размещаются опросы подписчика, хотя подписчик может выбрать расположение отдельных опросов в других географических местоположениях.

Комментарий По:

Расположение может предлагаться автоматически, на основе IP-адреса пользователя, например с помощью такой службы, какhttp://ipinfodb.com/ip_location_api.php.

Сведения о базе данных

В ходе регистрации подписчик может также выбрать провиженинг базы данных SQL Azure для хранения и анализа данных их опросов. Приложение создает эту базу данных в том же географическом расположении, что и учетные записи подписчиков. Приложение использует псевдоним подписчика для формирования имени базы данных и пользователя базы данных. Приложение также создает случайный пароль. Приложение сохраняет строку подключения к базе данных в хранилище Windows Azure вместе с другими данными учетной записи подписчика.

Примечание.

База данных SQL Azure все же принадлежит компании Tailspin и оплачивается из ее средств. Компания Tailspin взимает с подписчиков оплату за эту услугу. Дополнительные сведения об использовании SQL Azure приложением Surveys см. в разделе «Использование SQL Azure» главы 6, «Работа с данными в приложении Surveys» этой книги.

Выставление счетов клиентам

Tailspin планирует ежемесячно выставлять каждому из клиентов счета на фиксированную сумму за использование приложения Surveys. Клиенты смогут подписаться на один из нескольких пакетов, описанных в следующей таблице.

Тип подписки	Учетные записи пользователей	Максимальная длительность опроса	Максимальное количество активных опросов
Trial (пробдная)	Одноучетная запись, связанная с социальным поставщиком удостоверений, например Windows Live ID или OpenID	5 дней	1
Basic (Базовая)	Одноучетная запись, связанная с социальным поставщиком удостоверений, например Windows Live ID или	14 дней	1
Standard (Стандратная)	Приложение Surveys предоставляет до пяти учетных записей пользователей.	28 дней	10
Premium (Премиум)	Неограниченное число учетных записей, связанных с собственным поставщиком удостоверений подписчика.	56 дней	50

Преимуществом такого подхода является простота как для компании Tailspin, так и для ее подписчиков, так как сумма ежемесячной оплаты для каждого из подписчиков фиксирована. Компании Tailspin необходимо провести исследование рынка, чтобы оценить число подписчиков и выбрать соответствующие тарифы по каждому из уровней подписки.

Комментарий Бхарата:

Компании Tailspin нужны хорошие оценки планируемых объемов использования услуг, чтобы определить затраты, доходы и прибыль.

В будущем Tailspin планирует предлагать возможность расширения базовых типов подписки. Например, компания заинтересована в предложении подписчикам возможности расширять длительность опроса сверх текущего предела или повышать количество одновременно активных опросов сверх текущего лимита. Для этого ей необходима возможность фиксирования показателей использования в приложении, что позволит рассчитывать дополнительные затраты, которые появятся в связи с обслуживанием подписчика. Компания Tailspin ожидает, что планируемые к выходу интерфейсы API Windows Azure, предоставляющие доступ к сведениям о тарификации и показателям использования хранилищ, упростят реализацию данных расширений.

Примечание.

На момент написания этого текста наилучшим способом получения показателей использования было ведение журналов. Может оказаться полезным ведение нескольких файлов журналов. Журнал служб IIS поможет определить, какой из клиентов генерировал основной трафик для веб-роли. Приложение сможет записывать пользовательские сообщения в WADLogsTable. Представление sys.bandwidth_usage в базе данных master каждого из серверов SQL Azure показывает использование полосы пропускания этой базой данных.

Настройка пользовательского интерфейса

Общей чертой мультитенантных приложений является предоставление подписчикам возможности настройки внешнего вида приложений для своих клиентов. Текущая версия приложения Surveys позволяет подписчикам настроить внешний вид страницы для своей учетной записи, загрузив изображение своего логотипа. Подписчик может загрузить изображение в свою учетную запись, а приложение Surveys сохранит изображение в составе данных учетной записи подписчика в хранилище больших двоичных объектов.

В следующих версиях приложения компания Tailspin планирует расширить доступные подписчикам параметры настройки. Эти расширения будут включать модификацию вида страниц опроса за счет добавления файлов графики, а также предоставление подписчику возможности загрузки файла каскадных таблиц стилей (CSS-файла) для изменения внешнего вида страниц опроса в соответствии со схемой корпоративного брэндинга.

Компания Tailspin проводит оценку рисков с точки зрения безопасности в случае разрешения подписчикам использования пользовательских CSS-файлов и планирует ограничить список функций каскадных таблиц стилей, которые будут поддерживаться сайтом. Будет реализован механизм сканирования, проверяющий содержание передаваемых подписчиками CSS-файлов на наличие функций, не поддерживаемых сайтом Surveys.

Комментарий По:

Одна из функций, которые не будут поддерживаться сайтом Surveys, — это behaivors в каскадных таблицах стилей.

Текущее решение позволяет подписчикам загружать изображения в общедоступный контейнер больших двоичных объектов под именем «logos». В рамках процесса передачи приложение добавляет URL-адрес изображения логотипа к данным больших двоичных объектов клиента, хранящимся в контейнере больших двоичных объектов под именем «tenants». Класс **TenantController** извлекает URL-адрес и перенаправляет его в представление.

Масштабирование приложений с помощью рабочих ролей

Масштабируемость является актуальной задачей как для выделенной, так и для мультитенантной архитектуры. Для выделенного приложения может быть приемлемым выполнять ресурсоемкие операции в определённый интервал времени (например, в наименее загруженный период), это гарантирует максимальную эффективность использования ресурсов. Но подобный подход не применим для большинства мультитенантных приложений, т.к. различные клиенты используют приложение в разное время и с различной периодичностью.

Рабочие роли в Windows Azure могут быть использованы для вынесения на них ресурсоемких и длительных операций с веб-ролей, которые предназначены для взаимодействия с пользователем. При этом рабочие роли могут выполнять задачи асинхронно, если веб-роли не требуется немедленного ответа.

Примеры сценариев использования рабочих ролей

В следующей таблице описаны некоторые примеры сценариев, в которых рабочие роли могут использоваться для асинхронной обработки заданий. Не все из этих сценариев взяты из приложения Surveys, но тем не менее, для каждого из сценариев в таблице указан способ активации задания и количество экземпляров рабочих ролей, которые могут в нем использоваться.

Сценарий	Описание	Решение
Обновление статистики опроса	Владелец статистики хочет просмотреть сводную статистику по опросу, в частности, общее количество ответов и средние значения оценок вопросов. Обработка такой статистики — очень ресурсоемкая задача.	Каждый раз, когда пользователь отвечает на вопрос, приложение помещает сообщение в очередь под именем *statistics-queue*, сообщение содержит информацию о расположении данных ответа. Каждые 10 минут рабочая роль извлекает ожидающие сообщения из очереди *statisticsqueue* и обновляет статистику по опросу с учетом новых ответов на опрос. Проводить обработку очереди должен только один рабочий экземпляр, чтобы избежать проблем, связанных конкуренцией за обновления данных статистики **Триггер**: по таймеру **Модуль выполнения**: один экземпляр рабочей
Выгрузка данных	Владелец опроса хочет **опроса в базу данных** проанализировать данные опроса **SQL Azure** в реляционной базе данных. Передача больших объемов данных — это длительная операция роли	Владелец опроса делает запрос к серверу на выгрузку ответов на опрос. Это действие создает строку в таблице *exports* и помещает в очередь *export-queue* сообщение, указывающее на эту строку в таблице. Любая рабочая роль может забрать сообщение из очереди *export-queue* и выполнить экспорт. После завершения экспорта экземпляр обновляет строку в таблице *exports*, указывая статус выполнения процедуры. **Триггер**: сообщением в очереди **Модель выполнения**: несколько экземпляров рабочей роли

Сохранение ответа на вопрос опроса	При каждом заполнении опроса респондентом данные ответов необходимо сохранить в хранилище. Пользователь не должен ждать, пока приложение выполнит сохранение данных опроса, и должен иметь возможность продолжить опрос.	Каждый раз, когда пользователь отправляет ответ на вопрос, приложение записывает эти данные в хранилище больших двоичных объектов и помещает сообщение в очередь *responses-queue*. Рабочая роль опрашивает очередь *responsesqueue* и при получении нового сообщения сохраняет данные ответа на вопрос в хранилище таблицы, а также помещает сообщение в очередь *statistics-queue* для вычисления статистики. **Триггер**: сообщением в очереди **Модель выполнения**: несколько экземпляров рабочей роли
Heartbeat	Все рабочие роли через фиксированные интервалы времени должны отправлять сигнал *ping*, чтобы сообщить контроллеру, что они все еще активны. Отправка сообщения ping не должна приводить к перерыву в процессе обработки основной выполняемой задачи.	Каждую минуту каждая из рабочих ролей выполняет кода, отправляющий подтверждение. **Триггер**: по таймеру **Модуль выполнения**: несколько экземпляров рабочей роли жны надежно отправляться без прерывания главной задачи рабочей роли. topicTitle: Построение масштабируемого многопользовательского приложения для Windows Azure

Примечание.

Сценарий обновления статистики опроса, описанный в предыдущей таблице, можно масштабировать, используя одну очередь и один экземпляр рабочей роли для каждого подписчика (тенанта) или даже для каждого из опросов. При этом важно, чтобы каждый из экземпляров рабочих ролей обрабатывал и обновлял свое собственное подмножество данных в наборе данных (dataset) и эти подмножества не пересекались.

После анализа сценариев вы можете сделать вывод, что рабочие роли, выполняющие фоновую обработку, относятся к одной из следующих категорий:

Триггер	Модель выполнения	Типы выполняемых задач
По таймеру	Одна рабочая роль	Операция над набором данных, которая часто обновляет данные и требует монопольной блокировки для избегания проблем, связанных с конкретным доступом. Примерами могут служить агрегирование, вычисление суммирующих результатов и денормализация

По таймеру	Несколько рабочих ролей	Независимые операции, работающие с независимыми с точки зрения доступа к данными. Примером может служить операция *ping*.
По сообщению в очереди	Одна рабочая роль или несколько	Операции над небольшим набором данных (например, большим двоичным объектом или несколькими строками таблицы), которая должна выполниться как можно раньше.

Триггеры для фоновых задач

Триггером для фоновой задачи может быть таймер или наличие сообщения в очереди. Запускаемые по расписанию фоновые задачи уместны в тех случаях, когда задача должна обрабатывать большой объем постепенно поступающих данных. Этот подход дешевле и обеспечивает большую пропускную способность по сравнению с подходом, при котором каждый из элементов данных обрабатывается сразу после его поступления. Это объясняется тем, что операции вы можете выполнять пакетно (batch), сокращая количество требуемых для обработки данных транзакций к хранилищу.

Если частота появления новых данных невелика, а обрабатывать их необходимо как можно быстрее, то рекомендуется в качестве триггера использовать сообщение в очереди.

Триггер по времени вы можете реализовать с помощью объекта **Timer** в рабочей роли, выполняющей задачу через постоянные промежутки времени. Чтоб реализовать в рабочей роли триггер на основе сообщений, вы можете создать бесконечный цикл, который опрашивает очередь сообщений. Из очереди вы можете извлекать либо одно, либо сразу несколько сообщений и выполнять их обработку.

Комментарий Маркуса:

В рамках одной транзакции из очереди вы можете извлечь несколько сообщений.

Модель выполнения

В Windows Azure фоновые задачи обрабатываются с помощью рабочих ролей. В приложение могут существовать отдельные типы рабочих ролей для каждого из видов фоновых задач, но при этом подходе потребуется как минимум по одному экземпляру рабочей роли для каждого из видов задач. Часто обработка одной рабочей ролью нескольких видов задач обеспечивает более эффективное использование доступных вычислительных ресурсов, особенно для больших объемов данных, так как такой подход снижает вероятность недозагрузки мощностей вычислительных ресурсов. Такой подход, часто называемый слиянием ролей (role conflation), связан с двумя компромиссами. Первый компромисс связан с тем, что сложность реализация такого решения значительно выше, но при этом достигается бо́льшая экономия на затратах на ресурсы за счет уменьшения числа выполняющихся экземпляров рабочих ролей. Второй — это компромисс между временем, необходимым на реализацию и проверку решения со слиянием ролей и другими бизнес-приоритетами, например, временем вывода продукта на рынок. В таком сценарии приложение может масштабировать горизонтально за счет запуска дополнительных экземпляров рабочей роли. Диаграммы на рис 1 показывают эти два варианта.

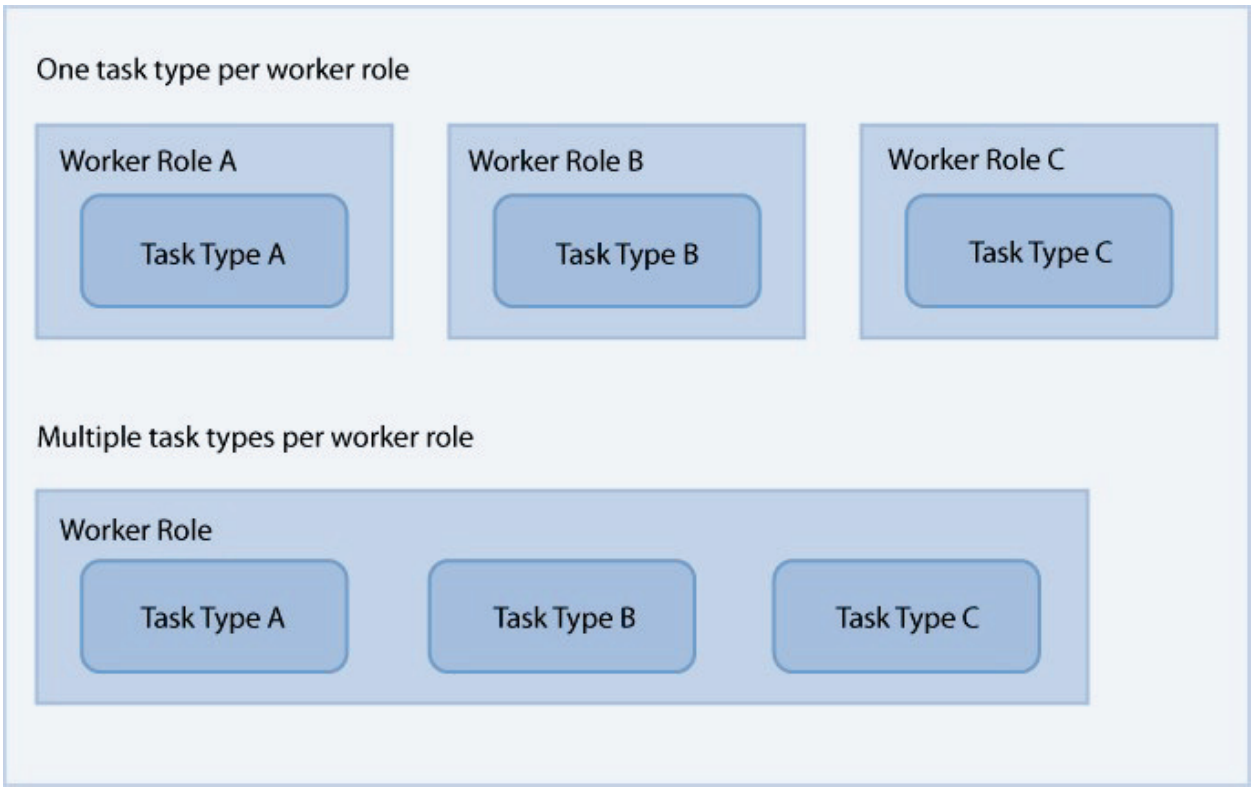


Рис. 1 **Обработка нескольких типов фоновых задач**

В сценарии, где несколько экземпляров рабочих ролей выполняют одинаковые типы задач, необходимо явно иметь возможность определять:

- какие задачи могут безопасно выполняться в нескольких рабочих ролях одновременно;
- какие задачи могут безопасно выполняться в любой момент времени только в одной рабочей роли.

Чтобы гарантировать, что единицу времени выполняется только одна копия задачи, необходимо реализовать механизм блокировок. В Windows Azure для этой цели можно использовать очереди сообщений или «захват» (leasing) BLOB объекта. На рис. 2 показано, что несколько копий задач А и В могут выполняться одновременно, но для задачи Б допустимо выполнение только одной копии.
Одна из копий задачи Б «захватывает» BLOB объект и запускается; другие копии задачи Б не запустятся, пока они не смогут получить доступ к этому BLOB объекту.

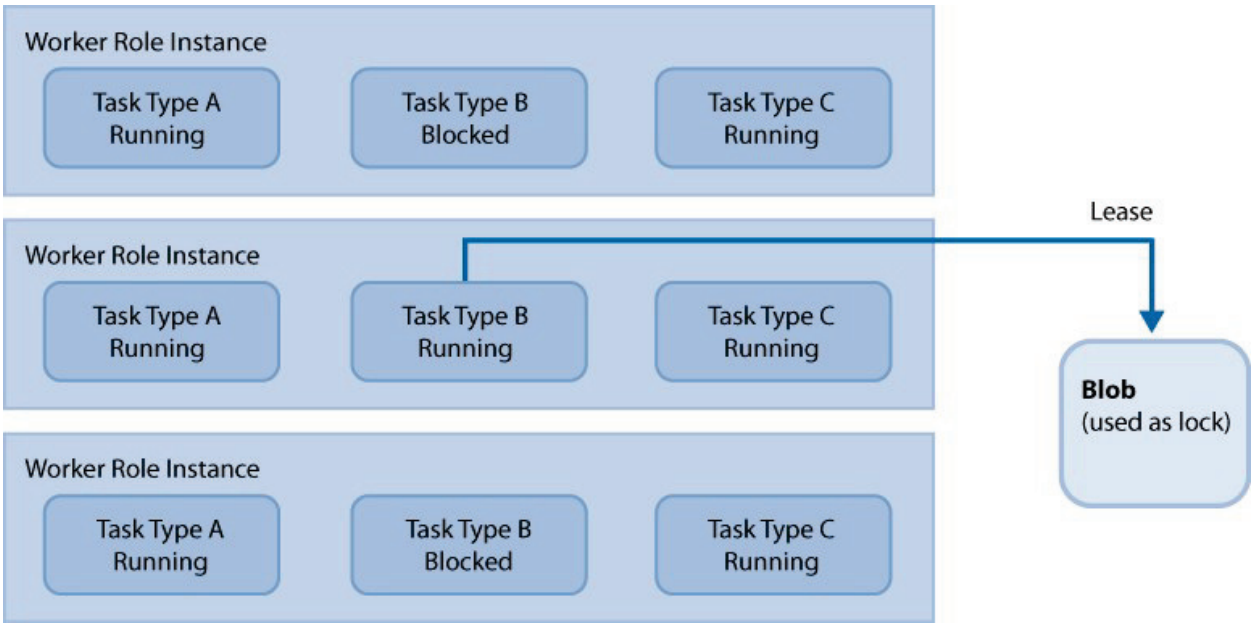


Рис. 2 **Несколько экземпляров рабочей роли**

Алгоритм MapReduce

Для некоторых приложений Windows Azure ограничение в один экземпляр задачи может существенно повлиять на производительность, например, это сложные и длительные вычисления. В такой ситуации алгоритм MapReduce может обеспечить возможность параллельного выполнения вычислений в нескольких экземплярах задач в нескольких рабочих ролях.

