Министерство образования и науки республики Казахстан

Костанайский государственный университет имени А.Байтурсынова

Кафедра программного обеспечения

Т.Г.Бортко, В.Б. Синицина

**Облачное программирование**

Учебно- методическое пособие

(часть 1)

Костанай, 2013

**ББК85.63**

**Б 65**

**Рецензенты**

Уаисова Майра Маликовна–зам. декана инженерного факультета КИПУ имени М. Дулатова

Исмаилов Арман Оразалинович - кандидат физико-математических наук, доцент кафедры программное обеспечение

Иванова Ирина Владимировна - кандидат педагогических наук, ст. преподаватель кафедра программного обеспечения КГУ

**Составители:**

Бортко Татьяна Георгиевна, доктор педагогических наук, профессор кафедры программного обеспечения

Синицина Виктория Брисовна, ст.преподаватель кафедры программного обеспечения

Б65 Бортко Т.Г.

Облачное программирование. Учебно-методическое пособие (часть 1) – Костанай, 2013.72 с.

В учебно-методическом пособии описываются различные облачные решения, их возможности, преимущества и риски, рассматриваются Azure Services Platform: Windows Azure, Windows .Net Services, широко освещен вопрос Windows Azure Queue: технология, программирование, модели данных. Вторая часть пособия раскрывает практические вопросы облачного программирования.

Предназначено для магистрантов технических специальностей; может быть рекомендовано преподавателям высших учебных заведений, изучающих вопросы облачных технологий.

ББК 85.63

Утверждено и рекомендовано к изданию Учебно-методическим советом Костанайского государственного университета имени А.Байтурсынова,

­­­\_\_ \_\_\_\_\_\_2013г., протокол №\_\_

© Бортко Т.Г., 2013

Содержание

|  |  |
| --- | --- |
| Тема 1. Введение в Cloud Computing…………………………………………. | 4 |
| Тема 2. Облачные решения: возможности, преимущества, риски…………. | 10 |
| Тема 3. Стратегия развертывания облака……………………………………. | 16 |
| Тема 4. Azure Services Platform. Windows Azure, Windows .Net Services….. | 19 |
| Тема 5. Azure Services Platform. Microsoft SQL Services, Live Services……. | 23 |
| Тема 6. Платформа Windows Azure…………………………………………... | 26 |
| Тема 7. Сервисы хранения данных в Windows Azure. VM – роль…………. | 29 |
| Тема 8. SQL Azurе……………………………………………………………... | 33 |
| Тема 9. Windows Azure AppFabric……………………………………………. | 36 |
| Тема 10. Инструметарий разработчика Windows Azure…………………….. | 40 |
| Тема 11. Структурированное хранилище данных Windows Azure Table….. | 43 |
| Тема 12. Windows Azure Table – программирование………………………... | 46 |
| Тема 13. Windows Azure Blob: введение, модель данных, REST – интерфейс………………………………………………………………………. | 49 |
| Тема 14. Windows Azure Blob, как список блоков. Примеры REST – запросов………………………………………………………………………… | 53 |
| Тема 15. Windows Azure Queue. Введение. Модель данных. REST – интерфейс……………………………………………………………………… | 58 |
| Тема 16. Windows Azure Queue: примеры использования, REST – запросы…………………………………………………………………………. | 61 |
| Тема 17. Microsoft .Net Access Services. Идентификация на основе утверждений……………………………………………………………………. | 65 |
| Тема 18. Отладка и развертывание проекта…………………………………. | 68 |

**Тема 1. Введение в Cloud Computing**

Прежде чем начать рассматривать облачные вычисления, необходимо охарактеризовать методику распределенной обработки данных.

Распределенная обработка данных - методика выполнения прикладных программ группой систем. Таким образом, фактически пользователь получает возможность работать с прикладными процессами, расположенными в нескольких взаимосвязанных системах.

Распределенные вычисления являются способом решения сложных задач, требующих больших вычислительных мощностей, основанном на объединении нескольких компьютеров в параллельную вычислительную систему.

В качестве исторического примера распределенных вычислений можно привести решение французского математика и вычислителя при французском правительстве барона Гаспара де Прони.

Под его руководством была начата работа по уточнению логарифмических и тригонометрических таблиц, в связи с переходом на метрическую систему. Для того времени требовалось осуществление огромного объема вычислений. Де Прони перенес идею о разделении труда на вычислительный процесс, исполнители были распределены по трем уровням:

1. низший уровень - люди-вычислители (computers), которые должны были только аккуратно складывать и вычитать числа;
2. средний уровень - "технологи", которые занимались организацией конкретного рутинного вычислительного процесса;
3. высший уровень - математики, организовывающие подготовку математического обеспечения и обобщения полученных результатов.

**Характеристика распределенной обработки данных**

Суть распределенной обработки данных заключается в том, что пользователь и прикладное программное обеспечение могут работать с удаленными средствами посредством каналов связи.

Компьютеры, являющиеся частями распределенной среды обработки данных делятся на серверы и клиенты. Первые предоставляют вычислительные ресурсы вторым.

Управление распределенной средой осуществляется сетевой операционной системой. Кроме того, данная среда подразумевает наличие распределенной базы данных и инструментарий для управления этой базой.

Преимущества распределенной системы обработки данных:

* возможность обслуживания большого числа пользователей;
* снятие пиковых нагрузок с централизованной БД, путем распределения обработки и хранения локальных БД;
* обеспечение доступа исполнителей к вычислительным ресурсам всей сети;
* обеспечение симметричного обмена данными между пользователями сети.

В зависимости от способа распределения данных, выделяют следующие способы организации обработки данных:

* централизованный;
* децентрализованный;
* смешанный.

Централизованная организация данных. С точки зрения реализации, является наиболее простым вариантом организации. Имеется единственная централизованная база данных, доступ к которой организовывается и регулируется сервером, на котором она находится. Доступ осуществляется при помощи удаленного запроса.

В настоящее время все большую популярность набирает идея добровольного использования для организации вычислений компьютеров обычных пользователей.

Преимуществом данного способа являются: простота реализации и поддержки; недостатком - ограничение на параллельную обработку данных, поскольку все запросы направляются к единственному серверу.

Децентрализованная организация данных. Данный подход предполагает разбиение логически единой базы данных на несколько физически распределенных. Каждый клиент имеет доступ к собственной локальной базе данных, которая является либо частью общей базы данных (распределение), либо копией базы в целом (дублирование), второй вариант приводит к дублированию базы у каждого клиента.

Существование копий отдельных частей базы не допускается.

Преимущества данного метода:

* большая часть запросов удовлетворяется локальными базами, что снижает временные затраты;
* увеличивается доступность данных;
* повышенная надежность хранения данных;
* частичная работоспособность системы при выходе из строя одного сервера.

Недостатки:

* могут возникнуть запросы или транзакции, требующие доступа ко всем серверам, что увеличивает время ожидания;
* необходимо наличие информации о хранении данных в БД.

Смешанный подход. Подразумевает объединение двух способов децентрализации: разбиения и дублирования. Возникает необходимость наличия информации о том, где в сети размещены данные. Достигается компромисс между требуемым объемом памяти под базу в целом и объемом памяти, необходимом каждому серверу. При этом легко организуется параллельная обработка данных. Тем не менее остаются такие проблемы как: надежность и выполнение требований к памяти. Данный подход можно реализовывать только при наличии сетевой системы управления базой данных, обеспечивающей хранение, целостность и безопасность хранимых данных.

**Cloud computing**

Небольшое отступление: термин "облачные" возник из способа представления Интернета, как облака на различных диаграммах, иллюстрациях и схемах.

Найти человека, чья работа или интерес связан со сферой информационных технологий, ничего не слышавшего про "облачные вычисления" будет крайне затруднительно.

Эта тема является одной из самых обсуждаемых в последнее время.

Такой ажиотаж объясняется просто - появилось решение, способ, позволяющий существенно сократить расходы на ИТ - услуги, отказаться от инвестирования в инфраструктуру и расходов, связанных с ее сопровождением и модернизацией, свести время развертывания приложения к минимуму.

Существует три основные модели расположения приложений:

* на стороне заказчика;
* хостинг;
* в "облаке".

Наиболее привычной и традиционной моделью расположения приложения является, безусловно, на **стороне заказчика**, локально, в рамках его инфраструктуры.

Основной сложностью данной модели являются существенные затраты на формирование ИТ - инфраструктуры, отвечающей требованиям бизнеса и используемых приложений, а также затраты, связанные с ее сопровождением и модернизацией.

В качестве положительно момента можно отметить, что инфраструктура, аппаратное и программное обеспечение полностью подконтрольны заказчику.

**Хостинг**. Данная модель развертывания приложений получила развитие в связи с распространением глобальной сети и увеличением ее роли в профессиональной деятельности человека. Ранее такая модель называлась Application Service Provider (ASP), теперь - Software as a service (SaaS).

**Программное обеспечение, как сервис (Software as a service, SaaS)**

Представляет собой бизнес - модель продажи и использования программных средств, при котором поставщик разрабатывает и управляет приложением самостоятельно, предоставляя пользователям доступ к нему посредством глобальной сети.

Очевидно, что основным преимуществом такого подхода является отсутствие затрат для потребителя, связанных с установкой программного продукта, сопровождением его и аппаратной платформы.

Отметим положительные особенности SaaS модели с точки зрения поставщиков услуг и конечных пользователей.

С точки зрения SaaS пользователей

* ПО не нужно устанавливать на рабочих местах пользователей;
* отсутствие временных затрат на развертывание системы в рамках организации;
* упрощение процессов обновления и сопровождения конечного продукта;
* работоспособность ПО не зависит от платформы, используемой пользователями.

С точки зрения SaaS разработчиков:

* снижение затрат на обслуживание отдельных клиентов;
* упрощение процессов проникновения на новые рынки;
* защита от нелегального распространения;
* заказчик "привязывается" к разработчику, нельзя отказаться от услуг разработчика и продолжить использовать программный продукт.

Особенности SaaS:

* программный продукт разработан для удаленного использования;
* единовременно приложением может пользоваться множество клиентов;
* оплата представляет собой либо абонентскую плату, либо зависит от степени использования приложения (к примеру, объем и сложность осуществленных операций);
* сопровождение осуществляется поставщиком услуг и входит в установленную оплату;
* обновление продукта осуществляется постепенно и является "прозрачным" для конечных пользователей.

Таким образам, заказчик платит не за обладание программным продуктом, а за его аренду у поставщика услуг. Кроме всего вышеперечисленного, отсутствие у конечно пользователя продукта, как такового, может рассматриваться как способ борьбы с нелегальным распространением ПО.

Данная модель отличается меньшим контролем за инфраструктурой, аппаратным и программным обеспечением.

**Расположение приложений в "облаке"**. Самая "молодая" модель размещения приложений. Отличается практическим отсутствием контроля за инфраструктурой и арендуемым обеспечением со стороны заказчика. Расходы же, соответственно, существенно снижаются. Осуществляется переход от инвестиций в инфраструктуру к операционным расходам.

Таким образом, облачные вычисления - это подход к размещению, предоставлению и потреблению приложений и компьютерных ресурсов, при котором приложения и ресурсы становятся доступны через Интернет в виде сервисов (услуг), потребляемых на различных платформах и устройствах .

Оплачиваются такие услуги, как правило, по факту.

**Облачные технологии и виртуализация**

Одним из положений, на котором основывается "облачный" подход, является положение о неравномерности запросов к ресурсам со стороны клиента. Для сглаживания этой неравномерности между аппаратными ресурсами и middleware помещается слой виртуальных серверов.

Middleware или подпрограммное обеспечение - слой ПО, состоящий из "посредников" между различными компонентами крупного приложения.

Под виртуализацией в ИТ, как правило, понимается абстракция вычислительных ресурсов и предоставление пользователю системы, скрывающей в себе собственную реализацию.

Таким образом, балансировка нагрузки осуществляется как средствами программного обеспечения, так и средствами распределения виртуальных серверов по реальным.

**Концепция "облачной" обработки данных - всё как услуга (Everything as a service)**

**Инфраструктура, как сервис (Infrastructure as a service, IaaS)**

Как правило, под этим понимают предоставление компьютерной инфраструктуры в форме виртуализации, как сервиса согласно концепции облачных вычислений. Иными словами - это возможность арендовать инфраструктурные ресурсы (сетевое оборудование, устройства хранения, серверы).