Исходные концепции алгоритма MapReduce взяты из функций **map** и **reduce**, широко применяемых в языках функционального программирования, таких как Haskell, F# и Erlang. В текущем контексте MapReduce является моделью программирования (запатентованной Google), которая позволяет распараллелить операции над большим набором данных. В случае приложения Surveys этот подход можно применить для вычисления сводной статистики несколькими выполняющимися параллельно задачами вместо одной задачи. Это ускорило бы вычисление сводной статистики, но за счет дополнительных затрат, связанных с наличия дополнительных экземпляров рабочих ролей.

Комментарий Яны:

Для приложения Surveys скорость при вычислении сводной статистики не является крайне важной. Компания Tailspin готова смириться с задержками при вычислении этих сводных данных, поэтому она не использует алгоритм MapReduce.

В следующем примере показано, каким образом можно было бы использовать этот подход для ускорения вычисления сводной статистики.

В этом примере предполагается, что приложение сохраняет ответы на опрос в больших двоичных объектах, содержащих показанные на рисунке 3 данные.

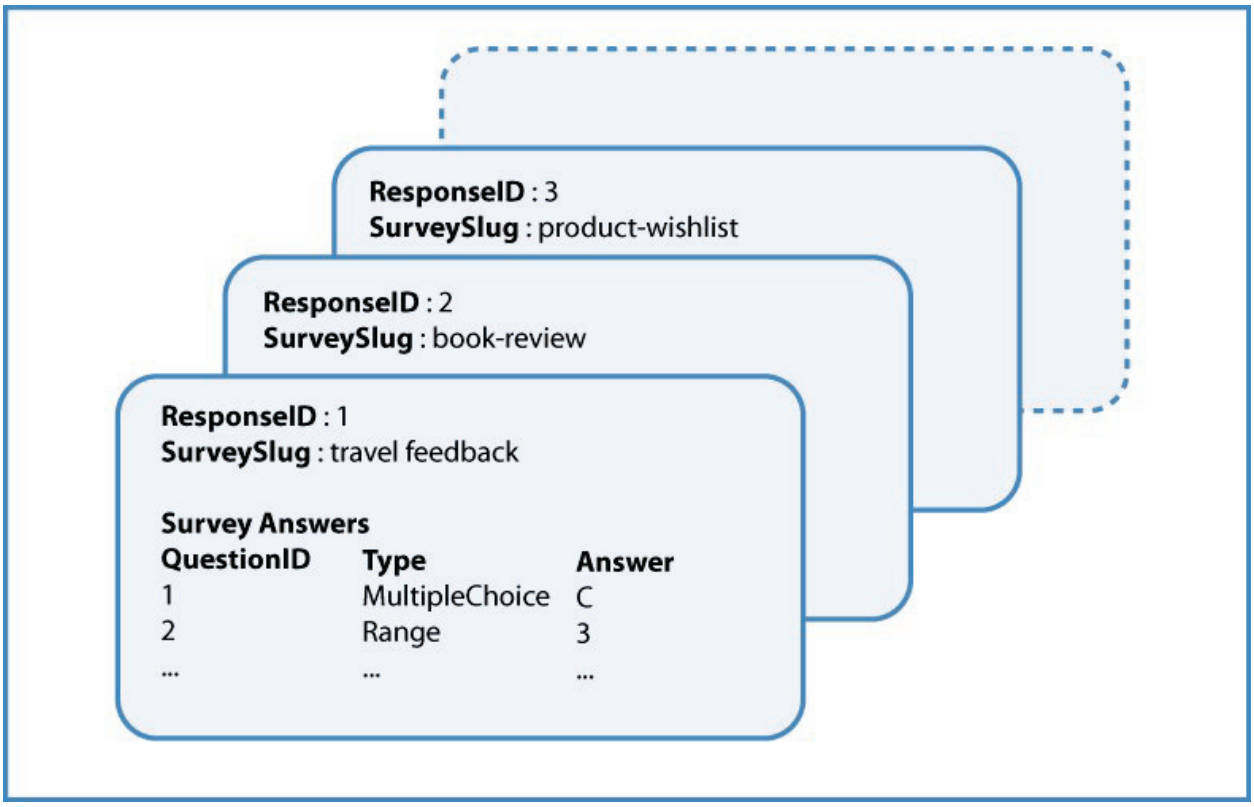


Рис. 3 **Пример большого двоичного объекта, содержащего данные ответов на опрос**

В следующей таблице показан начальный набор данных, на основе которого приложение должно рассчитать сводную статистику. На практике алгоритм MapReduce используется для обработки очень больших наборов данных; в этом примере небольшой набор данных используется для демонстрации способа работы MapReduce. В этом примере также показаны шаги вычисления сводки только для первого вопроса с несколькими вариантами ответа и первого вопроса с диапазоном ответов из всех ответов на опрос, но процесс можно легко расширить.

ResponseID	SurveySlug	Ответ на первый вопрос с множественным выбором	Ответ на первый вопрос с диапозоном ответов	Ответы на другие вопросы
1	travel-feedback	C	3	
2	book-review	D	3	
3	product-wishlist	A	4	
4	service-satisfaction	E	3	
5	travel-feedback	D	5	
6	travel-feedback	C	4	
7	purchase-experience	C	2	
8	brand-rating	B	3	
9	book-review	A	3	
10	travel-feedback	E	4	
11	book-review	D	3	

На первом этапе алгоритм MapReduce преобразует данные в формат, который можно последовательно сокращать до тех пор, пока не будут получены необходимые результаты. Обе фазы (map и recuce), можно распараллелить, именно поэтому алгоритм MapReduce может улучшить производительность при вычислениях на больших наборах данных.

В этом примере обе фазы (map и recuce) разобьют свои входные данные на блоки из трех элементов. Map фаза в этом примере использует четыре параллельных задачи, каждая из которых обрабатывает три BLOB объекта результатов вопроса. В итоге получается следующий результат, приведенный в таблице:

AggregationID	SurveySlug	Количество ответов	Гистограмма первого вопроса с несколькими вариантами ответа	Среднее для первого вопроса с дипазоном ответов
1.1	travel-feedback	1	C	3
1.2	book-review	1	D	3
1.3	product- wishlist	1	A	4
2.1	service-satisfaction	1	E	3
2.2	travel-feedback	2	CD	4.5
3.1	purchase-experience	1	C	2
3.2	brand-rating	1	B	3
3.3	book-review	1	A	3
4.1	travel-feedback	1	E	4
4.2	book-review	1	D	3

На следующем этапе эти данные еще больше сокращаются. В настоящем примере параллельно будут выполняться две задачи, одна будет обрабатывать агрегаты 1.X, 2.X и 3.X, а вторая — агрегат 4.X.

Важно понимать, что каждый этап сокращения ссылается только на данные предыдущего этапа, а не на исходные данные. В следующей таблице показаны результаты этого этапа сокращения:

AggregationID	SurveySlug	Количество ответов	Гистограмма первого вопроса с несколькими вариантами ответа	Среднее для первого вопроса с диапазоном ответов
1.1	travel- feedback	3	CCD	4
1.2	book-review	2	AD	3
1.3	product-wishlist	1	AD	4
1.4	service-satisfaction	1	E	3
1.5	purchase-experience	1	C	2
1.6	brand-rating	1	B	3
2.1	travel-feedback	1	E	4
2.2	book-review	1	D	3

На следующем этапе задача будет только одна, так как для нее имеется лишь два входных блока. В следующей таблице показаны результаты этого этапа:

AggregationID	SurveySlug	Количество ответов	Гистограмма первого вопроса с несколькими вариантами ответа	Среднее для первого вопроса с диапазоном ответов
1.1	travel-feedback	4	CCDE	4
1.2	book-review	3	ADD	3
1.3	product-wishlist	1	AD	4
1.4	service-satisfaction			
1.5	purchase-experience	1	C	2
1.6	brand-rating	1	B	3

В этот момент дальнейшее сокращение данных становится невозможным, то есть сводная статистика вычислена для всех данных опроса.
Теперь можно обновить сводные данные с учетом ответов на опрос, полученных после запуска начальной map-фазы. Сначала все новые данные опроса обрабатываются с помощью алгоритма MapReduce, а затем на последнем шаге полученные новые данные совмещаются со старыми.

Масштабирование приложения Surveys

В этом разделе описывается, каким образом Tailspin спроектировала приложения Surveys для обеспечения масштабируемости. Компания Tailspin предполагает, что у некоторых из опросов могут быть тысячи и даже сотни тысяч респондентов. Общедоступный веб-сайт должен всегда быстро реагировать на запросы пользователей. В то же время владельцам опросов необходима возможность просмотра сводной статистики на текущую дату, вычисляемой по ответам на вопрос.

Цели и требования

В главе 4, «Доступ к приложению Surveys», были описаны два веб-сайта, используемые компанией Tailspin для приложения Surveys: на одном подписчики разрабатывают свои опросы и управляют ими, а на втором пользователи вводят ответы. Приложение Surveys в настоящее время поддерживает три типа вопросов: с произвольным текстом ответов, с числовым диапазоном ответов (значения от одного до пяти) и с несколькими вариантами ответов. Владельцам опросов необходима возможность просмотра некоторой базовой сводной статистики, вычисляемой приложением для каждого из опросов, например общего числа полученных ответов, гистограмм по вопросам с несколькими вариантами ответов и агрегированными значениям, например, средним по ответам на вопросы с диапазоном ответов. Приложение Surveys реализует предопределенный набор сводных статистических показателей, который не может изменяться подписчиками. Если подписчику необходим более сложный анализ ответов на свои опросы, он может экспортировать данные опроса в экземпляр SQL Azure.

Если количество обрабатываемых ответов велико, то расчет сводной статистики становится дорогостоящей операцией.

В связи с большим объемом ожидаемых данных ответов на опросы разработчики компании Tailspin предполагают, что формирование сводной статистики будет дорогостоящей операцией в связи с большим числом транзакций в хранилище, необходимых при считывании приложением ответов на опросы. Однако сводная статистика не обязательно должна быть всегда актуальной и основываться на всех доступных ответах на опросы. Разработчики компании Tailspin готовы допустить задержку при вычислении приложением сводных данных, если это снизит стоимость их расчета.
У общедоступного сайта, на котором респонденты дают ответы на опросы, всегда должно быть минимальное время отклика при сохранении пользователями ответов, и он должен точно записывать ответы, чтобы исключить возможность наличия ошибок в данных при анализе результатов подписчиком.

Разработчикам в Tailspin также необходима возможность без каких-либо зависимостей от хранилища Windows Azure выполнять unint-тесты компонентов, вычисляющих сводную статистику.

Комментарий Маркуса:

Также имеются тесты на интеграцию, проверяющие комплексную работу приложения с использованием хранилища Windows Azure.

Решение

Разработчики в Tailspin решили создать рабочую роль, которая будет выполнять задачу формирования сводной статистики на основе результатов опроса. Использование рабочей роли позволяет приложению выполнять этот ресурсоемкий процесс как фоновую задачу и гарантировать, что веб-роль, ответственная за сбор ответов на опрос, не блокируется на время вычисления сводной статистики в приложении.

Рабочая роль созданная на основе принципов, описанных в предыдущем раздел. Асинхронная задача будет вызываться по расписанию и будет выполняться как процесс с одним экземпляром, так как она обновляет единственный набор результатов.

Приложение может использовать дополнительные задачи в той же рабочей роли для проведения любой дополнительной обработки данных отчета; например, оно может создать список упорядоченных ответов, чтобы обеспечить пролистывание данных ответов.

При вычислении данных статистики Tailspin рассмотрела два основных подхода. При первом подходе задача в рабочей роли получает все ответы на опрос на текущую дату через постоянные интервалы времени, пересчитывает сводную статистику и сохраняет сводные данные поверх существующих. Второй подход заключается в том, что задача в рабочей роли получает все данные ответов на опросы, записанные приложением с момента последнего запуска задачи, и использует эти данные для коррекции сводной статистики и отражения в ней новых результатов опросов.

Первый подход проще всего в реализации, так как для второго подхода необходим механизм, отслеживающий, какие из результатов опроса являются новыми (еще не учтены). Второй подход также зависит от возможности рассчитывать новые сводные данные на основе старых сводных данных и новых результатов опроса без повторного считывания всех исходных результатов опроса.

Комментарий Маркуса:

Для ведения списка всех новых ответов на опрос вы можете использовать очередь. Запуск этой задачи все еще выполняется по расписанию, которое определяет, насколько часто задача должна проверять очередь на наличие новых результатов опроса для обработки.

Примечание.

Все сводные данные в приложении Surveys вы можете перерассчитывать с помощью второго подхода. Однако можно представить ситуацию, когда один из компонентов сводных данных должен быть списком из 10 самых популярных слов, использованных в ответах на вопрос с произвольным текстом ответа. В этом случае всегда придется обрабатывать все результаты опроса, если только не ведется отдельный список всех использованных слов и количества их вхождений. Это увеличивает сложность второго подхода.

Ключевое различие между этими двумя подходами заключается в стоимости. График на рисунке 4 показывает результаты анализа, в котором сравниваются затраты при этих двух подходах для трех различных объемов ежедневных ответов на опросы. Первый подход показан на графике на верхней линии с меткой «Перерасчет», а второму соответствует нижняя линия с меткой «Слияние».

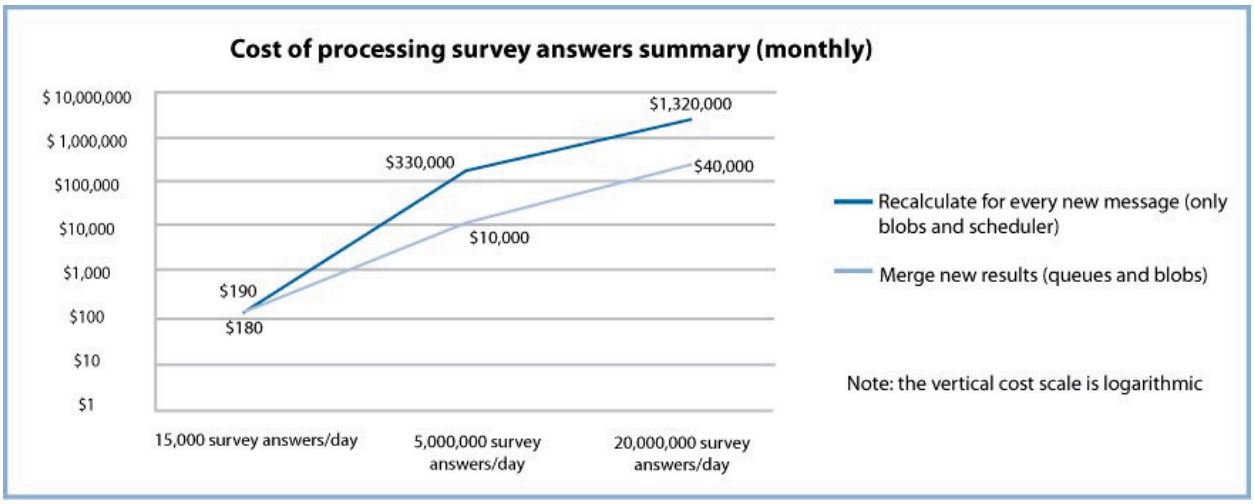


Рис. 4 Сравнение стоимости различных подходов к вычислению сводной статистики

Из графика очевидно, насколько подход слияния дешевле, чем подход повторного вычисления, после превышения определенного объема транзакций. Разница в стоимости практически полностью обусловлена стоимостью транзакций, связанных с этими подходами. Компания Tailspin решила реализовать в приложении Surveys подход слияния.

Примечание.

Масштаб вертикальной шкалы затрат на графике — логарифмический. В анализе, на основе которого построен этот график, использован ряд допущений («худших случаев») в отношении способа обработки результатов опросов в приложении. Задача данного графика — показать сравнительные характеристики двух подходов в отношении затрат; он не дает точных цифр.

Если для вычисления сводных данных будут использоваться лишь образцы (выборки) ответов на опросы вместо обработки каждого из ответов, то подход повторного вычисления можно оптимизировать. Для вычисления сводной статистики с допустимой погрешностью необходимо будет провести подробный статистический анализ и определить, какую часть результатов необходимо обработать.

В приложении Surveys также можно было бы формировать сводную статистику с помощью основанного на алгоритме MapReduce подхода. Преимущество такого подхода заключается в возможности использования нескольких экземпляров задач для расчета сводной статистики. Однако компания Tailspin готова смириться с задержкой при вычислении сводной статистики, так что производительность для этой задачи не очень важна. Описание модели программирования MapReduce см. в разделе «MapReduce» ранее в этой главе.