Таким образом, IaaS можно представить, как сочетание:

* Аппаратных средств;
* Операционных систем и системного программного обеспечения;
* Связующего программного обеспечения.

Непосредственное управление всей инфраструктурой осуществляется поставщиком услуг. Клиент, со своей стороны управляет операционной системой и установленным ПО. Обычно в зависимости от нужд заказчика, меняется и объем используемой инфраструктуры.

Подход IaaS избавляет компании от необходимости поддержки "сложных" инфраструктурных решений (центры обработки данных, сетевая инфраструктура), кроме того, уменьшаются и соответствующие издержки.

Облачная платформа предоставляет сервисы для запуска виртуальных машин и сервисы хранения данных.

Таким образом, к особенностям IaaS можно отнести:

* виртуализацию (по факту, покупку аппаратных мощностей заменит покупка серверного времени, дискового пространства и сетевой пропускной способности);
* интегрированные системы управления - единая платформа для управления различными типами оборудования;
* возможность использования готовых инфраструктур, на основе проверенных архитектур.

**Платформа, как сервис (Platform as a service, PaaS)**

Платформа, как сервис - это возможность аренды платформы, которая обычно состоит из операционной системы и прикладных сервисов. Такая платформа представляет собой комплексное решение для разработки, тестирования, развертывания и поддержки приложений. PaaS также включает в себя и IaaS.

К особенностям PaaS можно отнести:

* модель оплаты "pay as you go" - услуга оплачивается только тогда, когда необходима;
* масштабируемость - количество выделяемых ресурсов зависит от количества обслуживаемых пользователей;
* отказоустойчивость и безопасность - не требуют дополнительных затрат от компании, поскольку уже реализованы в PaaS ;
* виртуализация;
* интегрированная платформа разработки, тестирования, развертывания и сопровождения приложений;
* доступность - для использования PaaS необходимо только подключение к Internet.

Как уже упоминалось, возможность управления и контроля отдельных элементов инфраструктуры, аппаратного и программного обеспечения зависят от выбранного способа размещения приложений.

Для большей наглядности приведем таблицу границ управляемости, где "+" - управляется клиентом, "-" - управляется поставщиком услуги.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1.1 - Границы управляемости | | | |
|  | **Собственная инфраструктура** | **IaaS** | **PaaS** |
| Приложения | + | + | + |
| Среда выполнения | + | + | - |
| Безопасность и интергация | + | + | - |
| БД | + | + | - |
| Серверы | + | - | - |
| Виртуализация | + | - | - |
| Аппаратная платформа | + | - | - |
| Хранилище | + | - | - |
| Сетевые ресурсы | + | - | - |

**Список дополнительных материалов для самостоятельного изучения**

1. Распределенная обработка данных
   * <http://global-july.com/lecture_t4.html>
   * <http://abc.vvsu.ru/Books/up_inform_tehnol_v_ekon/page0019.asp>
   * <http://www.market-journal.com/itvupravlenii/45.html>
   * <http://denizzone.com/baset5.html>
2. Виртуализация
   * <http://www.datasystems.ru/dir.php?id=1370>
   * <http://www.trinitygroup.ru/solution/infrastucture/virtualization/storage/>
3. "Облачные вычисления"
   * <http://www.parallels.com/ru/spp/understandingclouds/>
   * <http://www.3dnews.ru/editorial/cloud_computing/>
   * <http://netler.ru/pc/cloud.htm>
   * <http://www.pcweek.ru/themes/detail.php?ID=112879>

**Тема 2. Облачные решения: возможности, преимущества, риски**

**Особенности проектирования облачных решений**

"Облачное" решение, как уже отмечалось в предыдущей лекции обладает рядом существенных отличий, от других способов размещения приложений. Одним из наиболее существенных отличий, в частности, является возникающая независимость жизненного цикла приложения от жизненного цикла инфраструктуры.

Таким образом, при проектировании "облачных" решений необходимо учитывать следующее (на примере платформы Windows Azure ):

* Любое приложение, являющееся частью "облака" - удаленный сервис. На этапе планирования решения необходимо учитывать временные задержки, между отправкой запроса и получением результата, а также необходимость контроля статуса соединения и его возобновления.
* Динамичная инфраструктура облака обеспечивает горизонтальную масштабируемость, таким образом, количество единовременно работающих экземпляров приложения может меняться. В связи с этим состояние приложения необходимо хранить в долговременном хранилище (не на локальных дисках).
* Ряд сценариев, стоимость которых велика, в случае их реализации на основе локальной инфраструктуры (к примеру обработка больших объемов данных), могут быть реализованы в "облачном" решении в виде платформенных сервисов.
* Управление выделением вычислительных ресурсов для приложения, их сопровождение осуществляется платформой.
* Стоимость облачного решения. Данное понятие стало актуальным при появлении "облачных решений". Рассмотрим его более детально.

**Стоимость "облачного" решения**

При использовании "облака" заказчиком оплачивается аренда вычислительных ресурсов, что и является основной статьей расходов, при использовании такого решения.

В первую очередь, на стоимость решения будет влиять трафик - объем переданных, или полученных данных. Для уменьшения стоимость данной составляющей целесообразно использовать сжатие данных, переход к SOA - архитектуре и размещение кода, активно работающего с данными рядом с их хранилищем (т.е. вычисления не будут выходить за рамки "облака" и влиять на объем передаваемой информации).

Стоимость второй компоненты - вычислительных ресурсов - начинает измеряться с момента начала развертывания приложения и зависит от мощности предоставленных виртуальных машин.

Третьей компонентой является хранилище данных. Стоимость хранения и обработки данных определяется их объемом и количеством операций с ними. Таким образом, для уменьшения стоимость данной компоненты целесообразно подбирать хранилище оптимального размера и минимизировать количество операций с данными.

**Мультитенантная архитектура**

Мы рассмотрели, из чего складывается стоимость "облачного" решения. Так каким же образом стоит выстраивать архитектуру приложения, чтобы минимизировать затраты не снижая качества предоставляемых услуг?

Таким способом снижения стоимости вычислительных ресурсов и хранилища данных является так называемая **мультитенантная архитектура приложения**.

Отметим, что данная архитектура применяется, как правило, для приложений, обслуживающих группы изолированных друг от друга пользователей.

Отличие данной архитектуры от обычной заключается в том, что, в отличие от общего случая, когда каждая группа пользователей использует свой экземпляр приложения и свою изолированную БД, состоит в том, что мультитенантное приложение обладает встроенными возможностями для обслуживания нескольких групп пользователей.

Таким образом, для каждой группы пользователей, в случае размещения приложения в облаке, выделяется только необходимое, не фиксированное количество ресурсов, т.е. "стоимость" начисляется по - факту использования "облачных" ресурсов.

Рассмотрим **этапы перехода к мультитенантной архитектуре**:

1. **Выделенная архитектура**. Каждая группа пользователей использует свою копию приложения (сервиса). Каждая копия ориентирована только на свою группу пользователей, таким образом, каждая копия содержит персональное расширение, в котором хранятся пользовательские настройки. Вся неэффективность данной модели станет наглядна при увеличении числа пользователей. Также становится невозможным повышения эффективности за счет роста масштабов, поскольку фактически пользователи пользуются разными экземплярами приложения.
2. **Настраиваемая архитектура**. В данном случае приложение, либо сервис настраивается для каждого конкретного пользователя через конфигурацию. Т.е. каждый пользователь работает со своей копией приложения, не смотря на то, что копии идентичны. Вычислительные мощности не разделяются между различными экземплярами приложения, что также не позволяет добиться повышения эффективности за счет увеличения масштабов.. Разделения же самого приложения может быть как виртуальным, так и физическим.
3. **Мультетенантная** архитектура. Настройка для каждого пользователя выполняется исключительно через конфигурацию, при использовании инструментов для самостоятельной настройки. При этом отсутствует возможность горизонтального масштабирования, т.е. повышение производительности может быть достигнуто только через вертикальное масштабирование.
4. **Масштабируемая** архитектура. Данная модель поддерживает мультитенатный подход, а также возможность горизонтального масштабирования. При этом новые экземпляры приложений добавляются в общий пул. Общая нагрузка распределяется по всей инфраструктуре. Архитектура настраивается через конфигурацию.

**Модели организации мультитенантного хранилища данных**

1. Отдельные базы данных. Каждая группа пользователей имеет собственную базу данных, с собственной схемой данных. Может быть эффективной при небольшом числе пользователей в расчете на базу данных. Кроме того, данный вариант предпочтителен, в случае предъявляемых пользователями строгих требований к изоляции и безопасности данных.
2. Совместно используемые базы данных с разными схемами. Все пользователи работают с одной БД, но имеют различные наборы предопределенных полей. Наиболее полезен данный вариант в случае, если хранение данных разных пользователей в одной таблице допустимо, а также заранее известно какие поля потребуются. Может привести к ситуации разряженного наполнения таблиц.
3. Совместно используемые базы данных с совместной схемой. Для хранения расширений данных используются специальные подходы и техники, пользователи работают с одной и той же базой данных. При этом, можно предложить пользователям практически неограниченное количество полей, однако, возникают проблемы при организации индексации и поиска информации. Данный вариант применим, если хранение данных различных групп пользователей в одних и тех же таблицах удовлетворяет требованиям безопасности и изоляции данных.

Прежде, чем перейти к рассмотрению преимуществ и рисков для бизнеса, вкратце рассмотрим подход **SOA (Service-oriented architecture)**.

SOA, или сервисно - ориентированная архитектура - подход, суть которого заключается в разработке программного обеспечения, основанной на использовании сервисов со стандартизированными интерфейсами. Иными словами, приложение "собирается" по частям из нескольких модулей - комплектующих, а не пишется "от и до".

Главными целями, к достижению которых стремится SOA, являются:

1. повышение масштабируемости создаваемых систем;
2. упрощение процессов контроля и управления созданными системами;
3. сокращение издержек, связанных с разработкой приложений;
4. увеличение доли повторного использования кода;
5. независимость систем от платформ, инструментов разработки и языков программирования.

В основе данного подхода лежат:

* многократное использование функциональных элементов;
* устранение дублирования функционала;
* стандартизация и унификация типовых процессов;
* переход компаний на функциональную организацию.

Одним из основных преимуществ SOA является надежность, особенно при реализации, основанной на применении веб - технологий.

При планировании SOA - архитектуры важно помнить о необходимость наличия централизованного репозитория - справочника всех доступных сервисов. Данный справочник должен обеспечивать группировку сервисов по бизнес - функциям, подразделениям, источникам данных и т.п. Классификация сервисов должна быть построена таким образом, чтобы обеспечить быстрый поиск разработчиками необходимых им сервисов для создания новых приложений. Компоненты, из которых собирается приложение представляют собой "черные ящики".

Принципы SOA:

* архитектура не должна быть привязана к какой - либо технологии;
* система не должна зависеть от используемой платформы;
* система не должна зависеть от применяемых языков программирования;
* используемые сервисы должны быть независимы от конкретных приложений и обладать стандартизованными интерфейсами доступа к ним;
* сервисы должны быть "слабосвязанными" между собой.

SOA - не является продуктом, или спецификацией. Если вы хотите реализовать данный подход, то потребуется тщательное и детальное планирование выстраиваемой архитектуры.

**Преимущества SOA, SaaS и облачных решений**

Рассмотрим более детально, какие именно преимущества способен извлечь бизнес из применения SOA, SaaS и облачных решений:

* Повышение эффективности предоставления услуг. Достигается более глубокое понимание, что же именно необходимо для бизнеса, за счет разработки измеримых компонентов (SOA). Компаниям, реализующим "облачный" подход с использованием SOA и SaaS нет нужны расходовать финансовые, человеческие и ИТ - ресурсы на поддержание собственной инфраструктуры.
* Гибкость. Сформированные службы обладают высокой совместимостью. Новые решения образуются за счет объединения служб ранее созданных решений, что повышает скорость реакции ИТ - служб на постоянно возникающие изменения в потребностях целевой аудитории.
* Масштабирование. Возможность быстро получить необходимое количество ресурсов обеспечивает гибкое масштабирование без внесения ощутимых организационных изменений.
* Доступность. Для доступа к "облачной" платформе необходимо лишь Интернет - соединение.
* Производительность. Территориально распределенные центры данных, ресурсы и платформы достигают величин производительности недоступных для инфраструктуры отдельной компании.
* Контроль. Возможность сочетания различных вариантов размещения приложений: на собственных серверах, на серверах "облака" или их сочетание, позволяет получить необходимую степень контроля за приложением и объединять имеющиеся ресурсы (локальные и "облачные") для решений текущих задач.

Разумеется, кроме преимуществ у "облачных" подходов есть и ряд рисков.

**Риски SOA и SaaS**

* Существенная прибыль и экономия возможна в случае, если производительность ранее используемого приложения была низка. В ином случае, переход на SaaS - решение может оказаться даже убыточным. Таким образом, не стоит направлять усилия на изменение решения, от которого не зависит прибыль.
* Сложности проектирования. Всегда есть риск, что спроектированные и реализованные решения не будет приносить заявленных преимуществ. Это может случиться по разным причинам: от ошибок проектирования, до несоответствия решения предъявляемым к нему требованиям (к примеру, конфиденциальность данных).
* Риски, связанные с реализацией решения. Для построения SOA - архитектуры и работы с "облачными" решениями коллектив должен обладать соответствующими компетенциями и опытом, не только в разработке, но и в проектировании решений. Влияние данного риска можно минимизировать, путем прохождения обучения и заключения партнерских соглашений с фирмой - поставщиком.
* Риски использования. Переход на новую модель управления и работы требует начальных инвестиций, для реорганизации технологических процессов и методов работы.

Таким образом, прежде чем начать переход на новый стиль работы, или принять решение о необходимости такового, необходимо:

1. провести анализ бизнес - функций, возможно сначала необходимо перевести на новую основу сравнительно небольшую и автономную часть бизнеса, сравнения эффективности "до" и "после";
2. сформировать план технологического развития - применение "облачных" решений должно соответствовать нуждам компании и ее клиентов, первоначально имеет смысл только дополнять уже существующие решения там, где это имеет смысл;
3. поддерживать план технологического развития - риски и проблемы будут возникать всегда, только по мере внедрения станет понятно, где и каким образом SOA, SaaS и "облака" могут быть наиболее полезны, необходимо всегда четкое понимание того, что происходит с бизнесом, какие есть проблемные места и как их можно устранить, либо степень их влияния.

**Список дополнительных материалов для самостоятельного изучения**

SOA

1. <http://www.ccc.ru/magazine/depot/06_02/read.html?0104.htm>
2. <http://www.osp.ru/cio/2008/12/5572736/>
3. <http://citforum.ru/internet/webservice/soa/>
4. <http://www.information-management.com/news/7992-1.html>
5. <http://www.information-management.com/news/8262-1.html>

SaaS

1. <http://www.ibs.ru/content/rus/608/6083-article.asp?archive=1>
2. <http://www.itpractice.ru/component/content/article/28-interview/221-saasinrussia.html>
3. <http://www.samag.ru/art/10.2010/10.2010_06.html>
4. <http://www.interface.ru/home.asp?artId=20217>

"Облачные" решения и бизнес

1. <http://www.zdnet.com/blog/hinchcliffe/eight-ways-that-cloud-computing-will-change-business/488>
2. <http://www.intel.com/cd/corporate/pressroom/emea/rus/414904.htm>

**Тема 3. Стратегия развертывания облака**

**Мифы о SaaS**

В данной части лекции мы рассмотрим несколько часто звучащих, наиболее распространенных в IT - кругах мифах о SaaS и их вариациях.

**Миф №1**. Хранить информацию на сервере сторонней компании небезопасно (или "Организация не может отказаться от услуг SaaS - поставщика, не потеряв данные").

По статистике, как правило утечка данных происходит по внутриорганизационным каналам. Т.е. риск потери данных из-за кражи сотрудниками значительно выше, чем сбои в работе поставщика SaaS - услуг. Риски существуют всегда, необходимо грамотно их соизмерять.

**Миф №2**. Невозможно принять оперативные меры, в случае некорректной работы арендуемого программного обеспечения.

SaaS -приложение, в настоящий момент, функционально не отличается от аналогичного desktop - приложения. Функции оперативного управления и контроля приложения, как правило, осуществляются самим клиентом, в то время как, сопровождение, обновление и т.п. - функции поставщика услуг.

**Миф №3**. Организация, оплатившая длительную аренду, переплачивает (или "Экономия при использовании SaaS неочевидна").

Зачастую, при расчете затрат упускают из вида, что в стоимость SaaS включены: ПО, оборудование, плата за электроэнергию, каналы связи, доступ, архивирование, оплата труда персонала.

**Миф №4**. SaaS подходит только для малого и среднего бизнеса.

Практика мировых поставщиков "облачных" услуг показывает, что доля крупных компаний среди клиентов SaaS увеличивается.

**Что и когда нужно переводить в облако**

Предыдущая Тема уже содержала положение , что не каждое решение или сервис следует немедленно переводить на "облачную" основу. В помощь проектировщикам решений, приведем некоторые соображения касательно того, в каком случае "облака" будут наиболее выгодными.

Эволюция развития сферы IT - услуг, в частности, концепция "облачных" вычислений позволяют предположить, что рано или поздно, но все компании будут либо формировать свои "облака", либо арендовать облачные платформы. При этом, очевидно, что построением собственных "облаков" займутся крупные компании, особенно те из них, кто уже вложился в формирование собственных центров обработки данных.

Согласно статистике наиболее востребованы следующие SaaS - приложения (см п. 4 списка дополнительных источников):

* Почта и коммуникации;
* Антивирусные системы;
* Службы технической поддержки;
* Проектный менеджмент;
* Дистанционное обучение.

Когда же следует задуматься об использовании "облака"? Самый очевидный ответ - в тот момент, когда нехватка IT - ресурсов станет очевидной и острой. Но, строго говоря, если такая ситуация возникает, то это говорит о том, что задуматься над реорганизацией IT - инфраструктуры следовало много раньше. Отметим, некоторые косвенные признаки, по наличию которых можно судить о назревающей необходимости в "облачном" обеспечении:

1. Необходимость быстро разворачивать новые версии сред разработки и тестирования.
2. Необходимость отсутствия ограничений на число экземпляров различных вариантов пользовательских сред.
3. Необходимость оперативного учета количества и стоимости потребляемых IT - ресурсов.

**Сценарии использования облака**

Существует два способа использования "облачных" технологий:

1. Размещение приложения в "облаке"

В данном случае, само приложение и данные полностью находятся в рамках "облака". Т.е. клиенту предоставляется полностью готовый к использованию сервис. При этом сам код и данные приложения развертываются либо как PaaS, либо как IaaS. В первом случае, код и данные помещаются в пакет, публикуемый в облаке, во втором - контроль и управление приложением осуществляется на уровне виртуальной машины.

2. Использование сервисов "облака"

Данные сервисы доступны посредством стандартных протоколов передачи данных в сети Интернет и, благодаря этому, не зависят от платформы или устройств. При этом, можно как использовать встроенные сервисы облачных платформ, так и размещать в облаке собственноручно созданные. При этом, могут быть использованы следующие варианты использования сервисов, при которых часть кода остается в рамках локальной инфраструктуры:

* перенос данных в "облако", их часть, или все данные остаются в локальной инфраструктуре;
* перенос кода приложения в "облако", все данные, или их часть остается в локальной инфраструктуре;
* интеграция - код и данные, размещенные в "облаке" взаимодействуют с приложениями локальной инфраструктуры.

**Стратегия развертывания облака**

Во - первых, перед началом процесса формирования "облака" необходимо убедиться в том, что это именно то, необходимое вам решение. Необходимо четко осознавать стоящие перед вами задачи и перспективы.

Если же вы решили формировать "облако", учитывая все вышесказанное, можно условно представить стратегию перехода на "облачную" основу, как последовательность этапов:

1. Пробный этап. Переход на облако, это прежде всего смена стратегии, переосмысление роли информационных технологий в организации, в целом. Таким образом, прежде чем менять IT - подход в организации, необходимо сменить "стиль" мышления соответствующим образом. Имеет смысл, предоставить своим сотрудникам возможность работы с публичными "облаками", следует перевести ряд задач на "облачную" основу. Для конечных пользователей "облаков" важно обладание четким пониманием, что такое "облако", какие задачи с помощью него можно решать. Данный этап поможет оценить степень готовности организации в целом к реорганизации IT - инфраструктуры, а также привить основные навыки работы с "облаками".
2. Формирование требований. После того, как в организации сформируются определенные правила и нормы, по началу скорее неформальные, использования облачных платформ, необходимо задуматься о формировании общеорганизационных требований к предоставляемым внешним поставщиком услугам. К примеру, необходимо определиться с тем, какие именно сервисы и платформы нужны для работы, сколько необходимо систем хранения данных. На данном этапе следует определиться с поставщиками "облачных" услуг, сформировать требования к архитектуре и функционалу используемых решений. Результатом данного этапа должен быть набор формализованных требований к "облакам" организации.
3. Построение частного облака. Данный этап актуален для крупных компаний. Началом данного этапа можно считать момент, когда большая часть сотрудников активно пользуется "облачными" сервисами в своей работе. При наличии экономической целесообразности, можно начать формирование своего корпоративного вычислительного облака, на базе имеющегося центра обработки данных. В конечном счете, эволюционным развитием подобного центра и является корпоративное "облако". При этом, формирование своего центра облачных вычисление не обязательно означает отказ от услуг внешних поставщиков подобных решений. Вполне может использоваться гибридная схема работы, когда для решения поставленных задач используется как частное, так и внешнее "облако".

**Список дополнительных материалов для самостоятельного изучения**

**Мифы о SaaS в России**

<http://www.e-xecutive.ru/knowledge/announcement/1341745/>

**Мифы об облачных вычислениях**

<http://www.ibm.com/ru/cloud/pdf/Dispelling_the_vapor.pdf>

**Критерии оценки на соответствие требованиям "облака"**

<http://www.bureausolomatina.ru/node/99>

**Облачные вычисления**

[http://www.devbusiness.ru/mkozloff/2010/10/26/будущее-облаков-россии](http://www.devbusiness.ru/mkozloff/2010/10/26/%D0%B1%D1%83%D0%B4%D1%83%D1%89%D0%B5%D0%B5-%D0%BE%D0%B1%D0%BB%D0%B0%D0%BA%D0%BE%D0%B2-%D1%80%D0%BE%D1%81%D1%81%D0%B8%D0%B8)

<http://www.osp.ru/nets/2010/07/13004617/>

<http://www.itsec.ru/articles2/Oborandteh/oblako-ili-korobka>

<http://materials.it-event.ru/470/SaaS.pdf>

<http://www.mobiset.ru/articles/text/?id=5186>

**Тема 4. Azure Services Platform. Windows Azure, Windows .Net Services**

**Введение в технологии Azure**

Платформа Windows Azure - это "облачная" платформа компании Microsoft, реализующая модель PaaS. Инструменты данной платформы предоставляют функционал для создания решения, включающее в себя "облачную" операционную систему и набор сервисов для разработчиков.

Иными словами, платформа Windows Azure представляет собой группу "облачных" технологий, каждая из которых предоставляет определенный набор сервисов.

Платформа Windows Azure является частью "облака" компании Microsoft, которое состоит из следующих категорий сервисов:

1. "Облачные" приложения (cloud-based applications) - представляют собой набор постоянно доступных, масштабируемых сервисов, размещенных в "облаке" Microsoft, которые потребители могут использовать напрямую. К примеру, к таковым относятся: Bing, Windows Live Hotmail, Office Live и т.д.
2. Программные сервисы (software services) - представляют собой набор SaaS - сервисов, таких как Exchange Online, SharePoint Online, Office Communications Online и т.д.
3. Платформенные сервисы ( platform services ) - используются, как платформа, представляющая собой публичное "облако", которую разработчики могут использовать для внедрения решений нового поколения. К данным сервисам, в частности относятся SQL Azure, AppFabric и Windows Azure.
4. Инфраструктурные сервисы (infrastructure services) - набор компонент платформы Windows Azure, обеспечивающих поддержку "облачных" инфраструктурных ресурсов.

Платформа Windows Azure включает в себя:

* Windows Azure - операционная система в "облаке", предоставляет вычислительные ресурсы, средства хранения данных и инструменты управления сервисами.
* SQL Azure - реляционная база данных, предоставляет основные возможности MS SQL Server по хранению данных, предоставляется как сервис.
* Windows Azure AppFabric - программные модули, обеспечивающие коммуникации (Service Bus) и контроль доступа ( Access Control ). Используются для обеспечения взаимодействий между приложениями потребителя и приложениями облака.