Реализация

Сейчас будет уместно более подробно проанализировать код, реализующий асинхронную задачу, которая вычисляет сводную статистику. В ходе изучения этого раздела, возможно, потребуется загрузить решение Visual Studio приложения Tailspin Surveys со страницы http://wag.codeplex.com/.

Использование рабочей роли для вычисления сводной статистики

Группа разработчиков в Tailspin решила реализовать асинхронную фоновую задачу, которая вычисляет сводную статистику по результатам опроса, с помощью подхода слияния. Каждый раз при запуске задача обрабатывает ответы на опрос, полученные приложением с момента прошлого запуска задачи; новая сводная статистика вычисляется путем объединения новых результатов со старыми статистическими данными.

Рабочая роль в проекте TailSpin.Workers.Surveys регулярно проверяет очередь на наличие ожидающих обработки ответов на опрос.

В следующем примере кода из класса **UpdatingSurveyResultsSummaryCommand** показано, как рабочая роль обрабатывает каждый из временных ответов на опрос и использует их для перерасчета сводной статистики.

C#

```
private readonly IDictionary<string, SurveyAnswersSummary>
      surveyAnswersSummaryCache;
private readonly ISurveyAnswerStore surveyAnswerStore; private
readonly ISurveyAnswersSummaryStore
      surveyAnswersSummaryStore;

public UpdatingSurveyResultsSummaryCommand(
     IDictionary<string, SurveyAnswersSummary>
      surveyAnswersSummaryCache,
    ISurveyAnswerStore surveyAnswerStore,
    ISurveyAnswersSummaryStore surveyAnswersSummaryStore)
{
        this.surveyAnswersSummaryCache =
              surveyAnswersSummaryCache;
        this.surveyAnswerStore = surveyAnswerStore;
         this.surveyAnswersSummaryStore =
              surveyAnswersSummaryStore;
    }
public void PreRun()
{
    this.surveyAnswersSummaryCache.Clear();
}

public void Run(SurveyAnswerStoredMessage message)
{
   this.surveyAnswerStore.AppendSurveyAnswerIdToAnswersList(
         message.Tenant,
         message.SurveySlugName,
        message.SurveyAnswerBlobId);

    var surveyAnswer =
        this.surveyAnswerStore.GetSurveyAnswer(
```

```
            message.Tenant,
            message.SurveySlugName,
            message.SurveyAnswerBlobId);

    var keyInCache = string.Format(CultureInfo.InvariantCulture,
        "{0}_{1}", message.Tenant, message.SurveySlugName);
    SurveyAnswersSummary surveyAnswersSummary;

    if (!this.surveyAnswersSummaryCache.ContainsKey(keyInCache))
    {
        surveyAnswersSummary = new
            SurveyAnswersSummary(message.Tenant,
            message.SurveySlugName);
        this.surveyAnswersSummaryCache[keyInCache] =
              surveyAnswersSummary;
    }
    else
    {
        surveyAnswersSummary =
            this.surveyAnswersSummaryCache[keyInCache];
    }

    surveyAnswersSummary.AddNewAnswer(surveyAnswer);
}

public void PostRun()
{
    foreach (var surveyAnswersSummary in
          this.surveyAnswersSummaryCache.Values)
    {
        var surveyAnswersSummaryInStore =
              this.surveyAnswersSummaryStore
            .GetSurveyAnswersSummary(surveyAnswersSummary.Tenant,
              surveyAnswersSummary.SlugName);

        surveyAnswersSummary.MergeWith(
             surveyAnswersSummaryInStore);

        this.surveyAnswersSummaryStore
            .SaveSurveyAnswersSummary(surveyAnswersSummary);
    }
}
```

Приложение Surveys использует Unity Application Block (Unity) для инициализации экземпляра класса **UpdatingSurveyResultsSummaryCommand** и переменных **surveyAnswerStore** и **surveyAnswersSummaryStore**. Переменная **surveyAnswerStore** является экземпляром типа **SurveyAnswerStore,** который используется методом **Run** для считывания ответов на опрос из хранилища больших двоичных объектов. Переменная **surveyAnswersSummaryStore** является экземпляром типа **SurveyAnswersSummary**, который используется методом **PostRun** для записи сводных данных в хранилище больших двоичных объектов.
Словарь **surveyAnswersSummaryCache** содержит объект **SurveyAnswersSummary** для каждого опроса.

Примечание.

Unity — это упрощенный, расширяемый контейнер внедрения зависимостей, который поддерживает перехват, внедрение конструктора, свойств и вызова методов. Контейнер Unity вы можете использовать различными способами для упрощения процесса разделения компонентов приложения, обеспечения максимальной согласованности компонентов и рационализации проектирования, реализации, тестирования и администрирования таких приложений. Дополнительные сведения о контейнере Unity см. на странице шаблонов и рекомендаций по контейнеру Unity на сайте CodePlex; на этой же странице можно загрузить блок приложений (http://unity.codeplex.com).

Метод **PreRun** выполняется перед тем, как задача считывает какие-либо сообщения из очереди и инициализирует временный кэш для новых данных по ответам на опрос.

Метод **Run** выполняется один раз для каждого нового ответа на опрос. С помощью сообщения из очереди он находит новый ответ на опрос и затем добавляет этот ответ в объект **SurveyAnswersSummary** для соответствующего опроса, вызвав метод **AddNewAnswer**. Метод **AddNewAnswer** обновляет сводные статистические данные в экземпляре **surveyAnswersSummaryStore**. Метод **Run** также вызывает метод **AppendSurveyAnswerIdToAnswersList** для обновления списка ответов на опрос, используемого приложением для разбиения на страницы.
Метод **PostRun** выполняется после того, как задача считает все необработанные ответы в очереди. Для каждого опроса он объединяет новые результаты с имеющимися сводными статистическими данными и затем сохраняет в хранилище больших двоичных объектов новые значения.

Рабочая роль использует «соединительный код», разработанный компанией Tailspin, для вызова методов **PreRun**, **Run** и **PostRun** в классе **UpdatingSurveyResultsSummaryCommand** по расписанию. В следующем примере кода показано, как приложение Surveys использует «соединительный код» в методе **Run** в рабочей роли для исполнения трех методов, охватывающих это задание.

C#

```
public override void Run()
{
    var updatingSurveyResultsSummaryJob =
        this.container.Resolve
        <UpdatingSurveyResultsSummaryCommand>();
     var surveyAnswerStoredQueue =
        this.container.Resolve
        <IAzureQueue<SurveyAnswerStoredMessage>>();
    BatchProcessingQueueHandler
        .For(surveyAnswerStoredQueue)
        .Every(TimeSpan.FromSeconds(10))
        .Do(updatingSurveyResultsSummaryJob);

    var transferQueue = this.container
        .Resolve<IAzureQueue<SurveyTransferMessage>>();
     var transferCommand = this
        .container.Resolve<TransferSurveysToSqlAzureCommand>();
    QueueHandler
        .For(transferQueue)
        .Every(TimeSpan.FromSeconds(5))
        .Do(transferCommand);

    while (true)
    {
        Thread.Sleep(TimeSpan.FromSeconds(5));
    }
```

```
}
```

Сначала этот метод с помощью контейнера Unity создает экземпляр объекта **UpdatingSurveyResultsSummaryCommand,** который определяет задание, и объекта **AzureQueue,** который содержит уведомления о новых ответах на опрос.

Затем этот метод передает такие объекты в качестве параметров «соединительным» методам **For** и **Do**. «Соединительный» метод **Every** определяет, как часто задание должно выполняться. Эти методы заставляют «соединительный код» вызвать методы **PreRun**, **Run** и **PostRun** в классе **UpdatingSurveyResultsSummaryCommand,** передавая сообщение из очереди методу **Run**.

В предыдущем примере кода также показано, как рабочая роль инициализирует задачу, определенную в классе **TransferSurveysToSqlAzureCommand**, выводящем данные опроса в SQL Azure. Эта задача несколько проще и содержит только метод **Run**.

Вы должны настроить частоту, с которой запускаются эти задачи, в зависимости от ожидаемых рабочих нагрузок, изменяя значение, передаваемое методу **Every**.

В заключение метод с помощью цикла **while** поддерживает экземпляр рабочей роли в активном состоянии.

«Соединительный код» рабочей роли

«Соединительный код» в рабочей роли дает возможность вызывать команды типа **IBatchCommand** или **ICommand** с помощью метода **Do** в очереди Windows Azure типа **IAzureQueue** с помощью метода **For** с заданным интервалом. На рис. 5 показаны основные типы, составляющие «соединительный код».

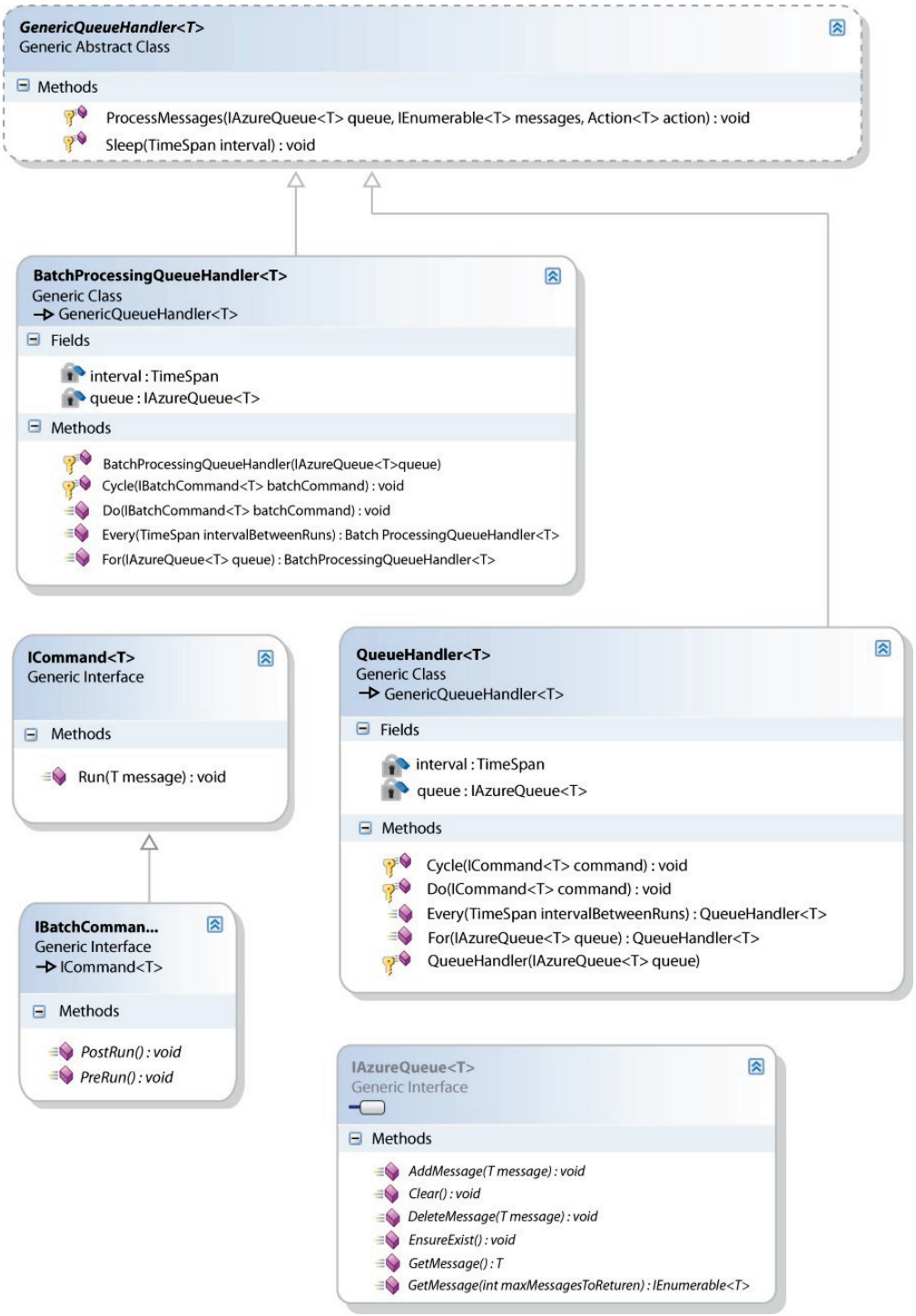


Рис. 5

Ключевые «соединительные» типы

На рис. 5 показаны классы **BatchProcessingQueueHandler** и **QueueHandler**. Класс **QueueHandler** исполняет задачи, реализующие более простой интерфейс **ICommand** вместо интерфейса **IBatchCommand**. Далее будет рассмотрена задача **BatchProcessingQueueHandlerTask,** используемая приложением для создания сводной статистики.

Сначала рабочая роль вызывает метод **For** в статическом классе **BatchProcessingQueueHandler,** который вызывает метод **For** в классе **BatchProcessingQueueHandler<T>** и возвращает экземпляр **BatchProcessingQueueHandler<T>**, содержащий ссылку на экземпляр **IAzureQueue<T>** для наблюдения. «Соединительный код» распознает очередь на основании типа сообщения очереди, производного от типа **AzureQueueMessage**. В следующем примере кода показано, как метод **For** в классе **BatchProcessingQueueHandler<T>** создает экземпляр **BatchProcessingQueueHandler<T>**.

C#

```
private readonly IAzureQueue<T> queue;
private TimeSpan interval;

protected BatchProcessingQueueHandler(IAzureQueue<T> queue)
{
    this.queue = queue;
    this.interval = TimeSpan.FromMilliseconds(200);
}

public static BatchProcessingQueueHandler<T> For(
    IAzureQueue<T> queue)
{
    if (queue == null)
    {
        throw new ArgumentNullException("queue");
    }

    return new BatchProcessingQueueHandler<T>(queue);
}
```

Комментарий Бхарата:

В текущей реализации используется одна очередь, но вы можете изменить **BatchProcessingQueueHandler** для считывания из нескольких очередей вместо одной. Согласно контрольным показателям, опубликованным на сайте http://azurescope.cloudapp.net, максимальная пропускная способность операции записи для очереди составляет от 500 до 700 элементов в секунду. Если приложению Surveys требуется обрабатывать более 2 миллионов ответов на опросы в час, оно достигнет порога записи для одной очереди. Можно изменить приложение так, чтобы оно использовало несколько очередей — предположительно отдельную очередь для каждого из подписчиков.

Затем рабочая роль вызывает метод **Every** объекта **BatchProcessingQueueHandler<T>**, чтобы определить, как часто следует выполнять задачу.

Далее рабочая роль вызывает метод **Do** объекта **BatchProcessingQueueHandler<T>**, передавая объект **IBatchCommand**, определяющий команду, которую «соединительный код» должен выполнить по каждому сообщению в очереди. В следующем примере кода показано, как метод **Do** использует метод **Task.Factory.StartNew** из библиотеки параллельных задач (TPL) для запуска методов **PreRun**, **ProcessMessages** и **PostRun** в очереди с заданным интервалом.

Используйте Task.Factory.StartNew, а не ThreadPool.QueueUserWorkItem.

C#

```
(public virtual void Do(IBatchCommand<T> batchCommand)
{
  Task.Factory.StartNew(() =>
           {
             while (true)
             {
               this.Cycle(batchCommand);
             }
           }, TaskCreationOptions.LongRunning);
}

protected void Cycle(IBatchCommand<T> batchCommand)
{
  try
  {
    batchCommand.PreRun{};

    bool continueProcessing;
    do
    {
      var messages = this.queue.GetMessages(32);
      ProcessMessages(this.queue, messages,
        batchCommand.Run);

      continueProcessing = messages.Count() > 0;
    }
    while (continueProcessing);

    batchCommand.PostRun();

    this.Sleep(this.interval);
  }
  catch (TimeoutException)
  {
  }
}
```

Метод **Cycle** циклически запрашивает до 32 сообщений из очереди за одну транзакцию обработки, пока сообщения не закончатся.

В следующем примере кода показан метод **ProcessMessages** в классе **GenericQueueHandler**.

C#

```
protected static void ProcessMessages(IAzureQueue<T> queue,
IEnumerable<T> messages, Action<T> action)
{
    …

    foreach (var message in messages)
    {
        var success = false;

        try
        {
            action(message);
            success = true;
        }
        catch (Exception)
        {
            success = false;
        }         finally
        {
            if (success || message.DequeueCount > 5)
            {
            queue.DeleteMessage(message);
            }
        }
    }
}
```

Этот метод использует параметр *action* для вызова пользовательской команды по каждому сообщению в очереди. В заключение метод проверяет наличие сообщений о сбое, просматривая свойство **DequeueCount** сообщения. Если приложение пыталось обработать это сообщение более пяти раз, метод удаляет такое сообщение.

Примечание.

Вместо того чтобы удалять сообщения о сбое, их следует направлять в очередь отклоненных (dead) сообщений для анализа и решения проблемы.

Тестирование рабочей роли

Реализация «соединительного кода» в рабочей роли и использование контейнера Unity позволяют выполнять модульные тесты (unit-тесты) в компонентах рабочей роли с помощью имитирующих объектов вместо очередей Windows Azure и больших двоичных объектов. В следующем коде из класса **BatchProcessingQueueHandlerFixture** показаны два примера модульных тестов.

C#

```
[TestMethod]
public void ForCreatesHandlerForGivenQueue()
{
    var mockQueue = new Mock<IAzureQueue<StubMessage>>();

    var queueHandler = BatchProcessingQueueHandler
         .For(mockQueue.Object);

    Assert.IsInstanceOfType(queueHandler,
        typeof(BatchProcessingQueueHandler<StubMessage>));
}

[TestMethod]
public void DoRunsGivenCommandForEachMessage()
{
    var message1 = new StubMessage();
     var message2 = new StubMessage();
     var mockQueue = new Mock<IAzureQueue<StubMessage>>();
     var queue = new Queue<IEnumerable<StubMessage>>();
     queue.Enqueue(new[] { message1, message2 });
     mockQueue.Setup(q => q.GetMessages(32)).Returns(()
=> queue.Count > 0 ?
      queue.Dequeue(): new StubMessage[] { });
     var command = new Mock<IBatchCommand<StubMessage>>();
     var queueHandler =
         new
BatchProcessingQueueHandlerStub(mockQueue.Object);

    queueHandler.Do(command.Object);

    command.Verify(c => c.Run(It.IsAny<StubMessage>()),
          Times.Exactly(2));
    command.Verify(c => c.Run(message1));
     command.Verify(c => c.Run(message2));
}

public class StubMessage: AzureQueueMessage
{
}
```

C#

```
private class BatchProcessingQueueHandlerStub:
    BatchProcessingQueueHandler<StubMessage>
{
    public BatchProcessingQueueHandlerStub(
         IAzureQueue<StubMessage> queue): base(queue)
```

```
    {
    }

    public override void Do(
        IBatchCommand<StubMessage> batchCommand)
    {
        this.Cycle(batchCommand);
    }
}
```

Модульный тест **ForCreateHandlerForGivenQueue** проверяет, что статический метод **For** корректно создает экземпляр объекта **BatchProcessingQueueHandler** с помощью фиктивной очереди. Модульный тест **DoRunsGivenCommandForEachMessage** проверяет, что метод **Do** вызывает выполнение команды в каждом сообщении в очереди с помощью фиктивной очереди и объектов команд.

Ссылки и материалы

Дополнительные сведения о маршрутизации ASP.NET см. На странице «Маршрутизация ASP.NET» в MSDN: http://msdn.microsoft.com/ru-RU/library/cc668201.aspx

Дополнительные сведения о модуле перезаписи URL-адресов для IIS см. на сайте IIS.net на странице «Перезапись URL-адреса»: http://www.iis.net/download/urlrewrite

Дополнительные сведения о текучих API-интерфейсах см. В Википедии на странице «Fluent interface»: http://ru.wikipedia.org/wiki/Fluent_interface

Дополнительные сведения об алгоритме MapReduce см. в следующих источниках:

- Википедия, статья «MapReduce»: http://ru.wikipedia.org/wiki/MapReduce
- Веб-сайт «The H», статья «Google patents Map/Reduce»: http://www.honline.com/open/news/item/Google-patents-Map-Reduce-908602.html

Дополнительные сведения о библиотеке параллельных задач (TPL) см. в статье «Библиотека параллельных задач» на MSDN: http://msdn.microsoft.com/ru-RU/library/dd460717.aspx

Дополнительные сведения о преимуществах использования библиотеки параллельных задач (TPL) вместо непосредственной работы с пулом потоков см. в следующих источниках:

- Статья «Оптимизация управляемого кода для многоядерных компьютеров» в *MSDN Magazine*: http://msdn.microsoft.com/ru-RU/magazine/cc163340.aspx
- Запись «Choosing Between the Task Parallel Library and the ThreadPool» в блоге «Parallel Programming with .NET»: http://blogs.msdn.com/b/pfxteam/archive/2009/10/06/9903475.aspx

Глава 6. Работа с данными в приложении Surveys

В этой главе описано, каким образом приложение Surveys использует данные. Сначала рассматривается модель данных, используемая приложением Surveys, а затем причины, по которым команда Tailspin выбрала данный метод со ссылкой на конкретные сценарии в приложении. Также в этой главе описан способ, с помощью которого разработчики обеспечили пригодность решения для тестирования. В заключение рассказывается, как и для чего приложение использует SQL Azure

Модель данных для мультитенантного приложения

В этом разделе описывается модель данных в приложении Surveys и объясняется, каким образом макет таблиц разделяет данные по клиентам.

Для хранения своих данных приложение Surveys использует как табличное хранилище, так и хранилище больших двоичных объектов. В разделах «Сохранение данных ответа на опрос» и «Разбиение на страницы результатов опроса» данной главы рассматриваются причины, по которым приложение использует хранилище больших двоичных объектов для некоторых данных. На рис. 1 показано на высоком уровне, какие данные в хранилищах какого типа хранятся.

Приложение Surveys использует хранилище больших двоичных объектов и табличное хранилище.

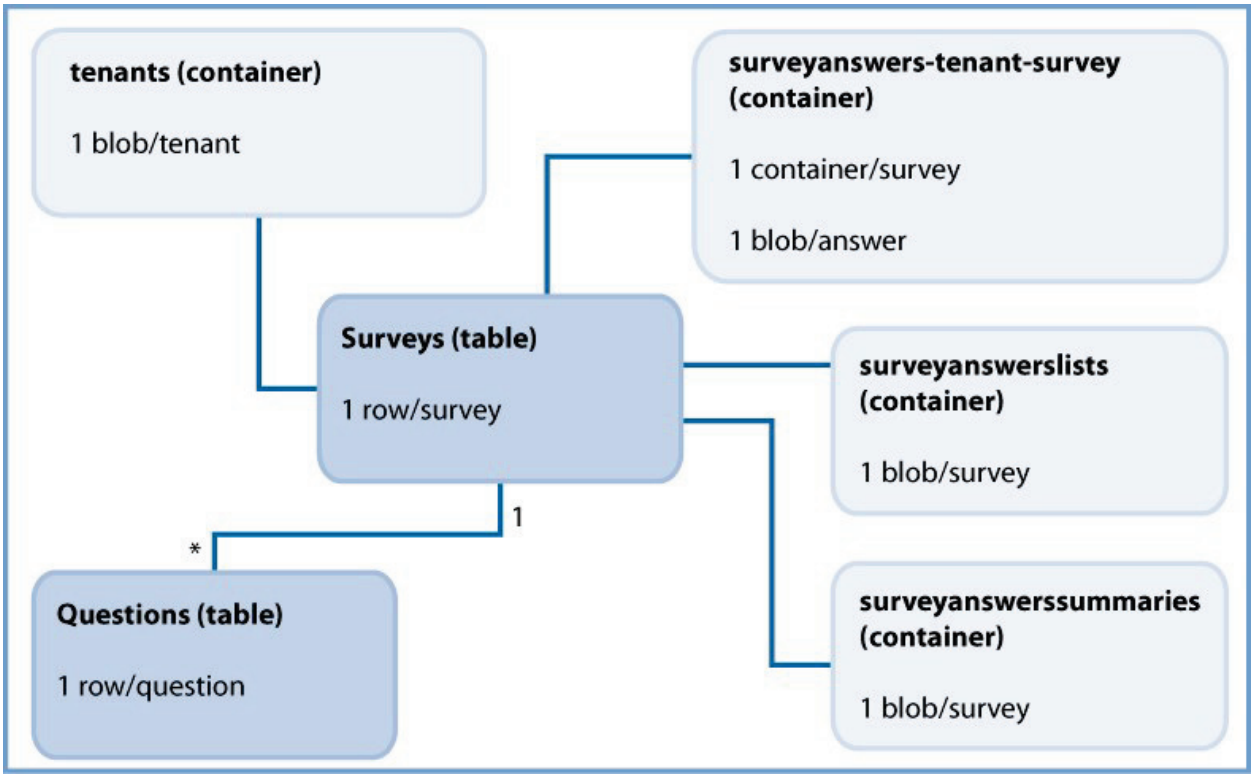


Рис. 1 Хранение данных в приложении Surveys

Хранение определений опроса

Приложение Surveys хранит определения опросов в двух таблицах Windows Azure. В данном разделе описываются эти таблицы и объясняется, почему компания Tailspin предпочла данное решение.

В следующей таблице рассмотрены поля таблицы Surveys. Эта таблица содержит список всех опросов в приложении.

Имя поля	Примечания
PartitionKey	В этом поле содержится имя клиента. Tailspin ввела это значение, чтобы иметь возможность быстрой фильтрации по имени клиента.
RowKey	
Timestamp	Табличное хранилище Windows Azure автоматически выставляет значение для этого поля.
SlugName	Имя опроса в «склеенном» виде.
CreatedOn	В этом поле отмечается, когда подписчик создал опрос. Это значение будет отличаться от значения Timestamp, если подписчик изменял опрос.

Title	Имя опроса.
PartitionKey	Это поле содержит имя клиента из поля PartitionKey таблицы Surveys, объединенное с именем опроса в «склеенном» виде. Это позволяет приложению вставлять все вопросы опроса в одну транзакцию и быстро выбирать все вопросы опроса из одной секции.
RowKey	Это поле содержит форматированный счетчик тактов, объединенный с положением вопроса в опросе. Это обеспечивает уникальность значения RowKey и определяет порядок вопросов.
Timestamp	Табличное хранилище Windows Azure автоматически выставляет значение для этого поля.
Text	Текст вопроса.
Тип	Тип вопроса: простой текст, множественный выбор или пять звезд (числовой диапазон).

Комментарий Маркуса:

Склеенное имя — это строка, в которой все пробелы и недопустимые символы заменены знаком дефиса (-). Этот термин не имеет ничего общего с конторским клеем!

В следующей таблице показаны поля таблицы «Questions». Приложение использует эту таблицу для хранения определений вопросов и формирования опроса.

Имя поля	Примечания
PossibleAnswers	Это поле содержит список возможных ответов, если вопрос имеет несколько вариантов ответа.

Комментарий Яны:

Помните, что табличное хранилище Windows Azure поддерживает транзакции только в пределах партиции в одной таблице.

Более детальное рассмотрение полей **RowKey** и **PartitionKey** в табличном хранилище Windows Azure см. В главе 5 («Этап 2. Автоматизация развертывания и использование хранилища Windows Azure») книги Миграция приложений в облако (или http://wag.codeplex.com/).

Хранение клиентских данных

Приложение Surveys сохраняет клиентские данные в хранилище больших двоичных объектов в контейнере с именем *tenants*. В следующем коде показан класс **Tenant**. Приложение сериализует экземпляры **Tenant** в большие двоичные объекты с идентификацией по имени подписчика (термин «tenant» (пользователь) используется в классах хранилищ для обозначения подписчиков).

C#

```
[Serializable]
public class Tenant
{
    public string ClaimType { get; set; }

    public string ClaimValue { get; set; }

    public string HostGeoLocation { get; set; }

    public string IssuerThumbPrint { get; set; }

    public string IssuerUrl { get; set; }

    public string Logo { get; set; }

    public string Name { get; set; }

    public string SqlAzureConnectionString { get; set; }

    public string DatabaseName { get; set; }

    public string DatabaseUserName { get; set; }
```

```
    public string DatabasePassword { get; set; }

    public string SqlAzureFirewallIpStart { get; set; }

    public string SqlAzureFirewallIpEnd { get; set; }
}
```

Приложение собирает большую часть данных подписчиков во время процесса регистрации. Свойство **Logo** содержит URL-адрес логотипа подписчика. Приложение хранит изображения логотипов в открытом контейнере больших двоичных объектов с именем *logos*.

Хранение ответов на опросы

Приложение Surveys сохраняет ответы на опросы в хранилище больших двоичных объектов. Приложение создает контейнер больших двоичных объектов для каждого опроса с именем по следующему шаблону: surveyanswers-*<имя пользователя>-<склеенное имя опроса>*. Это обеспечивает уникальность имени контейнера для каждого опроса.

Для каждого завершенного ответа на опрос приложение Surveys сохраняет большой двоичный объект в контейнер этого опроса. Имя большого двоичного объекта представляет собой счетчик тактов, производный от текущих даты и времени, что обеспечивает уникальность имени каждого большого двоичного объекта в контейнере. Содержимое каждого большого двоичного объекта представляет собой объект **SurveyAnswer**, сериализованный в формате JavaScript Object Notation (JSON). В следующем примере кода показаны классы **SurveyAnswer** и **QuestionAnswer**.

C#

```
public class SurveyAnswer
{
    …

    public string SlugName { get; set; }
     public string Tenant { get; set; }
     public string Title { get; set; }
    public DateTime CreatedOn { get; set; }
    public List<QuestionAnswer> QuestionAnswers { get;
set; }
}

public class QuestionAnswer
{
    public string QuestionText { get; set; }
    public QuestionType QuestionType { get; set; }

    [Required(ErrorMessage = "* You must provide an
answer.")]          public string Answer { get; set; }
    public string PossibleAnswers { get; set; }
}
```

Кроме того, приложение Surveys использует хранилище больших двоичных объектов для хранения упорядоченного списка ответов на каждый опрос. Для каждого опроса приложение сохраняет большой двоичный объект, содержащий сериализованный объект **List** с упорядоченным списком имен всех больших двоичных объектов ответов на опрос для данного опроса. Объект **List** сериализуется в формате JSON.
В разделе «Разбиение на страницы результатов опроса» данной главы объясняется, как приложение Surveys использует объекты **List** для разбиения на страницы результатов опроса.

Хранение сводок по ответам на опросы

Приложение Surveys хранит сводные статистические данные для каждого опроса в хранилище больших двоичных объектов. Для каждого опроса оно создает большой двоичный объект с именем *<имя пользователя>-<склеенное имя опроса>* в контейнере «surveyanswerssummaries». Для сохранения данных приложение сериализует объект **SurveyAnswersSummary** в формате JSON. В следующем примере кода показаны классы **SurveyAnswersSummary** и **QuestionAnswersSummary**, определяющие сводные данные.

C#

```
public class SurveyAnswersSummary
{
    …
    public string Tenant { get; set; }

    public string SlugName { get; set; }

    public int TotalAnswers { get; set; }

    public List<QuestionAnswersSummary>
QuestionAnswersSummaries
        { get; set; }
    …
}

public class QuestionAnswersSummary
{
    public string AnswersSummary { get; set; }

    public QuestionType QuestionType { get; set; }

    public string QuestionText { get; set; }

    public string PossibleAnswers { get; set; }
}
```

Обратите внимание, что сводка сохраняется как строка (string) для всех типов вопросов, в том числе и для числовых. Это позволяет минимизировать количество изменений, необходимых для добавления нового типа вопроса в приложение Surveys.

Классы хранения

Приложение Surveys использует классы хранения для управления хранилищем. В этом разделе кратко излагаются обязанности каждого из этих классов хранения.

Класс SurveyStore

Этот класс отвечает за сохранение определений опроса в хранилище таблиц и извлечение параметров опросов из хранилища таблиц.

Класс SurveyAnswerStore

Этот класс отвечает за сохранение ответов на опрос в хранилище больших двоичных объектов и извлечение ответов из этого хранилища. Он создает новый контейнер при сохранении первого ответа на новый опрос. Для отслеживания новых ответов на опрос используется очередь, которую приложение применяет для вычисления сводных статистических данных для опросов.

Этот класс также обеспечивает поддержку последовательного просмотра ответов на опрос.

Класс SurveyAnswersSummaryStore

Этот класс отвечает за сохранение сводных статистических данных опросов в большие двоичные объекты в контейнере *surveyanswerssummaries* и за извлечение этих данных.

Класс SurveySqlStore

Этот класс отвечает за сохранение данных ответов на опрос в SQL Azure. Дополнительные сведения см. в разделе «Использование SQL Azure» далее в этой главе.

Класс SurveyTransferStore

Этот класс отвечает за размещение сообщения в очереди, когда подписчик запрашивает приложение для сброса данных опроса в SQL Azure.

Класс TenantStore

Этот класс отвечает за сохранение и извлечение данных подписчиков и за сохранение загруженных логотипов. В примере кода этот класс создает некоторые данные по умолчанию для подписчиков Adatum и Fabrikam.

Тестирование хранилища Windows Azure

В этом разделе описывается, как при разработке и реализации классов хранения данных в приложении Surveys упростить механизм тестирования и обновления модулей, работающих с храненилищами.

Цели и требования

Приложение Surveys использует хранилище таблиц и больших двоичных объектов Windows Azure. Разработчикам компании Tailspin задумались о том, как это может повлиять на стратегию тестирования модулей. С точки зрения тестирования основное внимание при unit-тестировании следует сосредоточить на поведении определенного класса, а не на взаимодействии этого класса с другими компонентами приложения. С точки зрения Windows Azure любой тест, зависящий от хранилища Windows Azure, требует сложной настройки и дополнительно логики для обеспечения доступности корректных данных при выполнении тестирования. Вследствие этих причин разработчики компании Tailspin создали специальный модуль доступа к данным в приложении Surveys для целей тестирования. В частности, это средство позволяет выполнять тестирование классов хранения данных независимо от хранилища Windows Azure.

Решение

Решение, принятое разработчиками компании Tailspin, представляет собой специальный модуль (wrapper) для компонентов хранилища Windows Azure. Этот модуль позволяет заменять хранилище Windows Azure фиктивными объектами (mock object) при тестировании модулей и использовать контейнер Unity Application Block (Unity). Модульный тест должен иметь возможность создавать подходящий экземпляр фиктивного хранилища, использовать его во время теста, а затем удалять. Во всех комплексных тестах можно продолжать использовать исходные компоненты доступа к данным для тестирования функциональных возможностей приложения.

В приложении Surveys контейнер Unity используется для разделения компонентов и упрощения процесса тестирования.

Примечание.

Unity — это упрощенный, расширяемый контейнер внедрения зависимостей, который поддерживает перехват, внедрение конструктора, свойств и вызова методов. Контейнер Unity вы можете использовать различными способами для упрощения процесса отделения компонентов приложения, обеспечения максимальной согласованности компонентов и рационализации проектирования, реализации, тестирования и администрирования таких приложений. Дополнительные сведения о контейнере Unity см. на веб-сайте (там же можно загрузить этот контейнер) http://unity.codeplex.com/.

Реализация

Теперь самое время просмотреть код, подробно иллюстрирующий процесс тестирования классов хранения. В ходе изучения этого раздела, возможно, потребуется загрузить приложение Tailspin Surveys со страницы http://wag.codeplex.com/.

В этом разделе показано, каким образом проект приложения Surveys поддерживает модульное тестирование класса **SurveyStore**, который обеспечивает доступ к хранилищу таблиц. Основное внимание уделяется одному конкретному набору тестов, но и для других и других классов хранения приложение использует такой же подход.