**Введение в Windows Azure**

По своей сути, Windows Azure представляет собой платформу для запуска windows - приложений и хранения данных в облаке.

Возможность использования различных экземпляров Windows Azure позволяет пользователям запускать разнообразные сервисы. При этом эластичность, безопасность, доступность и масштабируемость обеспечиваются самой платформой, предоставленной в виде сервиса.

Хранение данных в Windows Azure возможно с помощью ряда сервисов, в виде таблиц, бинарных данных и очередей, которые мы подробно рассмотрим в последующих Темах данного курса.

Windows Azure работает на базе машин, расположенных в дата - центрах компании Microsoft, доступ к платформе обеспечивается посредством Интернета.

Платформа Windows Azure создана на основе технологий виртуализации. В отличие от обычного предоставления мощностей виртуальных машин для удаленных пользователей, платформа azure управляется с помощью специального инфраструктурного слоя Windows Azure Fabric Controller, или "фабрики".

Задача фабрики заключается в организации всего вычислительного массива виртуализированных экземпляров Windows Azure в виде логически единой единицы вычислений, а также управление ресурсами, нагрузкой и всем жизненным циклом приложений и сервисов в целом. Иными словами, фабрика связывает вычислительные мощности различных машин в единое целое.

Сервисы выполнения Windows Azure, естественно, основаны на Windows - технологиях. Создавать и запускать на Windows Azure можно только приложения основанные на .Net Framework, к примеру, это могут быть ASP.Net или WCF - приложения. Не стоит воспринимать Windows Azure только как платформу, для веб - приложений, платформа также поддерживает фоновые процессы, не зависящие от веб - приложений.

Также стоит отметить, что сервисы хранения данных, предоставляемые Windows Azure могут использоваться, как локальными приложениями пользователя, так и приложениями, размещенными непосредственно на Azure - платформе.

Клиент может контролировать свое azure - приложение, редактируя вручную, или программно, его конфигурационный файл, имеющийся у каждого приложения.

Доступ к возможностям конфигурации и мониторинга состояния приложения осуществляется через соответствующий портал, при предоставлении клиентского Windows Live ID.

**Примеры применения Windows Azure**

Подведем итог, для чего именно может быть использована Windows Azure:

1. Размещение в "облаке" веб - приложения. При этом инфраструктура "облака" направляет запрос клиента к одному из экземпляров веб - роли виртуальной машины. Виртуальный веб - сервер данной роли, получив запрос активирует соответствующий код.
2. Параллельная обработка данных. В данном случае, при обработке больших массивов данных, клиент, к примеру, с помощью WPF-приложения обращается к веб-сервису, запрос на обработку данных помещается в очередь Azure Queue. Обработка осуществляется в асинхронном режиме. Результат помещается в Azure Table.
3. Объединение локальных вычислительных мощностей и ресурсов облака.

Однако, кроме вопроса "как использовать" есть и "когда целесообразно". Приведем ряд типовых ситуаций для ответа на второй вопрос:

* Реализация нового проекта (стартап). Основное преимущество использования технологий Windows Azure в данном случае заключается в том, что для реализации стартапа не требуется наличие масштабной IT - инфраструктуры. Разработчики будут платить только за фактически потребленные ресурсы. Приложение будет масштабироваться Windows Azure в зависимости от числа его пользователей, избавляя, таким образом, от риска переплаты за поддержание и сопровождение инфраструктуры при изначально малом количестве пользователей.
* Создание SaaS версии локального . Net - приложения. Windows Azure предоставляет стандартное .Net окружение и соответствующий функционал поддержки и сопровождения приложения. Излишне говорить, что и в данном случае отсутствует необходимость подержания зачастую избыточной, с точки зрения решаемых задач, IT - инфраструктуры.
* Создание приложения для корпоративных клиентов. Возможности Windows Azure позволяют развернуть на ее основе приложение необходимой сложности и функциональности. При этом, наибольший эффект от использования именно данной платформы может быть достигнуть при наличии временной зависимости использования приложения клиентами, за счет оплаты фактически используемых ресурсов платформы.

**.Net Services компоненты, примеры использования**

.Net Services представляют собой набор сервисов для решения задач доступа к облачным приложениям и коммуникациям.

Компоненты .Net Services:

1. Access Control - способ идентификации, при котором пользователь предоставляет приложению набор утверждений, в зависимости от этих утверждений приложение решает что именно позволено пользователю.
2. Service Bus. Целью данной шины является обеспечение доступа к сервисам приложения извне. Также она обеспечивает прохождение файерволла без открытия нового порта, если приложение должно быть доступным извне.
3. Workflow - данный компонент позволяет создавать композитные приложения для интеграции enterprise - приложений.

.Net Services могут быть использованы для реализации WF - приложений, на основе сервисов Workflow, организации доступа клиентов к приложению с различными технологиями идентификации(Access Control), для обеспечения доступа сторонних пользователей к одному из внутренних корпоративных приложений ( Service Bus ).

**Список дополнительных материалов для самостоятельного изучения**

**Технологии Microsoft Azure**

1. <http://www.oszone.net/10952/Microsoft-Azure>
2. <http://www.zdnet.com/blog/microsoft/microsofts-azure-cloud-platform-a-guide-for-the-perplexed/1671>
3. <https://partner.microsoft.com/ukraine/40084702>

**Windows Azure**

1. <http://www.microsoft.com/windowsazure/whitepapers/default.aspx>
2. <http://www.microsoft.com/windowsazure/faq/default.aspx>
3. <http://venture-biz.ru/informatsionnye-tekhnologii/110-windows-azure>

**.Net Services**

1. <http://channel9.msdn.com/Blogs/nigel.watling/What-are-NET-Services>
2. <http://developers.de/blogs/damir_dobric/archive/2008/10/27/windows-azure-net-services-overview.aspx>
3. <http://www.azurejournal.com/2008/11/azure-net-services-access-control/>

**Тема 5. Azure Services Platform. Microsoft SQL Services, Live Services**

**Группа облачных технологий SQL Service**

Рассматривая возможности облачных вычислений, сложно не заметить, что одной из наиболее привлекательных и очевидных функций, которые можно перенести в облако, является хранение и работа с данными.

Группа технологий, объединенных общим названием SQL Services включает в себя разные сервисы, связанные с данными, включая хранение, анализ, создание отчетов. Общей целью данной группы является предоставление набора облачных сервисов для хранения и работы с разнообразными видами данных.

При этом, главной задачей SQL Services является доступность. Интерфейсы сервисов доступны через SOAP и REST, что позволяет работать с данными не только на Windows - системах.

Обратите внимание, что проект SQL Services 9 июля 2009 был переименован в SQL Azure. В доступной литературе по платформе Azure могут использоваться оба названия.

SQL Azure:

* является первым облачным сервисом полностью поддерживающим реляционную модель баз данных;
* предоставляет возможность работы с T-SQL, SQL - запросами, хранимыми процедурами, представлениями данных и т.д.;
* совместим со всеми существующими инструментами работы с реляционными БД и интегрированными средами разработки (IDE), такими как SQL Server Management Studio и Visual Studio ;
* не требует от IT - персонала принципиально новых навыков и компетенций, все знания и опыт полностью применимы к SQL Azure ;
* поддерживает PHP.

SQL Azure базируется на технологиях Microsoft SQL. Работа SQL Azure базируется на Cloud Fabric, управляющем экземплярами баз данных, обеспечивающим их развертывание, обновление, администрирование и мониторинг.

Следует отметить ряд ограничений, которые присутствуют в текущей версии SQL Azure:

1. невозможно получить доступ к серверу БД на физическом уровне;
2. невозможно получить доступ к конструкциям уровня сервера, командам DBCC и системным представлениям;
3. не реализованы полнотекстовый поиск, связанные сервера, отслеживание изменений, распределенные транзакции и т.п.

Целевой аудиторией SQL Azure являются:

* независимые поставщики ПО;
* поставщики SaaS - программных продуктов, разрабатывающие приложения на основе Windows Azure;
* разработчики корпоративных приложений уровня отделов на базе Windows Azure.

Учитывая вышеизложенной, SQL Azure ориентирован на реализацию следующих сценариев:

* создание масштабируемого веб - приложения небольшими организациями, компаниями с веб - приложениями любого уровня;
* создание новых продуктов, или расширение существующих решений поставщиками SaaS - услуг;
* реализация корпоративного приложения приложение уровня отдела;
* реализация проекта консолидации данных - объединение нескольких источников данных в облаке.

**Обзор служб Live Service**

Основой набора служб под общим названием Live Services является идея использования данных различными приложениями, для которых эти данные не являются "родными".

Иными словами, все мы используем разнообразные Интернет - приложения, число которых постоянно растет, каждое из этих приложений хранит свои данные. К примеру, это могут быть контакты, для приложений обмена сообщениями. Так почему бы не открыть эти данные для других приложений, разумеется в условиях контроля доступа.

Для этой цели компанией Microsoft был собран набор ресурсов в группу - Live Services. К примеру, широко известный набор приложений Windows Live осуществляет контроль и управление данными при помощи Live Services.

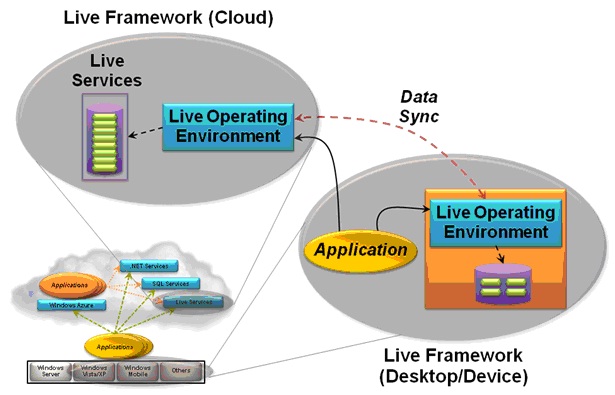


Рисунок 5.1 - Live Services

Доступ к данным Live Services осуществляется при помощи Live Framework, основой которого является среда Live Operating Enviroment (функционирующей в облаке) по протоколу HTTP, обеспечивая, таким образом, возможность доступа для приложений на .Net, Java, Java Script, RSS. Следует отметить, что доступ к данным может осуществляться как через облачный экземпляр Live Operating Enviroment, так и локальный, независимо от наличия соединения с облаком; в случае обращения через локальный экземпляр - доступ также осуществляется через HTTP запросы.

Управление Live Services осуществляется через портал Live Services Developert Portal.

Пользователь может объединить различные устройства на основе Windows (XP, Vista, 7, mobile) и Mas OS X в так называемый mesh (по сути - логическое объединение ряда устройств). При помощи Live Operating Enviroment определенные пользователем данные могут быть синхронизированы в перечисленных устройствах и облаке.

Подводя итог: Live Framework предоставляет инструменты для обеспечения доступа различных приложений к данным Live Services, а также предоставляет возможность синхронизации данных.

**Список дополнительных материалов для самостоятельного изучения**

**SQL Azure**

1. <http://www.brentozar.com/archive/2009/07/sql-azure-pricing-10-for-1gb-100-for-10gb/>
2. <http://blogs.msdn.com/b/sqlazure/archive/2009/07/09/9827971.aspx>
3. <http://www.microsoft.com/en-us/sqlazure/whitepapers.aspx>

**Windows Live**

1. <http://msdn.microsoft.com/en-us/windowslive/default.aspx>

**Live Framework**

1. <http://blogs.msdn.com/b/hlsdpe/archive/2008/12/16/live-framework-and-why-should-you-care.aspx>
2. <http://www.readwriteweb.com/archives/microsoft_live_mesh_first_look.php>

**Тема 6. Платформа Windows Azure**

Software+Services объединяет несколько феноменов, таких как SaaS, SOA и Web 2.0. Суть данной стратегии заключается в том, чтобы обеспечить на необходимом пользователю уровне комбинацию Интернет - сервисов и локального программного обеспечения. Иными словами, Software+Services - это предоставление нового уровня услуг, удобства и гибкости, отвечающих пользовательским потребностям.

Платформа Windows Azure является одним из основных компонентов стратегии Software+Services.

**Характеристика платформ**

Windows Azure - представляет собой Windows-платформу компании Microsoft, предоставляемой, как сервис ( PaaS ), развернутой на серверах и сопутствующей инфраструктуре дата - центров компании и имеющая доступ к Интернет. Т.е., Windows Azure - операционная система, предоставляемая, как сервис.

При использовании Windows Azure пользователи получают возможность запускать различные сервисы, при этом платформа обеспечивает масштабируемость, безопасность и доступность.