В следующем примере кода показан интерфейс **IAzureTable** и класс **AzureTable**, которые являются основными компонентами реализации.

C#

```
public interface IAzureTable<T> где T: TableServiceEntity
{
    IQueryable<T> Query { get; }
    void EnsureExist();     void Add(T obj);
    void Add(IEnumerable<T> objs);
     void AddOrUpdate(T obj);
    void AddOrUpdate(IEnumerable<T> objs);
     void Delete(T obj);
    void Delete(IEnumerable<T> objs);
}
public class AzureTable<T>
             : IAzureTable<T> where T: TableServiceEntity
{
    private readonly string tableName;
    private readonly CloudStorageAccount account;

    …

    public IQueryable<T> Query
    {         get         {
            TableServiceContext context =
this.CreateContext();
             return context.CreateQuery<T>(this.tableName)
                .AsTableServiceQuery();
        }
    }
    public void Add(T obj)
    {
        this.Add(new[] { obj });
    }
    private TableServiceContext CreateContext()
    {
        return new TableServiceContext(
             this.account.TableEndpoint.ToString(),
              this.account.Credentials);
    }

    private class PartitionKeyComparer:
          IEqualityComparer<TableServiceEntity>
    {
        public bool Equals(TableServiceEntity x,
            TableServiceEntity y)
        {
            return string.Compare
             (x.PartitionKey, y.PartitionKey,
                  true,
               System.Globalization.CultureInfo
               .InvariantCulture) == 0;
        }
        public int GetHashCode(TableServiceEntity obj)
        {
            return obj.PartitionKey.GetHashCode();
        }
    }
}
```

Примечание.

В методе «Add», принимающем параметр «anIEnumerable», следует проверить количество элементов в пакете (batch) и общий размер пакета перед вызовом метода «SaveChanges» с параметром «SaveChangesOptions.Batch». Дополнительные сведения о пакетах и хранилищах таблиц Windows Azure см. в разделе «Транзакции в приложении aExpense», глава 5 «Этап 2: Автоматизация развертывания и использование хранилища Windows Azure», книга *Руководство по архитектуре Windows Azure. Часть 1.* Миграция приложений в облако или на странице http://wag.codeplex.com/.

Универсальный интерфейс и класс имеют параметр типа **T**, производный от типа Windows Azure **TableServiceEntity**, который используется для создания собственных типов таблиц. Например, в приложении Surveys типы **SurveyRow** и **QuestionRow** являются производными от класса **TableServiceEntity**. Интерфейс определяет несколько операций: метод **Query** возвращает коллекцию **IQueryable** типа **T**, методы **Add**, **AddOrUpdate** и **Delete** принимают параметр типа **T**. В классе **AzureTable** метод **Query** возвращает объект **TableServiceQuery**, методы **Add** и **AddOrUpdate** сохраняют объект в хранилище таблиц, а метод **Delete** удаляет объект из хранилища таблиц. Чтобы создать имитирующий объект для модульного тестирования, необходимо создать экземпляр объекта типа **IAzureTable**.

В следующем примере кода из класса **SurveyStore** показан конструктор.

C#

```
public SurveyStore(IAzureTable<SurveyRow> surveyTable,
IAzureTable<QuestionRow> questionTable)
{
    this.surveyTable = surveyTable;
    this.questionTable = questionTable;
}
```

Конструктор принимает параметры типа **IAzureTable**, который обеспечивает передачу в реальные или mock- объекты, реализующие интерфейс.

Параметризированный конструктор активизируется в двух различных сценариях. Приложение Surveys неявно вызывает конструктор, когда приложение использует класс **SurveysController** MVC. Для создания экземпляров контроллеров MVC приложение использует платформу внедрения зависимостей контейнера Unity. Приложение Surveys заменяет стандартную фабрику контроллеров MVC классом **UnityControllerFactory** в методе **OnStart** в обеих веб-ролях. Так что, когда приложению требуется новый экземпляр контроллера MVC, ответственность за его создание возлагается на контейнер Unity. В следующем примере кода показана часть класса **ContainerBootstrapper** из проекта TailSpin.Web, в котором контейнер Unity используется для определения способа создания экземпляров объектов.

C#

```
public static class ContainerBootstraper
{
    public static void RegisterTypes(IUnityContainer
container)
    {
        var account = CloudConfiguration
```

```
            .GetStorageAccount("DataConnectionString");
         container.RegisterInstance(account);

        container.RegisterType<ISurveyStore,
SurveyStore>();

        container.RegisterType<IAzureTable<SurveyRow>,
              AzureTable<SurveyRow>>(
             new InjectionConstructor(typeof
            (Microsoft.WindowsAzure.CloudStorageAccount),
            AzureConstants.Tables.Surveys));
```

```
        container.RegisterType<IAzureTable<QuestionRow>,
              AzureTable<QuestionRow>>(
             new InjectionConstructor(typeof
            (Microsoft.WindowsAzure.CloudStorageAccount),

            AzureConstants.Tables.Questions));
     …
    }
}
```

Последние два вызова метода **RegisterType** определяют правила, в соответствии с которыми контейнер Unity создает экземпляры **AzureTable**, которые необходимо передать в конструктор **SurveyStore**.

Когда приложению требуется новый экземпляр контроллера MVC, за создание контроллера отвечает контейнер Unity. Что касается класса **SurveysController**, контейнер Unity создает экземпляр объекта **SurveyStore** с помощью параметризованного конструктора, показанного ранее, и передает объект **SurveyStore** в конструктор **SurveysController**.

Во втором сценарии использования для параметризованного конструктора **SurveyStore** создаются модульные тесты для класса **SurveyStore** с помощью непосредственного вызова конструктора и передачи параметров в имитирующие объекты. В следующем примере кода показан метод модульного теста, использующий конструктор таким образом.

C#

```
[TestMethod]
public void
GetSurveyByTenantAndSlugNameReturnsTenantNameFromPartitionKey()
{
    string expectedRowKey = string.Format(
        CultureInfo.InvariantCulture, "{0}_{1}", "tenant",
        "slug-name");
    var surveyRow = new SurveyRow { RowKey = expectedRowKey,
          PartitionKey = "tenant" };
    var surveyRowsForTheQuery = new[] { surveyRow };
     var mock = new Mock<IAzureTable<SurveyRow>>();
     mock.SetupGet(t => t.Query)
       .Returns(surveyRowsForTheQuery.AsQueryable());
```

```
     var store = new SurveyStore(mock.Object,
         default(IAzureTable<QuestionRow>));

    var survey = store.GetSurveyByTenantAndSlugName("tenant",
         "slug-name", false);

    Assert.AreEqual("tenant", survey.Tenant);
}
```

В ходе выполнения тестирования создается фиктивный экземпляр **IAzureTable<SurveyRow>**. Используйте этот экземпляр для создания экземпляра объекта **SurveyStore**, вызова метода **GetSurveyByTenantAndSlugName** и проверки результатов. Он выполняет этот тест без обращения к хранилищу таблиц Windows Azure.

Такой же подход приложение Surveys применяет для модульного тестирования других компонентов хранения, которые используют хранилище больших двоичных объектов и таблиц Windows Azure.

Сохранение данных ответов на опрос

После того как пользователь завершает опрос, приложение должно сохранить ответы пользователя на вопросы опроса в хранилище, где создатель опроса сможет оценить и проанализировать результаты.

Цели и требования

Формат, который приложение использует для хранения сводных данных ответов, должен обеспечивать соответствие приложения Surveys следующим трем требованиям.

- Владелец опроса должен иметь возможность просмотра результатов.
- Приложение должно иметь возможность рассчитывать сводную статистику по ответам.
- Владелец опроса должен иметь возможность экспортировать ответы в формате, который позволяет подробно анализировать результаты.

В компании Tailspin полагают, что в опросе примет участие большое количество пользователей, поэтому необходимо использовать максимально эффективный процесс первоначального сохранения данных. Приложение может обрабатывать любые данные после их сохранения с помощью асинхронного рабочего процесса. Дополнительные сведения о разработке функциональной возможности фоновой обработки данных в приложении Surveys см. в разделе «Масштабирование приложения Surveys», глава 5 «Построение масштабируемого многопользовательского приложения для Windows Azure» ранее в этой книге.

Основное внимание здесь уделяется способу хранения ответов на опрос в приложении Surveys. Какой бы тип хранения не использовало приложение Surveys, он должен соответствовать трем перечисленным ранее требованиям. Затраты на хранение также являются важным фактором в выборе типа хранения, так как большая часть требований к приложению по хранению определяется ответами на опрос как с точки зрения используемого пространства, так и по количеству транзакций в хранилище.

Комментарий Яны:

Транзакционные издержки будут значительными, поскольку для расчета сводных статистических данных и экспорта результатов опроса приложению потребуется считывать ответы на опрос из хранилища.

Решение

Для удовлетворения требований разработчики компании Tailspin проанализировали два возможных решения для хранения данных: шаблон отложенной записи с использованием очередей и хранилища таблиц, а также решение, сохраняющее данные непосредственно в хранилище больших двоичных объектов. В обоих случаях приложение сначала сохраняет ответы на опрос в хранилище, а затем использует асинхронную задачу рабочей роли для расчета и сохранения сводной статистики.

Приложение Surveys сохраняет каждый ответ на опрос в виде большого двоичного объекта.

Решение 1. Шаблон отложенной записи

На рис. 2 показан шаблон отложенной записи, который приложение Surveys может использовать для сохранения результатов заполненного опроса в хранилище таблиц Windows Azure.

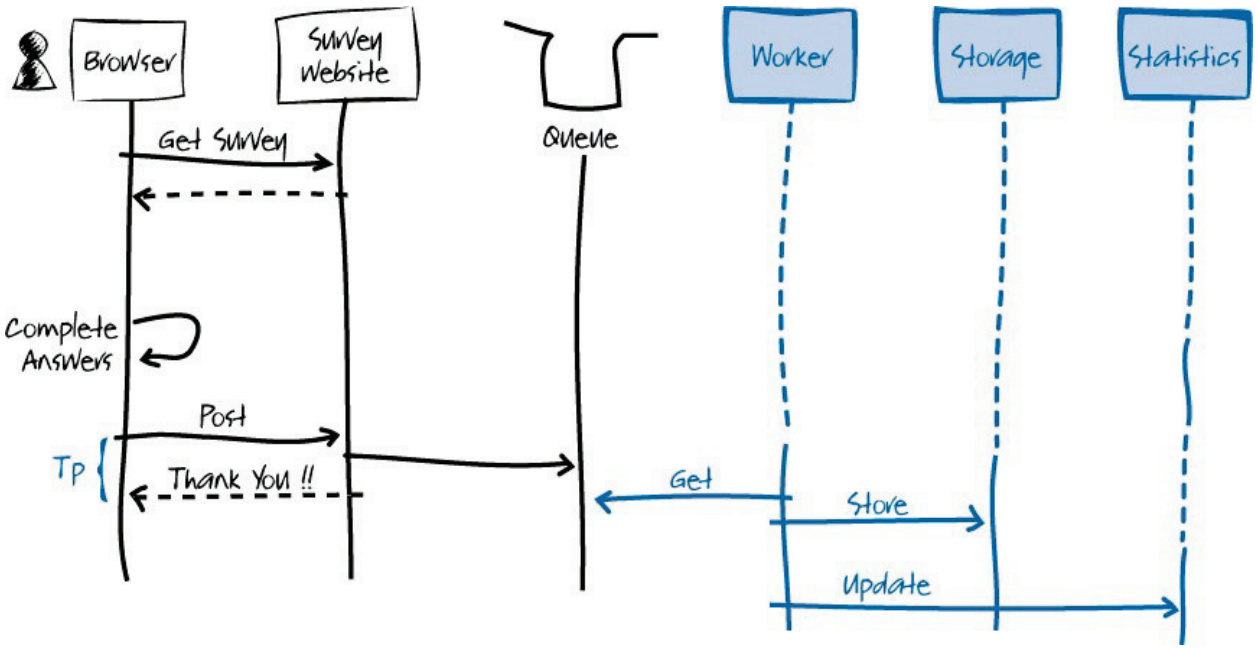


Рис. 2 Шаблон отложенной записи для сохранения ответов на опрос в приложении Surveys

В этом сценарии пользователь просматривает опрос, заполняет его и затем отправляет опрос со своими ответами обратно на веб-сайт Surveys. Веб-сайт Surveys вставит ответы на опрос в сообщение в очереди и возвращает как можно скорее пользователю сообщение «Благодарим за ответы», сводя к минимуму значение **Tp** на рис. 2. Задача в рабочей роли отвечает за считывание ответов на опрос из очереди и их

сохранение в хранилище таблиц. Чтобы исключить любую возможность двойного учета и искажения результатов, эта операция должна быть идемпотентной.

Примечание.

Можно использовать отдельные рабочие роли: одну для вычисления и сохранения сводной статистики, а другую для сохранения результатов опроса в хранилище таблиц на случай, если потребуется масштабировать приложение.

Комментарий Бхарата:

Приложение Surveys учитывает географическое расположение. Например, веб-сайт и очередь Surveys могут размещаться в центре обработки данных в США, а рабочая роль и хранилище таблиц — в центре обработки данных в Европе.

Максимальный размер сообщения в очереди Windows Azure составляет 8 килобайт (КБ). Поэтому такой подход можно применять только в случае, если ответ на опрос не превышает максимальный размер. На рис. 3 показано, как можно изменить это решение для обработки результатов опроса, размер которых превышает 8 КБ.

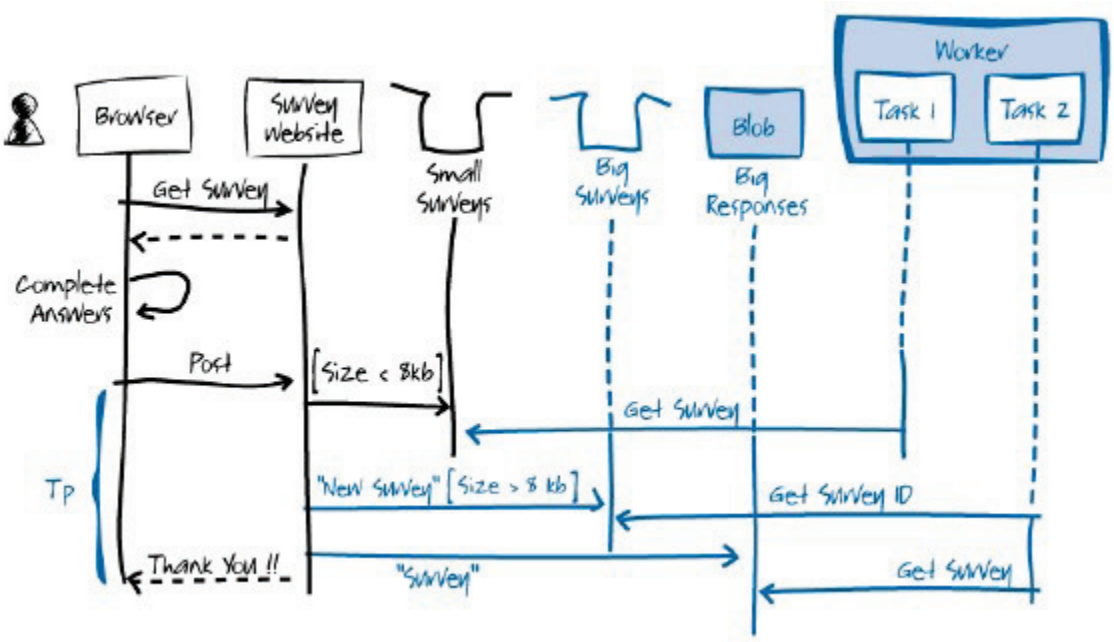


Рис. 3 **Обработка результатов опроса размером более 8 КБ**

На рис. 3 представлено оптимизированное решение, то есть приложение помещает сообщения, размер которых меньше 8 КБ, непосредственно в очередь, как и в предыдущем примере. Сообщения размером более 8 КБ приложение сохраняет в хранилище больших двоичных объектов Windows Azure и помещает сообщение в очередь «Big Surveys» для уведомления рабочей роли. Теперь рабочая роль содержит две задачи. Задача 1 — извлечение и обработка небольших опросов из очереди «Small Surveys». Задача 2 — опрос очереди «Big Surveys» для уведомления больших опросов, извлекающих данные из хранилища больших двоичных объектов и обрабатывающих их.

Комментарий Маркуса:

При вычислении размера сообщений необходимо учитывать влияние кодирования, например Base64, которое используется для кодирования данных перед их размещением в сообщении.

Решение 2. Запись непосредственно в хранилище больших двоичных объектов

Как было показано в предыдущем разделе, шаблон отложенной записи становится более сложным, если размер ответа на опрос превышает 8 КБ. В этом случае необходимо сохранить ответ в виде большого двоичного объекта и уведомить рабочую роль о новых данных ответа с помощью сообщения в очереди. Разработчики компании Tailspin также проанализировали более простой подход к сохранению и обработке запросов на опросы с использованием только хранилища больших двоичных объектов. На рис. 4 представлен этот альтернативный вариант.