Windows Azure:

* добавляет возможности веб - служб существующим пакетным приложениям;
* позволяет создавать, изменять и распространять приложения через веб при наличии минимальной IT - инфраструктуры;
* предоставляет сервисы хранения большого количества данных, пакетной обработки и высокопроизводительных вычислений;
* обеспечивает возможности оперативного тестирования и распространения веб - служб при минимальных затратах;
* уменьшает издержки, связанные с содержанием IT - инфраструктуры;
* упрощает процесс управления IT - инфраструктурой.

В основе Windows Azure лежат технологии виртуализации, платформа управляется при помощи инфраструктурного слоя Windows Azure Fabric Controller, обеспечивающим развертывание служб и приложений, управление нагрузкой и ресурсами.

Также введем понятие экземпляра - в Windows Azure это единица развертывания, отражающая виртуальную машину с рядом предопределенных конфигураций. При этом Azure Fabric Controller отвечает за физическое развертывание виртуальных машин. Со стороны пользователя требуется лишь указать необходимое число экземпляров виртуальной машины, развертываемых для определенного сервиса.

Жизненным циклом экземпляров управляет Azure Fabric Controller, пользователь, в свою очередь, может запускать и останавливать экземпляры.

**Сервисы Windows Azure**

Рассмотрим подробнее категории сервисов, предоставляемых Windows Azure.

1. Сервисы хранения данных

Предоставляют масштабируемую систему хранения данных, с поддержкой таких структур, как таблицы, бинарные объекты, асинхронные очереди сообщений и обычные файловые системы. Подробнее сервисы хранения данных будут рассмотрены в лекции №7. Также стоит отметить, что данные сервисы поддерживают тройную репликацию в рамках кластера и центра обработки данных. Автоматическое создание копий и балансировка нагрузки между серверами обеспечивают требуемую масштабируемость сервисов хранения данных.

2. Вычислительные сервисы

Представляют собой контейнеры для приложений, с поддержкой .Net, Java, PHP, Python и т.д. С этой точки зрения, Windows Azure представляет собой прикладной контейнер, в котором размещаются код и логика "облачного" приложения.

3. Коммуникационные сервисы

Доступны посредством "облачной" сервисной шины, могут быть использованы, как средство мгновенного обмена сообщениями с другими сервисами на стороне, как "облака", так и клиента.

4. Сервисы безопасности

Включают в себя сервисы управления доступом, основывающихся на механизмах федерации (функция единого входа, через Интернет для проверки пользователя и предоставления ему доступа к различным приложениям во время одного сеанса работы), обеспечивая тем самым возможность интеграции с уже имеющимися системами управления идентификацией. Более подробную информацию см в п. №5 списка дополнительных материалов.

5. Прикладные сервисы

Сервисы, используемые при разработке "облачных" приложений и других сервисов.

**Роли**

Как правило, сервис Windows Azure имеет более одного экземпляра. Каждый экземпляр может выполнять как всю логику приложения, так и ее часть. При этом разработчик контролирует число и типы ролей.

Можно провести аналогию между ролями в Windows Azure и стандартными типами проектов в Visual Studio. В данном случае экземпляр Windows Azure представляет собой отдельный проект.

Роли Windows Azure:

1. Веб - роль ( web role )
2. Прикладная роль ( worker role )

Основной задачей веб - роли является обеспечение поддержки протоколов HTTP и HTTPS. Размещается роль на базе IIS. Таким образом веб - роль, фактически, соответствует ASP.Net проекту Visual Studio, с учетом отличий в сборках приложений и способе конфигурации.

Прикладная роль отвечает за поддержку внешних точек входа через TCP\IP и ряд портов (кроме 80 и 443). Данная роль не размещается на веб - сервере. Продолжая аналогию, эту роль можно сравнить с Windows - сервисами, также она может быть использована для выполнения фоновых задач.

Таким образом, роли в Windows Azure - это "блоки" из которых строится "облачное" приложение. Экземпляр роли - виртуальная машина с рядом предопределенных характеристик.

**Возможности платформ**

Нами уже был дан общий обзор возможностей "облачных" приложений, а также SaaS, PaaS и IaaS решений. Но это были концептуальные описания. Подводя итог данной лекции, мы хотим еще раз сделать акцент на практических возможностях Windows Azure.

Windows Azure:

* предоставляет "облачную" среду для разработки, хостинга и управления сервисами;
* представляет собой контейнер, содержащий код и логику, в рамках которого приложение может быть развернуто;
* предоставляет среду, схожую с существующей Windows Server средой;
* позволяет разворачивать .Net проекты напрямую, без внесения существенных и трудоемких изменений;
* позволяет хранить данные пользователей, поддерживает тройную репликацию.

**Список дополнительных материалов для самостоятельного изучения**

**Software + Services**

1. <http://msdn.microsoft.com/ru-ru/library/cc891480.aspx>

**Windows Azure - основы**

1. [http://www.youtube.com/watch?v=poDRw\_Xi3Aw (видео)](http://www.youtube.com/watch?v=poDRw_Xi3Aw%20%28%D0%B2%D0%B8%D0%B4%D0%B5%D0%BE%29)
2. [http://www.youtube.com/watch?v=CfYjDHVaXSo (видео)](http://www.youtube.com/watch?v=CfYjDHVaXSo%20%28%D0%B2%D0%B8%D0%B4%D0%B5%D0%BE%29)
3. <http://blogs.msdn.com/b/vorobiev/archive/2011/02/07/faq-windows-azure-platform-windows-azure.aspx>

**Сервисы безопасности**

1. <http://www.oszone.net/14366/Windows-Azure>

**Роли Windows Azure**

1. <http://blogs.technet.com/b/isv_team/archive/2010/12/29/3377670.aspx>

**Обзор возможностей Windows Azure (видео)**

1. <http://www.techdays.ru/videos/2898.html>

**Тема 7. Сервисы хранения данных в Windows Azure. VM – роль**

VM - роль в Windows Azure предназначена для облегчения процесса миграции существующих Windows Server приложений в "облачную" структуру.

VM - роль предоставляет сервисы на уровне инфраструктуры (IaaS), облегчая процессы контроля и управления инфраструктуры для администраторов и разработчиков.

Отметим, что остальные роли (веб и прикладная) находятся "над" VM -ролью, позволяя тем самым разработчикам, чья деятельность осуществляется на более высоких уровнях абстракции, концентрироваться непосредственно на приложении, а не на ОС, на котором оно базируется.

В отличие от переноса сервисов и отдельных компонентов приложения, VM - роль позволяет перенести приложение целиком, снижая стоимость владения им, за счет предоставления сервисов автоматического управления, обновления и отказоустойчивости.

Таким образом, роль виртуальной машины позволяет создавать пользовательский образ виртуального жесткого диска (на основе Windows Server 2008 R2) и размещать его в "облаке".

Фактически VM - роль это виртуальная машина, но обладающая рядом особенностей:

* не поддерживается сохранность образов при аппаратных сбоях;
* каждый сервис может иметь только один внешний IP - адрес;
* невозможен автоматический мониторинг приложений, функционирующих внутри виртуальной машины;
* для обеспечения требуемого уровня доступности требуется наличие двух одинаковых экземпляров виртуальной машины.

Основываясь на требованиях развертываемых приложений, разработчики могут выбирать соответствующий размер виртуальной машины. Экземпляры Windows Azure поддерживают пять конфигураций виртуальных машин:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 7.1 - Конфигурации виртуальных машин | | | | |
| **Название** | **Частота процессора (ГГц)** | **Число ядер** | **Объем памяти** | **Размер хранилища данных** |
| Очень маленькая (extra small) | 1 | 1 | 768 Мб | 20 Гб |
| Малая (small) | 1,6 | 1 | 1,75 Гб | 225 Гб |
| Средняя (medium) | 1,6 | 2 | 3,5 Гб | 490 Гб |
| Большая (large) | 1,6 | 4 | 7 Гб | 1000 Гб |
| Огромная (extra large) | 1,6 | 8 | 14 Гб | 2040 Гб |

Каждая виртуальная машина создается при развертывании прикладных сервисов. Все виртуальные машины располагаются в центрах обработки данных Microsoft.

О ценах и лицензировании можно узнать из п. №1 списка материалов для самостоятельного изучения.

**Сервисы хранения данныx**

Хранение данных в Windows Azure обеспечивается набором сервисов под общим названием Windows Azure Storage . Каждый из сервисов подходит для хранения определенного типа данных:

* Table - сервис, позволяет хранить структурированные данные в таблице, доступ к которым осуществляется через REST API ;
* Queue - сервис, позволяет организовать неограниченное хранилище сообщений;
* Blob - сервис, позволяет хранить текстовую и бинарную информацию в специально организованных контейнерах.

Отметим, что Table - хранилище не является аналогом реляционного хранилища данных. В данной концепции, под таблицей понимается колТема сущностей (Entities), подобных кортежам в реляционном подходе. Сущность же представляет собой набор свойств (Properties). Свойство же является парой "имя (name) - типизированное значение (typed value)". Продолжая аналогию, сущности можно соотнести с полями в таблице в реляционном хранилище.

В Table - хранилище сущности могут содержать различные свойства, находясь в одной таблице. Т.е. таблица в Table - хранилище не задает структуру хранимых сущностей.

Таким образом, таблица состоит из набора объектов, каждый из которых имеет набор названий свойств и их значений. Объект может иметь до 256 свойств.

В табличном хранилище, к примеру, могут храниться состояния веб - приложений, текущее состояние пользовательских объектов (заказ, группа контактов) и т.д.

Для хранения бинарных (blob) объектов используется интерфейс, позволяющий сохранять именованные файлы вместе с метаданными.

Blob - объекты делятся на:

* блочные, оптимизированные для потокового ввода - вывода, размер объекта не может превышать 200 Гб;
* страничные, оптимизированные для случайных операций ввода - вывода, размер объекта не может превышать 1 Тб.

Хранилище бинарных объектов больше подходит для хранения резервных копий, изображений, документов и т.д.

Очередь (Queue) является хранилищем сообщений, как правило, обеспечивает коммуникации между различными ролями Windows Azure. Сообщения хранятся в очередях, размер сообщений ограничен 8 Кб. При этом возможно использование неограниченного числа очередей, число сообщений в очереди также не имеет ограничений.

Подробнее о table , blob и queue сервисах будет рассказано в Темах №11-16.

Кроме специализированных сервисов хранения данных Windows Azure поддерживает традиционные файловые операции, за счет поддержки NTFS формата.

NTFS тома хранятся как отформатированные виртуальные диски в страничных бинарных объектах. Работающие приложения могут сохранять состояния на томах. Доступ к диску осуществляется через стандартный NTFS интерфейс .

Каждая учетная запись Windows Azure Storage может хранить до 100 Тб данных.

С точки зрения архитектуры, существует 3 основных уровня доступа к данных в Windows Azure Storage:

1. Front - End (FE) уровень. Принимает поступающие запросы, авторизация и аутентификация также осуществляются на данном уровне. После аутентификации запросы направляются на сервер (partition server), на соответствующем уровне. На FE уровне хранится карта секторов ( partition map ), что позволяет направлять запрос напрямую необходимому серверу.
2. Уровень секторов (partition layer). Данный уровень управляет разделением данных в системе, обеспечивая доступ к необходимым серверам, автоматическую балансировку нагрузки секторов. Единственный partition - сервер может обслуживать множество секторов.
3. Уровень распределенной файловой системы (Distributed and replicated file system - DFS). Уровень, на котором фактически хранится информация, отвечающий за распределение и репликацию данных по всем серверам хранения. Каждый из DFS серверов доступен для любого partition - сервера.

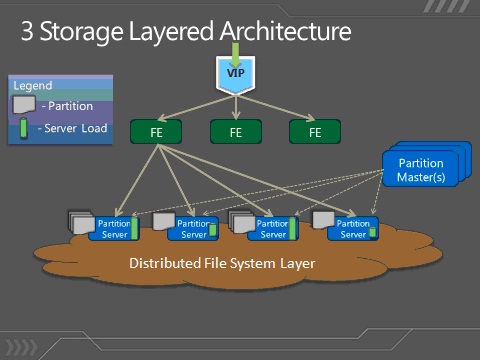


Рисунок 7.1 - Уровни доступа к данным Windows Azure Storage

**Список дополнительных материалов для самостоятельного изучения**

**VM - роль**

1. <http://www.microsoft.com/windowsazure/compute/#vmrole>
2. <http://msdn.microsoft.com/en-us/library/gg465398.aspx>

**Windows Azure Storage**

1. <http://www.oszone.net/13891/Windows-Azure-Storage>
2. <http://blogs.msdn.com/b/windowsazurestorage/archive/2010/12/30/windows-azure-storage-architecture-overview.aspx>
3. <http://blogs.msdn.com/b/windowsazurestorage/archive/2010/11/23/windows-azure-storage-client-library-potential-deadlock-when-using-synchronous-methods.aspx>

**Тема 8. SQL Azurе**

**Характеристики SQL Azurе**

В конце июля 2009 года компанией Microsoft был анонсирован SQL Azure.

SQL Azure:

* первый "облачный" сервис полностью поддерживающий реляционные модели данных;
* поддерживает SQL - запросы, T-SQL, использование хранимых процедур, представлений и т.п.;
* поддерживает модель безопасности Windows (пользователь\пароль);
* совместим практически со всеми инструментами и средами разработки реляционных баз данных (SQL Server Management Studio, Visual Studio);
* поддерживает PHP.

Таким образом, SQL Azure предоставляет масштабируемую, устойчивую базу данных, в качестве сервиса.

Работа SQL Azure базируется на Cloud Fabric, управляющем экземплярами баз данных, обеспечивающим их развертывание и сопровождение на протяжении всего жизненного цикла работы с данными.

**Сравнение с MS SQL Server**

В приведенной ниже таблице мы обобщили основные отличия MS SQL Server и SQL Azure:

|  |  |  |
| --- | --- | --- |
| Таблица 8.1 - Сравнение MS SQL Server и SQL Azure | | |
| **Функция** | **SQL Server** | **SQL Azure** |
| Хранилище данных | Нет ограничений на размер хранимых данных | Лимитировано, зависит от соглашения и, соответственно, оплаты. При превышении лимита хранения будут поддерживаться только SELECT и DELETE. Добавление и обновление данных вызовет ошибку. |
| Доступные издания | Express, Workgroup, Standart, Enterprise | Enterprise Edition |
| Доступ | Посредством: SQL Server Management Studio, SQLCMD | Посредством: SQL Server 2008 R2 Management Studio (более ранние версии имеют ограниченную поддержку), SQLCMD |
| Миграция данных | - | Поддерживаются: SQL Server Integration Services, BCP, SqlBulkCopyAPI |
| Аутентификация | Поддерживается SQL Server и Windows аутентификация | Только SQL Server аутентификация |
| Схемы | Нет ограничений | Не поддерживаются heap. Таблицы должны иметь кластерный индекс, прежде чем данные могут быть вставлены. |
| Поддержка TSQL | Нет ограничений | Ряд команд не поддерживается, или имеет ограничения. Более подробно см. п. № 3-5 списка дополнительных источников. |

**Ограничения**

Мы уже отмечали наличие ряда ограничений в SQL Azure, а именно:

1. невозможно получить доступ к серверу БД на физическом уровне;
2. невозможно получить доступ к конструкциям уровня сервера, командам DBCC и системным представлениям;
3. не реализованы полнотекстовый поиск, связанные сервера, отслеживание изменений, распределенные транзакции и т.п.

**Механизмы доступа**

Как уже упоминалось, SQL Azure поддерживает конструкции языка T-SQL через протокол TDS (Tabular Data Stream), а также обращения по протоколу ODBC.

Кроме того, имеется поддержка следующих технологий:

* ADO.Net 3.5 SP1
* LINQ
* WCF Data Services
* ADO.Net Entity Framework 3.5 SP1 4.0

Поддерживается использование в качестве клиентского приложения Microsoft Office 2010.

В случае, если технологии Microsoft не используются компанией, обращение к SQL Azure возможно при помощи драйверов SQL Server 2008 Native Client ODBC и SQL Server 2008 Driver for PHP, а также через REST - протокол.

**Организация хранения данных**

Есть три основных механизма, обеспечивающих работу с SQL Azure:

1. учетная запись
2. сервер
3. база данных

При этом, необходимо знать следующее.

Учетная запись является владельцем по отношению к серверу. У одной учетной записи может быть несколько серверов.

Сервер - логический элемент, функционально аналогичный Master DB в MS SQL Server. Сервер - это единица аутентификации, отчетности и георасположения.

База данных непосредственно хранит SQL - объекты в рамках сервера, такие как пользователи, таблицы, представления и т.п.

**Синхронизация данных**

Технология синхронизации данных в SQL Azure построена на основе технологии Microsoft Sync Framework.

Механизмы синхронизации позволяют связывать существующие хранилища данных, расположенные на стороне заказчика, с SQL Azure, обеспечивать доступ к локальным данным через платформу Windows Azure и реализовывать сценарии работы в отсоединенном режиме.

Особенностями SQL Azure Data Sync являются:

* Sync - расписание: пользователь может задавать интервал синхронизации самостоятельно;
* No - Code Sync конфигурация: пользователь задает данные, которые должны быть синхронизированы с использованием инструментария, поставляемого сервисом;
* Пользователь может задать таблицы для синхронизации между базами данных SQL Azure;
* Сервисы регистрации и мониторинга позволяют отслеживать статус синхронизации и их историю.

Выделяют три основных сценария, при которых может понадобиться синхронизация данных:

1. Синхронизация данных между MS SQL Server и SQL Azure.
2. Синхронизация данных между SQL Azure и хранилищами у сторонних поставщиков, с целью интеграции корпоративных приложений или интеграции данных и процессов между различными организациями.
3. Синхронизация данных между SQL Azure и приложениями в отсоединенном режиме. К примеру, распределенные мобильные решения сбора данных.

**Список материалов для самостоятельного изучения**

**Введение в SQL Azure**

1. <http://technet.microsoft.com/ru-ru/magazine/gg312148.aspx>

**Преимущества SQL Azure**

1. <http://blogs.msdn.com/b/vorobiev/archive/2011/02/08/faq-sql-azure.aspx>

**TSQL в SQL Azure**

1. Поддерживаемые команды
   * <http://msdn.microsoft.com/en-us/library/ee336270.aspx>
2. Команды с частичной поддержкой
   * <http://msdn.microsoft.com/en-us/library/ee336267.aspx>
3. Неподдерживаемые команды
   * <http://msdn.microsoft.com/en-us/library/ee336253.aspx>

**SQL Server 2008 R2 Native Client**

1. <http://msdn.microsoft.com/en-us/library/ms131415.aspx>

**SQL Server 2008 Driver for PHP**

1. <http://sqlsrvphp.codeplex.com/>

**SQL Azure Data Sync - пример**

1. <http://www.develop.com/sqlazure>

**Тема 9. Windows Azure AppFabric**

Windows Azure AppFabric (в дальнейшем - AppFabric), как видно из названия, является частью платформы Windows Azure и является промежуточным программным обеспечением (middleware).

По сути, функционал Windows Azure AppFabric - основное отличие "облачной" платформы Windows Azure от смежных решений компаний - конкурентов.

AppFabric позволяет пользователям "облака" получать доступ через локальную сеть к собственным "облачным" сервисам, обеспечивая интеграцию с существующими системами безопасности (Active Directory и т.д.).

Найти определение Windows Azure AppFabric не составляет труда - это программные сервисы обеспечения коммуникаций и контроля доступа, Service Bus и Access Control соответственно. Тем не менее, AppFabric остается одной из наиболее труднопонимаемых областей Windows Azure.

AppFabric предоставляет инструменты для создания приложений работающих не только в "облачной" среде, но и в рамках инфраструктуры заказчика, в том числе на платформах Windows Server, Java, Ruby, PHP и т.д.

Первое, что необходимо для начала знакомства с AppFabric - это адрес сайта, посвященного ему же: <http://www.microsoft.com/windowsazure/AppFabric/Overview/default.aspx>.

Практикующим разработчикам также пригодится следующий адрес - [http://blogs.msdn.com/b/windowsazureappfabric/](http://blogs.msdn.com/b/windowsazureappfabric/%20) - обновляемый ресурс советов, обновлений, шаблонов и примеров применения AppFabric.

По своей сути, AppFabric является звеном связывающим приложения на основе Windows Azure, а также иные приложения "облаком".

**Сервисы AppFabric**

**AppFabric Service Bus**

Предоставляет возможности для организации безопасного обмена сообщениями и распределенных, не связанных между собой приложений в "облаке", а также гибридные приложения на основе локальной инфраструктуры и "облака". Поддерживаются различные протоколы связи и обмена сообщениями.

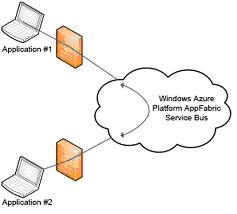


Рисунок 9.1 - AppFabric Service Bus

Service Bus используется для:

* установления связи между приложениями платформы Windows Azure с существующими приложениями и базами данных;
* связи между приложениями инфраструктуры заказчика;
* создания составных приложений.

К одним из главных преимуществ Service Bus можно отнести возможность установления связи между приложениями через проблемные границы сети (брандмаузеры, NAT и т.д.).

**AppFabric Access Control**

Сервисы Access Control обеспечивают управление доступом к приложениям и сервисам и интеграцию с имеющимися у заказчика средствами авторизации. Поддерживаются стандартные механизмы аутентификации (к примеру Windows Live ID, Active Directory). Основой сервиса Access Control является Windows Identity Foundation.

Таким образом, AppFabric Access Control:

1. Принимает запросы доступа от приложений.
2. Передает маркеры безопасности приложениям.
3. Выполняет проверку пользователей.

**Caching**

Сервисы кэширования AppFabric предоставляет, очевидно, распределенное кэширование для приложений на основе Windows Azure и SQL Azure. Сервис не требует установки, позволяет динамически изменять объем предоставляемой кэш-памяти.

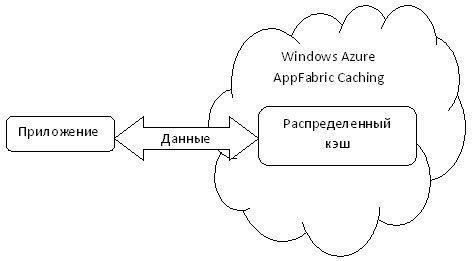


Рисунок 9.2 - Windows Azure AppFabric Caching

Сервис кэширования AppFabric позволяет:

* увеличить производительность ASP.Net приложений на базе Windows Azure, при отсутствии необходимости внесения изменений в код;
* уменьшает временную задержку доступа к данным, хранящимся в SQL Azure базах данных или Windows Azure Storage.

Безопасность авторизации и доступа обеспечивается сервисом Access Control.

**Integration**

Данный сервис еще не доступен для разработчиков, он должен появиться в составе CTP в 2011

Сервисы интеграции AppFabric обеспечивают возможности BizTalk Server в Windows Azure, на основе готовых шаблонов, упрощающих и ускоряющих процесс разработки.

По своей сути, данный сервис, предоставляет возможность объединения "облачных" приложений, локальных решений и приложений бизнес - партнеров в логическое целое [рис. 9.3](http://www.intuit.ru/studies/courses/677/533/lecture/6917?page=1#image.9.3).

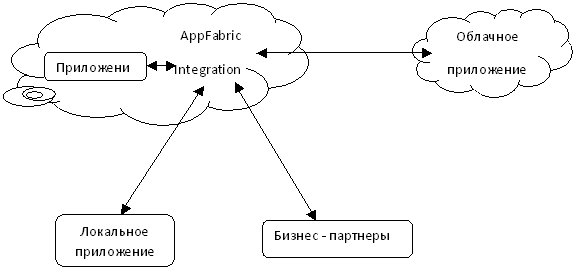


Рисунок 9.3 - Windows Azure AppFabric Integration

**CompositeApp**

Данный сервис еще не доступен для разработчиков, он должен появиться в составе CTP в 2011

CompositeApp сервис предоставляет мультитенатный, управляемый сервис, использующий .Net, основанный на AppFabric Composition Model, автоматизирующий развертывание приложения. Сервис напрямую запускает приложение, обеспечивая высокопроизводительную среду, оптимизированную для работы с "облачными" решениями. Также CompositeApp обеспечивает хостинг для веб - приложений, построенных на основе WCF и рабочих процессов, на основе WPF.