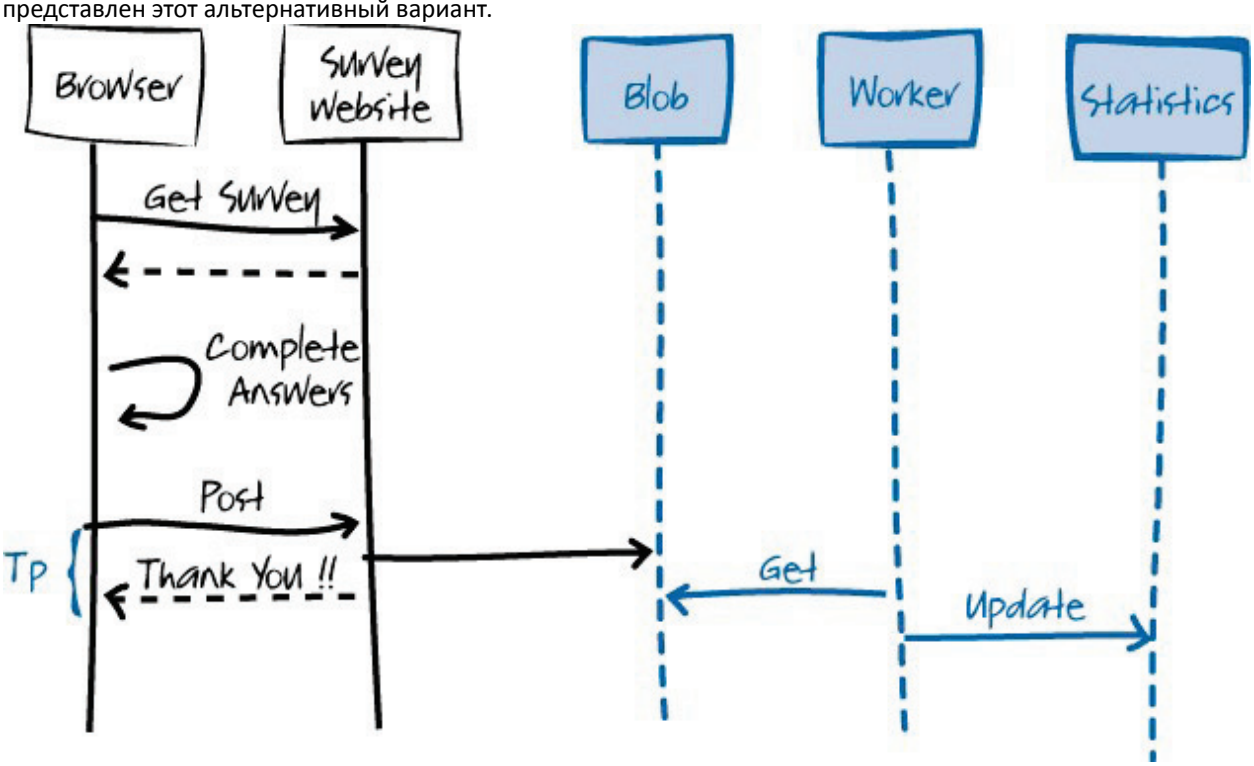


Рис. 4 Сохранение ответов на опрос непосредственно в хранилище больших двоичных объектов

Как видно из диаграммы последовательности, представленной на рис. 4, на первых этапах сохранения ответов на опрос процесс аналогичен шаблону отложенной записи. При таком подходе без использования очереди и хранилища таблиц приложение сохраняет результаты опроса непосредственно в хранилище больших двоичных объектов. Рабочая роль теперь создает сводные статистические данные непосредственно на основе ответов на опрос, находящихся в хранилище больших двоичных объектов.

На рис. 5 показан вариант такого сценария, где подписчик решил разместить опрос в стороннем центре обработки данных из своей учетной записи.

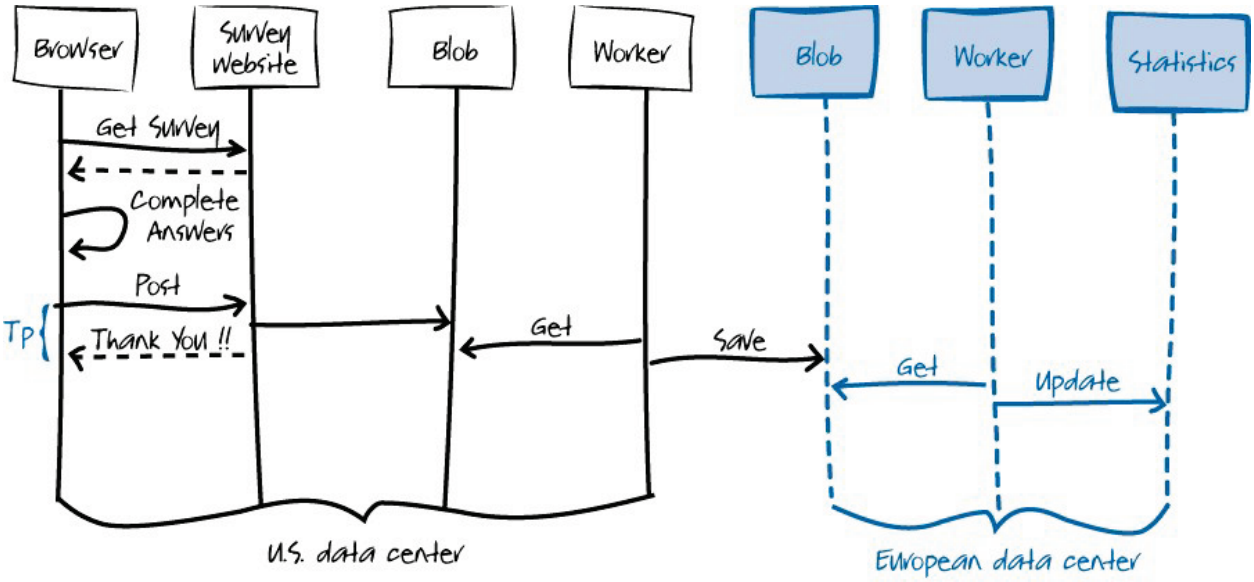


Рис. 5 **Сохранение ответов на опрос, размещенный в стороннем центре обработки данных**

В этом сценарии используется дополнительная рабочая роль. Эта рабочая роль несет ответственность за перемещение данных ответов на опрос из центра обработки данных, где подписчик разместил опрос, в центр обработки данных, где размещена учетная запись подписчика. Таким образом, приложение передает данные опроса между центрами обработки данных только один раз, а не каждый раз, когда приложению требуется считать данные. Это позволяет свести к минимуму расходы, связанные с этим сценарием.

Комментарий Яны:

Приложения считывает данные ответов на опрос при вычислении статистических данных, когда пользователь просматривает ответы и когда выполняется экспорт данных в SQL Azure.

Сравнение решений

Второе решение гораздо проще, чем первое. Но также необходимо убедиться, что хранение ответов на опрос в больших двоичных объектах вместо таблиц не повышает уровень сложности процессов, использующих эти результаты опроса. В программе Surveys использование больших двоичных объектов не приводит к значительному повышению уровня сложности при формировании сводной статистики. Это позволяет владельцу опроса просматривать ответы или экспортировать данные в SQL Azure.

Хотя второе решение не ограничивает функциональные возможности, необходимые для приложения Surveys, эта архитектура может иметь ограничения для других сценариев. Использование шаблона отложенной записи означает, что вы можете легко выполнять операции с данными до их сохранения в таблице. Поэтому в сценариях, где для использования необработанных данных их необходимо обработать, первое решение является предпочтительным. Во-вторых, хранение данных в таблицах упрощает доступ к данным для динамически создаваемых запросов.

Шаблон отложенной записи позволяет преобразовывать данные перед их сохранением без влияния на производительность веб-роли.

Третье отличие между решениями — стоимость хранения. В следующей таблице подведены итоги для этого различия и показано число транзакций в хранилище, которые должно выполнить приложение для сохранения одного ответа на опрос.

Решение 1	Решение 2
Шаблон отложенной записи	**Запись непосредственно в хранилище больших двоичных объектов**
1 сохранение в большой двоичный объект	1 сохранение в большой двоичный объект
1 добавление сообщения в очередь	
1 получение сообщения из очереди	
1 считывание большого двоичного объекта	
1 сохранение в таблицу	
Всего 5 транзакций в хранилище	Всего 1 транзакция в хранилище

Комментарий Бхарата:

Также следует убедиться, что второе решение не увеличивает количество транзакций в хранилище, необходимых приложению для обработки или использования сохраненных данных.

Реализация

Теперь самое время подробно рассмотреть код, сохраняющий ответы на опрос. В ходе изучения этого раздела, возможно, потребуется загрузить Tailspin Surveys со страницы http://wag.codeplex.com/.

Сохранение данных ответов на опрос во временный большой двоичный объект

В следующем коде из класса **SurveysController** в проекте TailSpin.Web.Survey.Public показано, как приложение инициирует асинхронное сохранение ответа на опрос.

C#

```
 [HttpPost]
public ActionResult Display(string tenant, string
surveySlug, SurveyAnswer contentModel)
{
    var surveyAnswer = CallGetSurveyAndCreateSurveyAnswer(
         this.surveyStore, tenant, surveySlug);

    …

    for (int i = 0; i < surveyAnswer.QuestionAnswers.Count;
i++)
    {
        surveyAnswer.QuestionAnswers[i].Answer =
              contentModel.QuestionAnswers[i].Answer;
    }

    if (!this.ModelState.IsValid)
```

```
    {
        var model =
            new
TenantPageViewData<SurveyAnswer>(surveyAnswer);
         model.Title = surveyAnswer.Title;
         return this.View(model);
    }

    this.surveyAnswerStore.SaveSurveyAnswer(surveyAnswer);

    return this.RedirectToAction("ThankYou");
}
```

Переменная **surveyAnswerStore** содержит ссылку на экземпляр типа **SurveyAnswerStore**. Приложение использует контейнер Unity для инициализации данного экземпляра с корректными экземплярами **IAzureBlob** и **IAzureQueue**. Контейнер больших двоичных объектов хранит ответы на вопросы опроса. В очереди хранится список новых ответов на опрос, которые еще не включены в сводную статистику или список ответов на опрос.

Метод **SaveSurveyAnswer** записывает данные ответов на опрос в хранилище больших двоичных объектов и помещает сообщение в очередь. После этого метод действия немедленно возвращает сообщение «Благодарим за ответы».

Комментарий По:

Убедитесь, что строки подключения к хранилищу в развертывании указывают на хранилище в географическом местоположении развертывания. Чтобы свести к минимуму задержки, в приложении следует использовать локальные очереди и хранилище больших двоичных объектов.

В следующем примере кода показан метод **SaveSurveyAnswer** в классе **SurveyAnswerStore**.

C#

```
public void SaveSurveyAnswer(SurveyAnswer surveyAnswer)
{
    var surveyAnswerBlobContainer=
        this.surveyAnswerContainerFactory
        .Create(surveyAnswer.Tenant,
surveyAnswer.SlugName);
    surveyAnswerBlobContainer.EnsureExist();
     DateTime now = DateTime.UtcNow;
     surveyAnswer.CreatedOn = now;
     var blobId = now.GetFormatedTicks();
    surveyAnswerBlobContainer.Save(blobId, surveyAnswer);
     this.surveyAnswerStoredQueue.AddMessage(
         new SurveyAnswerStoredMessage
            {
                SurveyAnswerBlobId = blobId,
                Tenant = surveyAnswer.Tenant,
                SurveySlugName = surveyAnswer.SlugName
            });
}
```

Этот метод сначала проверяет наличие контейнера больших двоичных объектов и при необходимости создает его. Затем метод создает уникальный идентификатор большого двоичного объекта с помощью счетчика тактов и сохраняет большой двоичный объект в контейнер опроса. В заключение метод добавляет сообщение в очередь. Приложение использует очередь для отслеживания новых ответов на опрос, которые необходимо включить в сводную статистику и список ответов для разбиения ответов на страницы.

Примечание.

Возможно, но весьма маловероятно, что приложение попытается сохранить два больших двоичных объекта с одним идентификатором, если два пользователя завершили опрос в одно и то же время. Код должен проверить эту возможность и при необходимости повторить попытку сохранения с новым значением счетчика тактов.

Комментарий Маркуса:

Возможна ошибка роли после добавления данных опроса в хранилище больших двоичных объектов и перед добавлением сообщения в очередь. В этом случае данные ответов не включаются в сводную статистику или в список ответов, используемых для разбиения на страницы. Однако ответ будет включен, если пользователь экспортировал опрос в SQL Azure.

Отображение данных

В этом разделе описываются несколько интересных сценариев в приложении Surveys, демонстрирующих отображение данных для пользователей и каким образом базовая модель данных поддерживает функциональную возможность отображения.

Разбиение результатов опроса на страницы

Владелец опроса должен иметь возможность просматривать результаты опроса с одновременным отображением одного ответа на опрос. Эта особенность дополняет возможность просмотра сводных статистических данных или возможность анализировать результаты, используя SQL Azure. Для этой функции приложение Surveys содержит страницу «Просмотр ответов».

Цели и требования

Техническое решение этой функции приложения должно удовлетворять двум конкретным требованиям.

Первое требование — приложение должно отображать ответы на опрос в порядке их исходной отправки.

Второе требование — следить за тем, чтобы эта функция не влияла на производительность веб-роли.

Решение

Разработчики компании Tailspin рассмотрели два решения, основанных на различной модели хранения. Первый параметр предполагал, что приложение хранит данные ответов на опрос в хранилище таблиц. Второй параметр, который и был выбран, предполагал, что приложение хранит данные ответов на опрос в хранилище больших двоичных объектов.

Разбиение на страницы с использованием хранилища таблиц

Разработчики компании Tailspin внимательно изучили две особенности API-интерфейса хранилища таблиц Windows Azure. Первая особенность — дополнительный маркер, который можно получить из запроса, позволяющего выполнить последующий запрос, который запускается после завершения предыдущего запроса. Можно использовать структуру данных стека для ведения списка дополнительных маркеров, который применяется для перехода на одну страницу вперед или назад в ответах на опрос. Затем необходимо сохранить этот стек дополнительных маркеров в состоянии сеанса пользователя, чтобы облегчить навигацию для пользователей.

Пример такого подхода см. в разделе «Реализация разбиения на страницы с использованием хранилища таблиц Windows Azure», глава 8 «Этап 4».

Добавление задач и настройка на странице http://wag.codeplex.com/ приложения», книга Миграция приложений в облако или

Второй полезной особенностью API-интерфейса является способность выполнения асинхронных запросов к хранилищу таблиц Windows Azure. Это помогает избежать нехватки в пуле потоков веб-сервера в веб-роли за счет выгрузки трудоемких задач в фоновый поток.

Разбиение на страницы с использованием хранилища больших двоичных объектов

В этом решении предполагается, что каждый ответ на опрос хранится в отдельном большом двоичном объекте. Для доступа к большим двоичным объектам в стандартном порядке необходимо вести список всех больших двоичных объектов. Затем этот список вы можете использовать для последовательного определения идентификатора предыдущих и следующих больших двоичных объектов и для предоставления пользователям средства навигации назад и вперед по ответам на опрос.

Для поддержки альтернативного упорядочивания данных необходимо вести дополнительные списки.

Сравнение решений

В предыдущем разделе, где рассматривались альтернативные варианты сохранения данных ответов на опрос, более низкие транзакционные издержки были определены как ключевое преимущество сохранения непосредственно в хранилище больших двоичных объектов вместо использования шаблона отложенной записи. Разбиение на страницы с использованием хранилища таблиц — это сложный процесс, поскольку необходимо управлять дополнительным стеком в состоянии сеанса пользователя.
Однако необходимо учитывать затраты и трудности, связанные с ведением упорядоченного списка больших двоичных объектов, присущие второму из альтернативных решений. Это влечет за собой две дополнительных транзакции в хранилище для каждого нового опроса: одна — извлечение списка из хранилища больших двоичных объектов, другая — сохранение списка обратно в хранилище больших двоичных объектов. Это все еще составляет меньшее количество транзакций за ответ на опрос по сравнению с табличным решением. Кроме того, можно избежать использования состояния сеанса за счет встраивания ссылок на следующие и предыдущие BLOB объекты непосредственно в ссылки на веб-странице.

Комментарий Бхарата:

То, что вначале кажется очевидным решением (в данном случае использование хранилищ таблиц), не всегда является лучшим.

Реализация

Теперь самое время подробно рассмотреть функцию разбиения данных на страницы, реализованную компанией Tailspin. В ходе изучения этого раздела, возможно, потребуется загрузить решение Surveys со страницы http://wag.codeplex.com/.

Данное пошаговое руководство разделено на два раздела. В первом разделе описывается, каким образом приложение ведет упорядоченный список больших двоичных объектов. Во втором разделе рассматривается, как приложение использует этот список для разбиения ответов на страницы.

Ведение упорядоченного списка ответов на опрос

Приложение Surveys еще использует в рабочей роли асинхронную задачу для вычисления сводных статистических данных по каждому опросу. Эта задача периодически обрабатывает новые ответы на опрос из очереди. При обработке каждого ответа обновляется упорядоченный список больших двоичных объектов, содержащих результаты опроса. При сохранении приложение назначает для каждого большого двоичного объекта идентификатор на основе счетчика тактов, а также добавляет сообщение в очередь для отслеживания новых ответов на опрос.

Для ведения упорядоченного списка больших двоичных объектов приложение Surveys использует асинхронную задачу в рабочей роли.

В следующем примере кода из класса **SurveyAnswerStore** показано, каким образом приложение создает идентификатор большого двоичного объекта, сохраняет этот объект в корректном контейнере больших двоичных объектов для опроса и добавляет сообщение в очередь, отслеживающую новые ответы на опрос.

C#

```
public void SaveSurveyAnswer(SurveyAnswer surveyAnswer)
{
    var surveyBlobContainer =
this.surveyAnswerContainerFactory
         .Create(surveyAnswer.Tenant,
surveyAnswer.SlugName);
     surveyBlobContainer.EnsureExist();
     DateTime now = DateTime.UtcNow;
     surveyAnswer.CreatedOn = now;
     var blobId = now.GetFormatedTicks();
    surveyBlobContainer.Save(blobId, surveyAnswer);
     this.surveyAnswerStoredQueue.AddMessage(
         new SurveyAnswerStoredMessage
            {
                SurveyAnswerBlobId = blobId,
                Tenant = surveyAnswer.Tenant,
                SurveySlugName = surveyAnswer.SlugName
            });
}
```

Метод **Run** в классе **UpdatingSurveyResultsSummaryCommand** в рабочей роли вызывает метод **AppendSurveyAnswerIdToAnswerList** для каждого ответа на опрос в очереди новых ответов.