**Список материалов для самостоятельного изучения**

**Введение в Windows AppFabric**

1. <http://www.cloudave.com/10053/a-short-introduction-to-windows-azure-appfabric/>
2. <http://www.wadewegner.com/2010/05/what-is-the-azure-appfabric/>
3. <http://www.microsoft.com/windowsazure/AppFabric/Overview/default.aspx>
4. <http://blogs.msdn.com/b/windowsazureappfabric/archive/2011/01/27/introduction-to-windows-azure-appfabric-blog-posts-series-part-1-what-is-windows-azure-appfabric-trying-to-solve.aspx>
5. <http://msdn.microsoft.com/ru-ru/magazine/gg490345.aspx>

**Service Bus и Access Control**

1. <http://blogs.msdn.com/b/windowsazureappfabric/archive/2011/02/07/introduction-to-windows-azure-appfabric-blog-posts-series-part-2-the-middleware-services.aspx>

**Caching**

1. <http://blogs.msdn.com/b/windowsazureappfabric/archive/2011/02/23/introduction-to-windows-azure-appfabric-blog-posts-series-part-3-the-middleware-services-continued.aspx>

**Тема 10. Инструметарий разработчика Windows Azure**

**VS 2010 - Windows Azure tools**

Windows Azure Tools - расширение Visual Studio (VS), позволяющее создавать, тестировать и отлаживать решения для Azure без необходимости непосредственного подключение к Windows Azure. Созданное приложение, по готовности, может быть развернуто в Windows Azure.

Доступны инструменты для следующих продуктов VS:

* VS 2008 - 2010;
* Visual Web Developer 2008 - 2010.

Возможности инструментов Windows Azure:

* Поддержка MS .Net Framework 4.0. Платформа .Net Framework поддерживается, как целевая для разработки ролей, как в самой Windows Azure, так и в средах разработки.
* Обозреватель хранилищ Windows Azure. Представляет собой расширение для окна обозревателя серверов, предоставляющее доступ для чтения Blob - объектов и таблиц хранилища Windows Azure.
* Обозреватель вычислений Windows Azure. Представляет собой расширение для окна обозревателя серверов, предоставляющее доступ (ReadOnly) к состоянию развертывания "облачных" служб Windows Azure.
* Интегрированное развертывание. Возможность развертывания "облачной" службы непосредственно из среды разработки.
* IntelliTrace - отладка. Позволяет вести журнал записей расширенных отладочных сведений для "облачной" службы. Поддерживается пошаговое выполнение кода на основе журнала IntelliTrace.

Отметим, что не все возможности инструментов Azure Tools доступны для поддерживаемых VS - продуктов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 10.1 - Возможности AzureTools | | | | |
| **Возможность** | **VS 2008** | **VS 2010** | **Visual Web Developer 2008** | **Visual Web Developer 2010** |
| Поддержка MS .Net Framework 4.0 | - | + | - | + |
| Обозреватель хранилищ | - | + | - | - |
| Обозреватель вычислений | + | + | - | - |
| Интегрированное развертывание | + | + | + | + |
| IntelliTrace - отладка | - | + | - | - |

В частности, Windows Azure SDK версии 1.3:

* поддерживает роль виртуальной машины - создание пользовательских образов виртуального жесткого диска (бета - версия);
* обеспечивает доступ к удаленному рабочему столу - подключение к отдельным экземплярам службы с клиентского приложения;
* обеспечивает поддержку IIS в веб-ролях, веб-роль Windows Azure может быть размещена во внешней IIS среде;
* поддерживает виртуальные сети (CTP);
* предоставляет расширенный инструментарий диагностики Windows Azure Diagnostics.

Для установки и полноценной работы с SDK 1.3 для Windows Azure необходимо наличие .Net Framework 3.5 SP1, IIS 7.0, MS SQL Server 2005-2008 (в т.ч. Express).

**Development Fabric (DF)**

Данный инструментарий непосредственно позволяет разрабатывать и разворачивать "облачное" приложение. А также тестировать и отлаживать его.

DF позволяет локально разрабатывать код для Windows Azure. Локальная эмуляция поддерживается в Windows Vista SP1 и выше, Windows 7 и Windows Server 2008.

В состав DF входят следующие исполняемые файлы:

* DFAgent.exe;
* DFLoadBalancer.exe;
* DFMonitor.exe.

С ноября 2010 Development Fabric переименован в "compute emulator" (начиная с SDK 1.3).

**Development Storage (DS)**

Development Storage является средством. позволяющим эмулировать SQL Azure на локальной рабочей станции. При этом в качестве локального хранилища может использоваться MS SQL Server 2005 - 2008, в том числе в Express - издании.

DF позволяет:

1. разрабатывать структуру хранилища;
2. создавать локальное хранилище;
3. запускать и тестировать приложение.

С ноября 2010 Development Storage переименован в " storage emulator" (SE) (начиная с SDK 1.3).

Существует ряд отличий между сервисами хранения данных Windows Azure и storage emulator, которые необходимо учитывать, при разработке приложений:

* SE поддерживает только один, фиксированный аккаунт и ключ аутентификации. Данные ключ и аккаут единственно возможные для работы с локальным хранилищем.
* SE не является масштабируемой службой, т.е. не может поддерживать большое число пользователей.
* SE поддерживает отличную от "облачной" URI - схему. Отличие в том, что имя учетной записи указывается как часть иерархического URI пути, а не как часть доменного имени, поскольку использование доменных имен доступно в "облаке", но не локально.
* SE поддерживает Blob - хранилище до 2 Гб
* размер строки таблицы ( Table service ) не может превышать 1 Мб в SE

Средства разработки на отличных от Microsoft платформах

Ранее упоминалось, что помимо поддержки средств разработки Microsoft инструментарий разработчика Azure также включает в себя:

1. Инструменты для работы с Eclipse
   * создание новых проектов
   * проектный менеджмент
   * развертывание проекта в Windows Azure
   * средство просмотра Windows Azure хранилищ
2. Windows Azure SDK для PHP
   * Zend Framework
   * PHP классы для Blob, Table и Queue сервисов
   * протоколирование операций
   * средства командной строки - Windowa Azure Command - line Tools for PHP
3. Windows Azure SDK для Java
   * Java классы для Blob, Table и Queue сервисов
   * возможность обзора Windows Azure хранилищ

Доступ к сервисам Azure можно получить посредством стандартов REST, SOAP, Atom, обеспечивая тем самым совместимость с большинством платформ.

**Список дополнительных материалов для самостоятельного изучения**

**Центр разработчиков Windows Azure**

1. <http://www.microsoft.com/windowsazure/getstarted/>
2. <http://msdn.microsoft.com/ru-ru/windowsazure/default.aspx>

**Совместимость с языками программирования**

1. <http://www.interoperabilitybridges.com/projects/tag/Azure.aspx>

**Инструменты для работы с Eclipse**

1. <http://www.windowsazure4e.org/>

**Инструменты для работы с PHP**

1. <http://phpazure.codeplex.com/>

**Compute Emulator**

1. <http://msdn.microsoft.com/en-us/library/gg432968.aspx#bk_Develop>

**Storage Emulator**

1. <http://msdn.microsoft.com/en-us/library/gg432983.aspx>

**Тема 11. Структурированное хранилище данных Windows Azure Table**

Любое приложение вне зависимости от способа его размещения должно где-то хранить данные. При этом данные могут хранить локально, либо удаленно. "Облачный" способ хранения данных предлагает несколько иной способ. Обратив внимание на следующие особенности, которые необходимо учитывать при переносе традиционной реляционной структуры данных в Table Storage в "облаке":

* Структура данных удаленного хранения должна быть предельно простой. Чем проще структура данных, тем выше производительность доступа к данным нескольких конкурирующих приложений.
* Table Storage подходит для хранения реляционных данных, но само по себе данное хранилище реляционным не является. Это значит, то при переносе реляционной структуры данных в "облако", управлять ограничениями между субъектами хранения нужно будет пользователю.

**Характеристики**

Windows Azure Table поддерживает:

* LINQ
* ADO.Net Data Services
* REST
* неограниченное число таблиц и сущностей, без ограничения размеров
* целостность каждой сущности
* блокировку обновлений и удалений
* возможность возврата частичных результатов запросов прерванных по времени ожидания, при этом имеется возможность продолжить дальнейшее выполнение запроса.

**Модель данных**

Для доступа к Windows Azure Table у приложения должна быть учетная запись. После создания учетной записи, пользователю предоставляется секретный ключ, используемый для аутентификации.

Ключевыми понятиями Table Storage являются:

* Таблица - содержит набор сущностей.
* Сущность - логически является строкой в таблице. Основной элемент данных. хранящихся в таблице. Содержит набор свойств.
* Свойство - Значение, хранимое в сущности.

Как мы уже отмечали ранее, проводя аналогии с реляционным подходом, получим следующее: под таблицей понимается колТема сущностей (Entities), подобных кортежам в реляционном подходе. Сущность же представляет собой набор свойств (Properties). Свойство же является парой "имя (name) - типизированное значение (typed value)". Сущности можно соотнести с полями в таблице в реляционном хранилище.

* ключ секции - свойство ключа таблицы. Используется для распределения по узлам хранения сущностей таблицы.
* ключ строки - свойство ключа таблицы, уникальный идентификатор сущности.
* временная метка - система хранит различные версии одной и той же сущности, для их различения используется временна метка.
* секция - набор сущностей с одинаковыми ключами секции.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 11.1 - Пример: | | | | | |
| **Ключ секции** | **Ключ строки** | **Свойство 1** | **...** | **Свойство k** |  |
| Отчетность1 | 1 | Значение свойства | ... | Значение свойства | Секция 1 |
| Отчетность1 | 2 | Значение свойства | ... | Значение свойства | Секция 1 |
| Документ1 | 1 | Значение свойства | ... | Значение свойства | Секция 2 |
| Документ1 | 2 | Значение свойства | ... | Значение свойства | Секция 2 |
| Документ1 | 3 | Значение свойства | ... | Значение свойства | Секция 2 |

* порядок сортировки - в CTP версии предоставлен только один индекс, сортирующий сначала по ключу секции, затем - по ключу строки.

Ограничения таблиц, сущностей и их свойств:

1. Имена свойств таблиц должны состоять только из букв и цифр.
2. Имя таблицы не должно начинаться с цифры.
3. Имена таблиц различают регистры.
4. Длина имени таблицы должна быть в пределах от 3 до 63 символов
5. Сущность может иметь не более 255 свойств
6. Свойства "ключ секции" и "ключ строки" не могут быть больше 1Кб размером.
7. Свойство "временная метка" является ReadOnly.
8. Windows Azure Table не хранит схем, т.е. значения свойств сущностей одной таблицы могут относиться к разным типам данных.
9. Суммарный объем всех данных не может превышать 1Мб

|  |  |
| --- | --- |
| Таблица 11.2 - Поддерживаемые WAT типы данных: | |
| **Тип** | **Описание** |
| Binary | Массив байтов до 64Кб |
| Bool | Булевское значение |
| Datetime | 64-битное значение временного UTC - формата, от 1.1.1600 до 12.31.9999 |
| Double | 64-битное действительное число |
| GUID | 128-битный уникальный идентификатор |
| Int | 32-битное целочисленное значение |
| Int64 | 64-битное целочисленное значение |
| String | Значение кодировки UTF-16, размером до 64Кб |

**Секционированиe**

Частично секционирование было нами пояснено в предыдущей части настоящей лекции. Рассмотрим его более детально.

Масштабируемость системы хранения данных зависит от распределения секций по узлам хранения.

Система распределяет секции по узлам хранения по результатам истории использования секций, т.е. если ряд секций запрашивается чаще остальных, то система автоматически распределит их по нескольким узлам хранения, распределяя трафик между несколькими серверами. Важно помнить, что при этом объем секции не ограничен размерностью узла хранения.

Кроме того, хранение сущностей одной секции вместе позволяет повысить эффективность кэширования и других методов повышения производительности.

Приложение может осуществлять несколько операций создания, обновления и удаления для набора сущностей сформировав один пакетный запрос к системе, в случае, если операции осуществляются над сущностями одной таблицы и секции. При выполнении операции происходит изоляция моментального снимка, вне зависимости от исхода самой операции, т.о. все последующие операции, выполняющиеся в это же время, осуществляются над снимком, сделанным до начала операции. Результат транзакции становится доступным только после ее успешного завершения.

**Особенности выбора ключа секции**

Выбор ключа секции - задача, которая сложнее чем кажется. С одной стороны, размещение сущностей в одной секции позволяет оптимизировать выполнение запросов, с другой - чем больше секций в одной таблице, тем проще для WAT распределить сущности по местам хранения, обеспечивая масштабируемость таблицы.