В следующем примере показан код метода **AppendSurveyAnswerIdToAnswerList** в классе **SurveyAnswerStore**.

C#

```
public void AppendSurveyAnswerIdToAnswersList(string
tenant,
      string slugName, string surveyAnswerId)
{
    string id = string.Format(CultureInfo.InvariantCulture,
        "{0}-{1}", tenant, slugName);
    var answerIdList =
this.surveyAnswerIdsListContainer.Get(id)
        ?? new List<string>(1);
answerIdList.Add(surveyAnswerId);
    this.surveyAnswerIdsListContainer.Save(id,
answerIdList);
}
```

Приложение хранит список ответов на опрос в объекте **List**, который сериализуется в формате JSON и сохраняется в большом двоичном объекте. Для каждого опроса используется один большой двоичный объект.

Комментарий Маркуса:

Приложение добавляет новые ответы в очередь в порядке их поступления. Когда приложение получает сообщения из очереди и добавляет идентификаторы больших двоичных объектов в список, сохраняется первоначальный порядок.

Реализация разбиения на страницы

Когда приложение Surveys отображает ответ на опрос, обнаруживается большой двоичный объект, содержащий ответ на опрос, с помощью идентификатора большого двоичного объекта. Приложение может использовать упорядоченный список идентификаторов больших двоичных объектов, чтобы создать ссылки для навигации по предыдущим и следующим ответам на опрос.

В следующем примере кода показан метод действия **BrowseResponses** в классе **SurveysController** проекта TailSpin.Web.

C#

```
public ActionResult BrowseResponses(string tenant,
     string surveySlug, string answerId)
{
    SurveyAnswer surveyAnswer = null;     if
(string.IsNullOrEmpty(answerId))
    {
        answerId = this.surveyAnswerStore
            .GetFirstSurveyAnswerId(tenant, surveySlug);
```

```
    }

    if (!string.IsNullOrEmpty(answerId))
    {
        surveyAnswer = this.surveyAnswerStore
            .GetSurveyAnswer(tenant, surveySlug, answerId);
    }

    var surveyAnswerBrowsingContext =
this.surveyAnswerStore
        .GetSurveyAnswerBrowsingContext(tenant,
         surveySlug, answerId);

    var browseResponsesModel = new BrowseResponseModel
        {
            SurveyAnswer = surveyAnswer,
            PreviousAnswerId =
                surveyAnswerBrowsingContext.PreviousId,
            NextAnswerId =
surveyAnswerBrowsingContext.NextId
        };

    var model = new TenantPageViewData<BrowseResponseModel>
        (browseResponsesModel);     model.Title =
surveySlug;     return this.View(model);
}
```

Этот метод действия использует метод **GetSurveyAnswer** в классе **SurveyAnswerStore** для получения ответов на опрос из хранилища больших двоичных объектов и метод **GetSurveyAnswerBrowsingContext** для извлечения объекта **SurveyBrowsingContext**, содержащего идентификаторы следующего и предыдущего большого двоичного объекта в последовательности. Затем этот метод заполняет объект модели данными для передачи в представление.

Хранилище данных сеанса

Приложение Surveys должно хранить некоторые данные состояния для каждого пользователя, который создает опросы. В этом разделе описываются разработка и реализация механизма управления состояниями пользователей в приложении Surveys.

Приложение Surveys должно хранить состояние сеанса, когда пользователь разрабатывает опрос.

Цели и требования

Когда пользователи разрабатывают новые опросы в приложении Surveys, они создают опрос, а затем по одному добавляют в него нужные вопросы. На рис. 6 показана последовательность экранов, когда пользователь создает опрос с двумя вопросами.

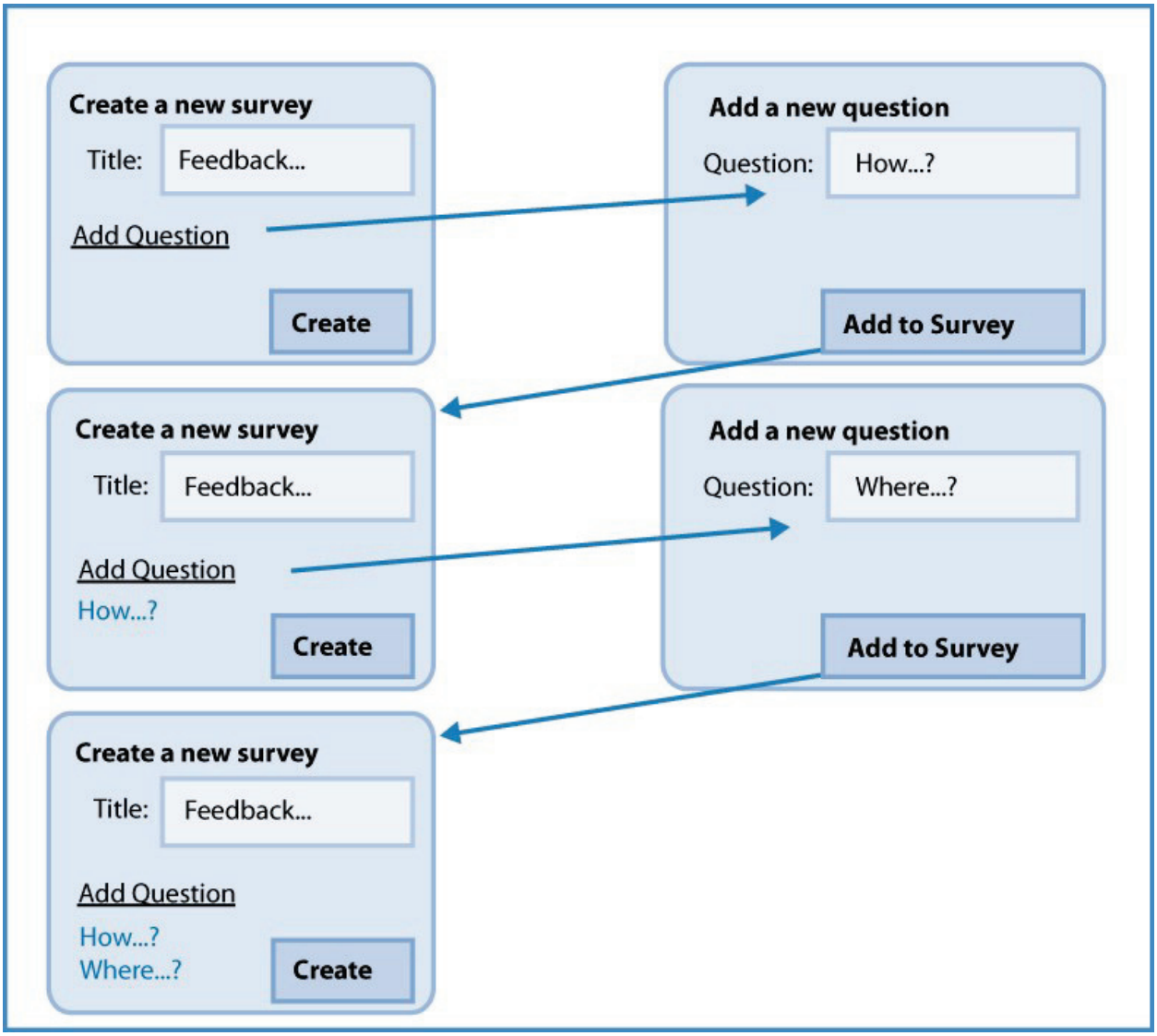


Рис. 6 Создание опроса с двумя вопросами

Как видно на диаграмме, этот сценарий включает два разных окна, для которых приложение должно сохранять состояние, когда пользователь добавляет вопросы в опросный лист. Разработчики компании Tailspin рассмотрели три параметра управления состоянием сеанса.

- Использовать JavaScript и управлять всем рабочим процессом на клиентском компьютере. Затем использовать вызовы Ajax для отправки готового опроса на сервер.
- Использовать стандартный встроенный объект **Request.Session** для хранения промежуточного состояния опроса, пока пользователь создает этот опрос. Поскольку веб-роль Tailspin будет работать на нескольких экземплярах узлов, компания не может использовать поставщик состояний сеансов в памяти по умолчанию, и ей придется использовать другой поставщик хранилища, например поставщик состояний сеансов, включенный в службу кэширования Windows Azure AppFabric. Дополнительные сведения о службе кэширования AppFabric см. по адресу http://msdn.microsoft.com/ru-RU/library/gg278356.aspx.
- Использовать подход, схожий с механизмом ViewState, который сериализует и выполняет десериализацию состояния рабочего процесса и передает его между двумя страницами.

Решение

Эти три варианта вы можете сравнить по нескольким критериям. Важность критериев будет зависеть от конкретных требований приложения.

Простота

Все, что просто внедрить, обычно просто и обслуживать. Первый вариант является самым сложным из трех. Он требует знания JavaScript и хорошее знание библиотеки Ajax. Сложно проводить и модульные тесты. Проще всего осуществить второй параметр, так как используется стандартный объект ASP.NET **Session**. Чтобы использовать поставщик состояний сеансов, достаточно «подключиться» к поставщику состояний сеансов службы кэширования Windows Azure AppFabric в файле конфигурации Web.config. Третий вариант имеет среднюю сложность, но его реализацию можно упростить, используя некоторые функции в ASP.NET MVC 3. В отличие от второго варианта, он не требует установки или настройки на стороне сервера, кроме стандартной настройки MVC 3.

Хотя второй вариант легко реализовать, он создает некоторые проблемы с долгосрочным обслуживанием приложения. Текущая версия службы кэширования AppFabric не позволяет отключить вытеснение данных для кэша, поэтому, если кэш полностью заполняется, данные сеанса могут быть удалены из кэша, когда сеанс все еще активен. Программное обеспечение Tailspin должно контролировать использование кэша и выявлять ситуации, когда из кэша были вытеснены элементы активного сеанса. При необходимости Tailspin может затем увеличить размер используемого кэша. Если необходимо удалить какие-то элементы из кэша, используется политика LRU, то есть удаляются элементы, которые дольше всего не использовались.

Комментарий Яны:

Если в будущем Tailspin решит использовать кэш AppFabric для других целей в приложении Surveys, это может увеличить нагрузку на кэш и повысить вероятность вытеснения элементов из кэша.

Стоимость

Первый вариант является самым экономичным, так как в нем используется только одно сообщение POST для отправки готового опроса на сервер. Второй вариант требует умеренных затрат, которые зависят от размера кэша, выбранного Tailspin (текущие тарифы можете узнать на веб-сайте Windows Azure). Расходы по третьему варианты зависят от используемой пропускной способности: Tailspin может оценить расходы на основе ожидаемого количества вопросов, созданных за день, и среднего размера вопросов.

Комментарий Яны:

В третьем параметре данные переводятся в кодировку Base64, и это необходимо учитывать при любой оценке среднего размера вопроса.

Производительность

Первый вариант обеспечивает наивысшую производительность, так как клиент выполняет почти всю работу без обращений к серверу до заключительного сообщения POST, содержащего готовый опрос. Второй вариант создаст небольшую задержку в приложении. Величина задержки будет зависеть от количества одновременных сеансов, количества данных в объектах сеанса и задержки между веб-ролью и кэшем Windows Azure AppFabric. Третий вариант также внесет некоторую задержку, так как для каждого вопроса потребуется обращение к серверу и получение ответа, а каждое сообщение HTTP-запроса и HTTP-ответа будет включать все текущие данные состояния.

Масштабируемость

Все три варианта хорошо масштабируются. Первый вариант хорошо масштабируется, так как не требует хранить какие-либо данные о состоянии сеанса вне браузера. Второй и третий варианты хорошо масштабируются, так как удобны для использования в веб-ферме и допускают развертывание в нескольких веб-ролях.

Надежность

Первый вариант наименее надежен, так как зависит от «хрупкого» кода JavaScript. Второй вариант надежен, так как использует функцию, которая является частью стандартных служб Windows Azure AppFabric. Третий вариант также надежен, так как в нем используется легко тестируемый код на стороне сервера.

Удобство работы пользователей

Первый вариант наиболее удобен для пользователей, так как не требует обратных передач в процессе создания опроса. Два других вариантов требуют обратной передачи для каждого вопроса.

Безопасность

Первые два варианты обеспечивают высокую безопасность. В первом варианте браузер сохраняет весь опрос в памяти, пока он не будет готов, а во втором варианте браузер сохраняет только куки-файл с идентификатором сеанса, а данные опроса хранятся в кэше службы кэширования AppFabric. Третий вариант не так безопасен, так как просто сериализует данные в Base64 без шифрования. Есть вероятность «утечки» конфиденциальных данных во время передачи между страницами.

Компания Tailspin решила использовать второй вариант, где используется поставщик состояний сеансов, включенный в службу кэширования AppFabric. Это решение соответствует критериям Tailspin для этой части приложения Tailspin Surveys.

Реализация

Теперь самое время подробно рассмотреть реализацию хранилища данных сеанса, выбранную компанией Tailspin. В ходе изучения этого раздела может потребоваться загрузить решение Tailspin Surveys со страницы http://wag.codeplex.com/.

В следующем примере кода показано, как методы действия (Action) в классе контроллера **SurveysController** в проекте TailSpin.Web используют свойство **TempData** в MVC для кэширования определений опроса, когда пользователь разрабатывает новый опрос. Внутри свойство **TempData** хранит кэшируемые объекты в объекте сеанса ASP.NET.

Этот метод **New** вызывается, когда пользователь переходит на страницу **Новый опрос**.

C#

```
[HttpGet]
public ActionResult New()
{
    var cachedSurvey = (Survey)this.TempData[CachedSurvey];

    if (cachedSurvey == null)
    {
        cachedSurvey = new Survey();
// Первый раз на странице
    }

    var model =
this.CreateTenantPageViewData(cachedSurvey);
     model.Title = "New Survey";

    this.TempData[CachedSurvey] = cachedSurvey;

    return this.View(model);
}
```

Метод **NewQuestion** вызывается, когда пользователь щелкает ссылку **Добавить вопрос** на странице **Создание нового опроса**. Метод извлекает кэшированный опрос, созданный методом **New** в готовом виде для показа пользователю.

C#

```
 [HttpPost]
[ValidateAntiForgeryToken]
public ActionResult NewQuestion(Survey contentModel)
{
    var cachedSurvey = (Survey)this.TempData[CachedSurvey];

    if (cachedSurvey == null)
    {
        return this.RedirectToAction("New");
    }

    cachedSurvey.Title = contentModel.Title;
    this.TempData[CachedSurvey] = cachedSurvey;
```

```
    var model = this.CreateTenantPageViewData
(new Question());     model.Title = "New Question";

    return this.View(model);
}
```

Метод **AddQuestion** вызывается, когда пользователь нажимает кнопку **Добавить в опрос** на странице **Добавление нового вопроса**. Этот метод извлекает кэшированный опрос и добавляет новый вопрос, прежде чем отправить опрос обратно в сеанс.

C#

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult AddQuestion(Question contentModel)
{
    var cachedSurvey = (Survey)this.TempData[CachedSurvey];

    if (!this.ModelState.IsValid)
    {
        this.TempData[CachedSurvey] = cachedSurvey;
         var model = this.CreateTenantPageViewData(
           contentModel ?? new Question());
         model.Title = "New Question";
         return this.View("NewQuestion", model);
    }

    if (contentModel.PossibleAnswers != null)
    {
        contentModel.PossibleAnswers =
          contentModel.PossibleAnswers.Replace("\r\n",
"\n");
    }
    cachedSurvey.Questions.Add(contentModel);
    this.TempData[CachedSurvey] = cachedSurvey;
    return this.RedirectToAction("New");
}
```

Этот метод **New** вызывается, когда пользователь нажимает кнопку **Create** на странице **Create a new survey**. Этот метод извлекает готовый кэшированный опрос, сохраняет его для постоянного хранения и удаляет из сеанса.

C#

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult New(Survey contentModel)
{
    var cachedSurvey =
(Survey)this.TempData[CachedSurvey];

    if (cachedSurvey == null)
    {
        return this.RedirectToAction("New");
    }

    if (cachedSurvey.Questions == null ||
          cachedSurvey.Questions.Count <= 0)
    {

this.ModelState.AddModelError("ContentModel.Questions",
            string.Format(CultureInfo.InvariantCulture,
"Please add            at least one question to the
survey."));
    }

    contentModel.Questions = cachedSurvey.Questions;
     if (!this.ModelState.IsValid)
    {
        var model =
this.CreateTenantPageViewData(contentModel);
         model.Title = "New Survey";
        this.TempData[CachedSurvey] = cachedSurvey;
         return this.View(model);
    }

    contentModel.Tenant = this.TenantName;     try
    {
        this.surveyStore.SaveSurvey(contentModel);
    }
    catch (DataServiceRequestException ex)
    {
        …
    }

    this.TempData.Remove(CachedSurvey);
     return this.RedirectToAction("Index");
}
```

Комментарий Маркуса:

Tailspin использует свойство TempData вместо работы непосредственно с объектом сеанса ASP.NET, так как записи в словаре TempData существуют только в течение жизни одного запроса, после чего они автоматически удаляются из сеанса. Это упрощает управление содержимым сеанса.

Все изменения, внесенные компанией Tailspin в приложение Surveys для перехода от использования поставщика состояний сеансов ASP.NET в памяти по умолчанию к поставщику состояний сеансов, использующему службу кэширования AppFabric, были изменениями в конфигурации. Никакие изменения в коде не потребовались. Эти изменения конфигурации описаны в следующих разделах.

Создание кэша AppFabric

Компания Tailspin использовала портал управления Windows Azure AppFabric для настройки нового кэша с уникальным пространством имен. Эта конфигурация определяет конечную точку для доступа к кэшу и ключи безопасности, которые будут использоваться клиентским приложением (Tailspin Surveys) для доступа к службе кэширования Tailspin.

Дополнительные сведения см. в разделе «Создание кэша Windows Azure AppFabric» по адресу http://msdn.microsoft.com/ru-ru/library/gg618004.aspx.

Добавление ссылок в проект TailSpin.Web

Компания Tailspin добавила в проект Tailspin.Web ссылки на следующие три сборки:

- Microsoft.ApplicationServer.Caching.Client
- Microsoft.ApplicationServer.CachingCore
- Microsoft.Web.DistributedCache

Эти три сборки включены в пакет Windows Azure AppFabric SDK, и их необходимо развернуть в составе своего решения. Дополнительные сведения см. в разделе «Подготовка проекта Visual Studio для использования кэширования Windows Azure AppFabric» по адресу http://msdn.microsoft.com/ru-ru/library/gg278344.aspx.

Настройка приложения TailSpin.Web

Последние изменения компания Tailspin внесла в файл Web.config в проекте TailSpin.Web. Эти изменения показаны в следующем примере.

XML

```
<configSections>
  …
  '<section name="dataCacheClients"
       type="Microsoft.ApplicationServer
       .Caching.DataCacheClientsSection,
      Microsoft.ApplicationServer.Caching.Core"
    allowLocation="true" allowDefinition="Everywhere"/>
</configSections>
…
<dataCacheClients>
  <dataCacheClient name="SslEndpoint">     <hosts>
      <host name="TallowLocation="true"
allowDefinition="Everywhere"/>net"
           cachePort="22243" />
    </hosts>
    <securityProperties mode="Message" sslEnabled="true">
      <messageSecurity
        authorizationInfo="YWNz…
      </messageSecurity>
    </securityProperties>
  </dataCacheClient>
</dataCacheClients>
…
<system.web>
  <sessionState mode="Custom"
    customProvider="AppFabricCacheSessionStoreProvider">
     <providers>
      <add name="AppFabricCacheSessionStoreProvider"
         type="Microsoft.Web.DistributedCache
        .DistributedCacheSessionStateStoreProvider,
        Microsoft.Web.DistributedCache"
       cacheName="default"
      applicationName="TailSpin.Web"
       useBlobMode="true"
      dataCacheClientName="SslEndpoint" />
    </providers>
  </sessionState>
…
</system.web>
```

Комментарий Маркуса:

Эти данные конфигурации вы можете скопировать и вставить с портала. Настроив кэш, найдите кнопку Просмотр конфигурации клиента на странице кэша AppFabric.

В разделе кода **dataCacheClient** находится пространство имен кэша, заданное компанией Tailspin на портале управления Windows Azure AppFabric. Здесь также находится ключ авторизации, который компания Tailspin скопировала с портала.

В разделе кода **sessionState** приложение настраивается для использования поставщика состояний сеансов службы кэширования AppFabric.

Примечание.

Если приложение Tailspin Surveys выполняется локально в эмуляторе Windows Azure, необходимо помнить о двух вещах. Во-первых, если приложение запускается локально и используется кэш AppFabric в облаке, приложение будет работать медленно. Причина в задержке, которая возникает при обращении приложения к кэшу в облаке. Эта задержка исчезнет после развертывания приложения в облаке. Во-вторых, если не добавить необязательный атрибут applicationName, сеанс в локальном эмуляторе будет работать некорректно. Это проблема только эмулятора вычислений, а не облака.

Вывод вопросов

Приложение хранит определение опроса и все вопросы в хранилище таблиц. Для вывода вопросов на странице в браузере приложение использует класс **EditorExtensions** в MVC.

Комментарий Маркуса:

Компания Tailspin выбрала этот механизм вывода вопросов, поскольку он упрощает ввод новых типов вопросов в будущем.

Когда метод действия **Display** в классе **SurveysController** в проекте TailSpin.Web.Survey.Public создает представление для вывода формы опроса, он извлекает определение опроса из хранилища, заполняет модель данными и передает модель в представление. Этот метод действия показан следующем примере кода.

C#

```
[HttpGet]
public ActionResult Display(string tenant, string
surveySlug)
{
    var surveyAnswer = CallGetSurveyAndCreateSurveyAnswer(
         this.surveyStore, tenant, surveySlug);
 var model =
new TenantPageViewData<SurveyAnswer>(surveyAnswer);
model.Title = surveyAnswer.Title;
return this.View(model);
}
```

Для вывода вопросов в представлении используется класс **EditorExtensions**. В следующем примере кода показано, как на странице Display.aspx используется элемент **Html.EditorFor**, определенный в классе **System.Web.Mvc.EditorExtensions**.

CSHTML

```
<% for (int i = 0;
   i < this.Model.ContentModel.QuestionAnswers.Count; i++ )
{ %>
   …
   <%: Html.EditorFor(m=>m.ContentModel.QuestionAnswers[i],
      QuestionTemplateFactory.Create(

Model.ContentModel.QuestionAnswers[i].QuestionType)) %>
   …
<% } %>
```

Этот элемент последовательно применяется ко всем вопросам, извлеченным контроллером из хранилища, и с помощью служебного класса **QuestionTemplateFactory** определяет, какие пользовательские элементы управления (файлы .ascx) следует использовать для обработки вопроса. Пользовательские элементы управления FiveStar.ascx, MultipleChoice.ascx и SimpleText.ascx находятся в папке EditorTemplates в проекте.

Вывод сводных статистических данных

Асинхронная задача, которая создает сводную статистику по опросам (см. главу 5, «Создание масштабируемого приложения, обслуживающего несколько развертываний, для Windows Azure»), сохраняет эти сводные данные в хранилище больших двоичных объектов, используя отдельный большой двоичный объект для каждого опроса. Приложение Surveys отображает эти сводные статистические данные точно так же, как и вопросы. В следующем примере кода показан метод действия **Analyze** класса **SurveysController** в проекте TailSpin.Web. Этот метод считывает результаты из хранилища больших двоичных объектов и заполняет модель данными.

C#

```
public ActionResult Analyze(string tenant, string
surveySlug)
{
    var surveyAnswersSummary =
          this.surveyAnswersSummaryStore
        .GetSurveyAnswersSummary(tenant, surveySlug);

    var model =

this.CreateTenantPageViewData(surveyAnswersSummary);
     model.Title = surveySlug;
    return this.View(model);
}
```

И здесь представление использует элемент **Html.EditorFor** для отображения вопросов. В следующем примере кода показана часть файла Analyze.aspx.

CSHTML

```
.QuestionTemplateFactory.Create
Model.ContentModel.QuestionAnswersSummaries[i
.QuestionType))%>
</li>
<% } %>
<% for (int i = 0;     i <
this.Model.ContentModel.QuestionAnswersSummaries.Count;
i++ ) { %>
<li>
    <%: Html.DisplayFor(m => m.ContentModel
        .QuestionAnswersSummaries[i],
        "Summary-" + TailSpin.Web.Survey.Public.Utility
```

Шаблоны пользовательских элементов управления для отображения сводных статистических данных называются Summary-FiveStar.ascx (когда отображается среднее для вопросов с числовым диапазоном), Summary-MultipleChoice.ascx (когда отображается гистограмма) и Summary-SimpleText.ascx (когда отображается облако слов). Эти шаблоны можно найти в папке DisplayTemplates в проекте TailSpin.Web. Для дополнительных типов вопросов необходимо добавить в эту папку дополнительные шаблоны пользовательских элементов управления.

Использование SQL Azure

Приложение Surveys использует хранилище Windows Azure для хранения определений опросов и ответов на вопросы. Компания Tailspin решила использовать хранилище Windows Azure из-за более низких затрат и потому что эти затраты зависят от объема фактического использования (в плане вычислительной мощности и количества транзакций в хранилище) за месяц. Но для контроля расходов, связанных с хранилищем, приложение Surveys не предлагает подписчикам особенно гибких средств анализа ответов на вопросы опроса. Подписчик может просматривать ответы на вопросы в том порядке, в котором они были представлены подписчиками, а также просмотреть набор готовых сводных статистических данных по каждому опросу.

SQL Azure позволяет подписчикам выполнять сложный анализ результатов опроса.

Для расширения возможностей анализа в приложении Surveys компания Tailspin позволяет подписчикам сбрасывать ответы на вопросы в базу данных SQL Azure. После этого подписчики могут выполнять любой подробный статистический анализ или загружать результаты опроса в локальные приложения путем экспорта данных из SQL Azure.

Эта функция входит в ежемесячную плату за подписку Premium. Подписчики на других уровнях могут приобрести эту функцию в дополнение к своей подписке.

Цели и требования

Приложение должно иметь возможность экспортировать в Azure SQL все данные опроса, включая определения вопросов, а не только ответы.

Подписчики, которые решили использовать эту функцию, получают свой собственный, закрытый экземпляр SQL Azure, что позволяет анализировать и обрабатывать данные любым способом, который они сочтут нужным. Например, они могут создавать новые таблицы сводных данных или макеты сложных запросов для анализа данных. Кроме того, закрытый экземпляр SQL Azure помогает сохранить конфиденциальность данных.

Комментарий По:

Предоставление отдельного экземпляра SQL Azure каждому подписчику позволяет подписчикам настраивать данные, что упрощает модель безопасности.

Решение

Во время разработки решения и его внедрения приложение предоставит новый экземпляр SQL Azure подписчикам, имеющим доступ к этой функции. Процесс провиженинга создаст необходимые таблицы в базе данных. В процессе разработки решения и его внедрения приложение Surveys сохраняет данные, необходимые приложению и подписчику для доступа к экземпляру SQL Azure, в хранилище больших двоичных объектов в составе сведений о подписчике.

Задача в рабочей роли отслеживает очередь на предмет сообщений, которые предписывают сбросить результаты опроса, проводимого подписчиком, в таблицы SQL Azure. На Рис. 7 показана структура таблиц в SQL Azure.

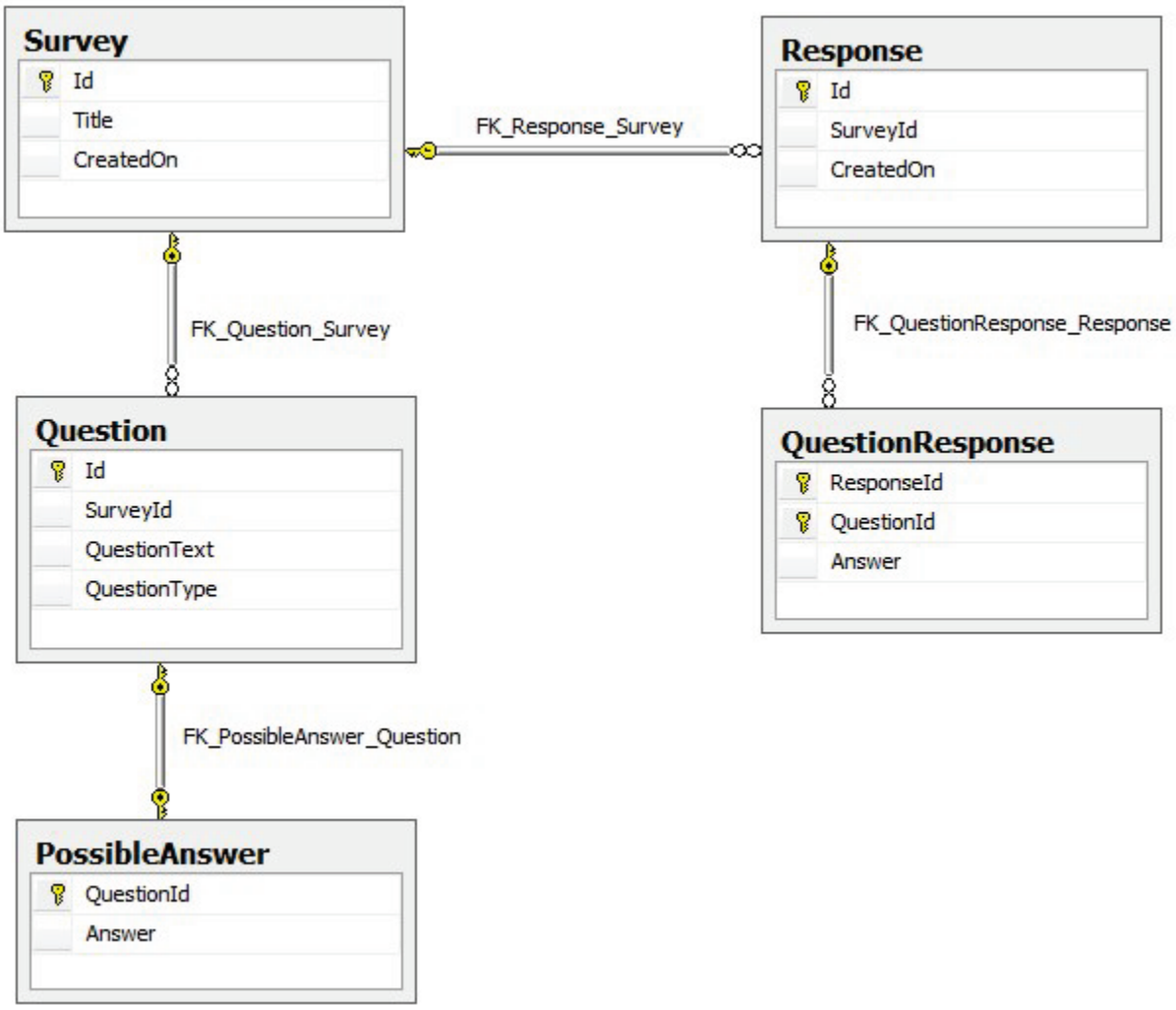


Рис. 7 **Структура таблиц опросов в SQL Azure**

Реализация

Теперь самое время подробно рассмотреть код, сбрасывающий ответы на вопросы в SQL Azure. В ходе изучения этого раздела может потребоваться загрузить решение Visual Studio для приложения Tailspin Surveys со страницы http://wag.codeplex.com/[2].
В следующем примере кода показана задача в рабочей роли, выполняемая сообщением из очереди. Метод **Run** находится в классе **TransferSurveysToSqlAzureCommand**.

C#

```
public void Run(SurveyTransferMessage message)
{
    Tenant tenant =
        this.tenantStore.GetTenant(message.Tenant);
```

```
     this.surveySqlStore.Reset(
        tenant.SqlAzureConnectionString, message.Tenant,
          message.SlugName);

    Survey surveyWithQuestions = this.surveyRespository
         .GetSurveyByTenantAndSlugName(message.Tenant,
          message.SlugName, true);

    IEnumerable<string> answerIds = this.surveyAnswerStore
        .GetSurveyAnswerIds(message.Tenant,
          surveyWithQuestions.SlugName);

    SurveyData surveyData =
surveyWithQuestions.ToDataModel();

    foreach (var answerId in answerIds)
    {
        SurveyAnswer surveyAnswer = this.surveyAnswerStore
             .GetSurveyAnswer(surveyWithQuestions.Tenant,
              surveyWithQuestions.SlugName, answerId);

        var responseData = new ResponseData
        {
            Id = Guid.NewGuid().ToString(),
            CreatedOn = surveyAnswer.CreatedOn
        };

        foreach (var answer in
surveyAnswer.QuestionAnswers)
        {
            var questionResponseData = new
QuestionResponseData
                {
                    QuestionId = (from question in
                      surveyData.QuestionDatas
                     where question.QuestionText ==
                      answer.QuestionText
                     select question.Id).FirstOrDefault(),
                    Answer = answer.Answer
                };

            responseData.QuestionResponseDatas
                .Add(questionResponseData);
        }
        if (responseData.QuestionResponseDatas.Count > 0)
        {
            surveyData.ResponseDatas.Add(responseData);
        }
```

```
    }

    this.surveySqlStore
        .SaveSurvey(tenant.SqlAzureConnectionString,
surveyData);
}
```

Параметр *message* этого метода указывает, какой опрос экспортировать. Этот метод сначала удаляет данные опроса в SQL Azure, а затем последовательно просматривает все ответы на вопросы и сохраняет самые последние данные в SQL Azure. Приложение не пытается выполнять эти операции параллельно; для подписчиков с большими объемами данных операция сброса данных в SQL Azure может занять некоторое время.

Комментарий Маркуса:

Эта задача является частью рабочей роли, описанной в главе 5, «Создание масштабируемого приложения, обслуживающего несколько развертываний, для Windows Azure». Она запускается не по расписанию, а сообщением из очереди.

Приложение использует LINQ to SQL для управления взаимодействием с SQL Azure. Следующий код из класса **SurveySqlStore** показывает, как приложение использует классы **SurveyData** и **SurveySqlDataContext**. Эти классы создаются конструктором SurveySql.dbml.

C#

```
public void SaveSurvey(string connectionString,
    SurveyData surveyData)
{
    using (var dataContext =
        new SurveySqlDataContext(connectionString))
    {
        dataContext.SurveyDatas.InsertOnSubmit(surveyData);
         try
        {
            dataContext.SubmitChanges();
        }
        catch (SqlException ex)
        {
            Trace.TraceError(ex.TraceInformation());
             throw;
        }
    }
}
```

Для шифрования строки подключения SQL в файле Web.config можно использовать поставщик конфигурации с защитой по стандарту Pkcs12, загрузить который можно по адресу http://archive.msdn.microsoft.com/pkcs12protectedconfg. Дополнительные сведения об использовании этого поставщика можно найти в записях блогов разработчиков SQL Azure, например: http://blogs.msdn.com/b/sqlazure/archive/2010/09/07/10058942.aspx.

Ссылки и материалы

Дополнительные сведения о службах хранения данных Windows Azure см. в разделе «Использование служб хранения данных Windows Azure» в сети MSDN и блоге группы разработки хранилищ Windows Azure: http://msdn.microsoft.com/ru-ru/library/ee924681.aspx http://blogs.msdn.com/b/windowsazurestorage/.

Дополнительные сведения о SQL Azure см. в *Руководстве по работе с SQL Azure*, загрузить которое можно по адресу: http://wag.codeplex.com.

Более подробные сведения о маркерах продолжения и хранилище таблиц Windows Azure см. в разделе «Реализация постраничного просмотра с помощью хранилища таблиц Windows Azure» в главе 8 книги «Миграция приложений в облако» или доступной по адресу http://wag.codeplex.com/.