1. Если планируется использовать запросы над группами сущностей, необходимо подбирать ключ секции таким образом, чтобы все сущности участвующие в запросе находились в одной секции, т.е. необходимо группировать сущности, участвующие в одном запросе.
2. В наиболее частых и важных запросах, помимо прочего следует указывать и ключ секции, поскольку при его отсутствии необходимые сущности будут отбираться просмотром всех секций таблицы.
3. Проверка масштабируемости: нагрузочное тестирование секции таблицы с целью проверки возможности обеспечения секцией необходимой производительности.

**Список дополнительных материалов для самостоятельного изучения**

**Обзор Windows Azure Table**

1. <http://blogs.msdn.com/b/jnak/archive/2010/01/06/walkthrough-windows-azure-table-storage-nov-2009-and-later.aspx>

Windwows Azure Table - подробный обзор

1. <http://msdn.microsoft.com/ru-ru/library/ee872426.aspx>
2. <http://www.bandgap.cs.rice.edu/classes/comp410/resources/Using%20Azure/Using%20Azure%20Table%20Storage.aspx>

**Тема 12. Windows Azure Table - программирование**

**Базовые операции таблиц и сущностей**

* создание;
* извлечение с применением фильтров;
* обновление (только сущности);
* удаление;
* транзакции над группами сущностей, находящимися в одной таблице и секции;

Таблица интерфейсов программирования приложений, поддерживаемых таблицами и сущностями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 12.1 - Интерфейсы программирования приложений | | | | |
| **Операция** | **ADO.NET Data Services** | **HTTP-команда** | **Ресурс** | **Описание** |
| Запрос | LINQ Query | GET | Таблица | Возвращает список всех таблиц данной учетной записи хранилища. В случае наличия фильтра таблицы возвращаются соответственно фильтру. |
| Сущность | Возвращает все сущности заданной таблицы или подмножество сущностей, если задан фильтр. |
| Обновление всей сущности | UpdateObject & SaveChanges(SaveChangesOptions.ReplaceOnUpdate) | PUT | Сущность | Обновляет значения свойств сущности. Операция PUT замещает всю сущность и может использоваться для удаления свойств. |
| Частичное обновление сущности | UpdateObject & SaveChanges() | MERGE | Сущность | Обновляет значения свойств сущности. |
| Создание новой сущности | AddObject & SaveChanges() | POST | Таблица | Создает новую таблицу в это учетной записи хранилища. |
| Сущность | Вставляет новую сущность в названную таблицу. |
| Удаление сущности | DeleteObject & SaveChanges() | DELETE | Таблица | Удаляет таблицу в данной учетной записи хранилища. |
| Сущность | Удаляет сущность из названной таблицы. |
| Транзакция над группой сущностей | SaveChanges(SaveChangesOptions.Batch) | POST | $batch | Поддержка транзакции над группой сущностей обеспечивается посредством пакетной операции над сущностями одной таблицы с одинаковым ключом секции. В ADO.NET Data Services опция SaveChanges требует, чтобы запрос выполнялся как одна транзакция. |

Расширенные операции над таблицами и сущностями:

* разбиение на страницы;
* обработка конфликтов параллельных обновлений.

**Контроль версий**

Для обеспечения контроля версия для всех решений Windows Azure Storage был введен HTTP-заголовок "x-ms-version". Все изменения API хранилища регистрируются как версии с помощью этого заголовка. Это обеспечивает возможность выполнять предыдущие версии команд и при этом расширять возможности существующих команд и вводить новые.

Заголовок "x-ms-version" должен задаваться для всех запросов к Windows Azure Storage. При поступлении анонимного запросе без указания версии система хранения выполнит самую старую из поддерживаемых версию этой команды.

**Класс сущностей**

Схема таблицы описывается как C#-класс. Такую модель использует ADO.NET Data Services. Схема известна только клиентскому приложению и упрощает доступ к данным. Сервер схему не применяет.

Для примера рассмотрим работу с таблицей контактов - Contact.

Рассмотрим описание сущностей Contact , хранящихся в таблице Contacts. Каждая сущность содержит следующие данные:

1. Группа контактов - Group;
2. Имя контакта - NameOf;
3. Дату создания контакта - DateOf;
4. Адрес электронной почты - Email;
5. Комментарии - Comment.

Для данной таблицы "Contacts" в качестве ключа секции используется Group, а в качестве ключа строки - NameOf. PartitionKey и RowKey – ключи таблицы Contacts, они объявляются посредством атрибута класса DataServiceKey (Ключ сервиса данных). Кроме ключей, в качестве свойств объявлены характерные для данного вида сущностей атрибуты. Все свойства имеют открытые (public) методы считывания и присвоения значения и хранятся в таблице Windows Azure Table . Итак, в примере ниже:

public class Contacts

{

public string PartitionKey { get; set; }

public string RowKey { get; set; }

public datetime DateOf { get; set; }

public String Email { get; set; }

public String Comments { get; set; }

}

**Создание таблиц**

Далее рассмотрим, как создать таблицу Contacts для учетной записи хранилища. Создание таблицы аналогично созданию сущности в основной таблице "Tables" . Эта основная таблица определена для каждой учетной записи хранилища, и имя каждой таблицы, используемой учетной записью хранения, должно быть зарегистрировано в основной таблице. Описание класса основной таблицы приведено ниже, где свойство TableName (Имя таблицы) представляет имя создаваемой таблицы.

public class TableStorageTable

{ public string TableName { get; set; }

}

Фактическое создание таблицы происходит следующим образом:

// Uri сервиса: "http://<Account>.table.core.windows.net/"

DataServiceContext context = new DataServiceContext(serviceUri);

TableStorageTable table = new TableStorageTable("Contacts ");

// Создаем новую таблицу, добавляя новую сущность

// в основную таблицу "Tables"

context.AddObject("Tables", table);

// результатом вызова SaveChanges является отклик сервера

DataServiceResponse response = context.SaveChanges();

serviceUri – это uri сервиса таблицы, http://<Здесь указывается имя учетной записи>.table.core.windows.net/. DataServiceContext (Контекст сервиса данных) – один из основных классов сервиса данных ADO.NET, представляющий контекст времени выполнения для сервиса. Он обеспечивает API для вставки, обновления, удаления и запроса сущностей с помощью либо LINQ , либо RESTful URI и сохраняет состояние на стороне клиента. Более подробный пример работы с Windows Azure Table, начиная от создания приложения, будет рассмотрен в рамках соответствующего практического занятия.

**Список материалов для самостоятельного изучения**

**Программирование Windows Azure Table**

1. <http://windowsclient.net/blogs/anshulee/archive/2010/05/27/best-practices-for-ado-net-dataservices-when-using-azure-table-storage.aspx>
2. <http://mscerts.net/programming/programming%20windows%20azure%20%20%20table%20operations%20-%20understanding%20pagination.aspx>
3. <http://mscerts.net/programming/programming%20windows%20azure%20%20%20table%20operations%20-%20querying%20data.aspx>

**Тема 13. Windows Azure Blob: введение, модель данных, REST - интерфейс**

**Общие сведения**

Аббревиатура BLOB расшифровывается как Binary Large Object, т.е. большой бинарный объект - массив двоичных данных. В СУБД BLOB - специальный тип данных, предназначенный, в первую очередь, для хранения медиа- информации и компилированного программного кода.

Blob - хранилище в Azure может быть представлено в виде специального табличного хранилища в "облаке". Windows Azure Blob расширяет табличное хранилище и рассчитано на хранение больших объемов информации. Разница между Table и Blob хранилищами заключается в следующем:

1. Табличное хранилище для управления таблицами использует ключи секций и строк. Бинарное хранилище для управления использует контейнер хранения (storage container) и идентификатор (blob ID).
2. Табличное хранилище может работать со всеми основными форматами данных, как то символы, строки, целые и действительные числа, XML и т.д. Бинарное хранилище рассчитано только на, очевидно из названия, бинарный тип данных, при этом данных хранятся в виде блоков (data chunks).

Доступ к Blob - хранилищу осуществляется через пользовательскую учетную запись. Одна учетная запись может создать несколько Blob - контейнеров. В свою очередь, Blob - контейнер может включать в себя несколько Blob - объектов. [(см. рис 13.1)](http://www.intuit.ru/studies/courses/677/533/lecture/6926?page=1#image.18.1).

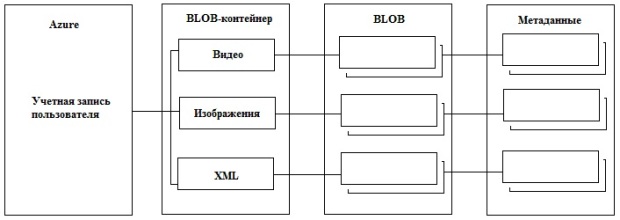


Рисунок 13.1 - Блоб - объекты

**Модель данных**

Рассмотрим модель данных Windows Azure Blob более детально. Выделим основные термины:

* Учетная запись. Любой доступ к Windows Azure Storage и его сервисам, осуществляется посредством учетной записи. Как уже отмечалось, одна учетная запись может иметь несколько Blob - контейнеров.
* Blob - контейнер. Контейнер группирует Blob - объекты. При этом политики использования данных задаются на уровне контейнера.
* Blob (blob - объект). Хранится в контейнере. Каждый объект может быть размером до 50 Гб и имеет уникальное строковое имя в рамках контейнера. С бинарными объектами могут быть ассоциированы метаданные [(рис 13.1)](http://www.intuit.ru/studies/courses/677/533/lecture/6926?page=1#image.18.1), задающиеся в виде пары "имя - значение" размером до 8Кб.

[На рис 13.2](http://www.intuit.ru/studies/courses/677/533/lecture/6926?page=1#image.18.2) приведена структура данных Windows Azure Blob, полученная средствами MS SQL Server 2008.

Отметим ряд особенностей работы с бинарными объектами и контейнерами:

* контейнеры хранятся распределено;
* область действия одного контейнера ограничена учетной записью пользователя;
* при удалении контейнера, возможно возникновение задержек при повторном его создании, особенно при наличии большого количества объектов, т.е. до тех пор пока система не очистит blob - объекты попытки создать контейнер с тем же именем, что и удаляемый, будут вызывать ошибку;
* подтверждение команд создания и удаления контейнера возвращаются от сервера клиенту, даже в случае если данные процессы занимают длительное время.

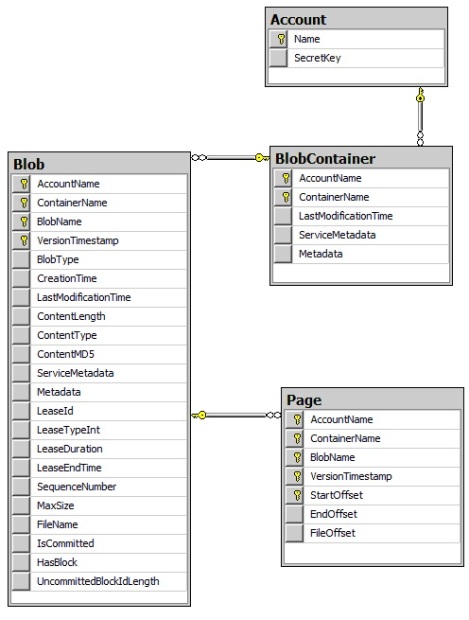


Рисунок 13.2 - Структура данных Windows Azure Blob

**REST - интерфейс Blob - объектов**

Любой доступ к Windows Azure Blob осуществляется через стандартные HTTP - команды PUT, Get и DELETE интерфейса REST.

Приведем перечень поддерживаемых HTTP/REST команд операций с Blob - объектами и контейнерами:

|  |  |  |
| --- | --- | --- |
| Таблица 13.1 - Blob-операции | | |
| **Операции** | **Метод HTTP** | **Описание** |
| List Containers | GET | Список всех контейнеров данной учетной записи |
| Create Container | PUT | Создание нового контейнера в рамках учетной записи |
| Get Container Properties | GET/HEAD | Возвращает все свойства и метаданные контейнера |
| Get Container Metadata | GET/HEAD | Возвращает только определенные пользователем метаданные указанного контейнера |
| Set Container Metadata | PUT | Задает заголовки метаданных контейнера |
| Get Container ACL | GET/HEAD | Получение списка управления доступом ( Access Control List - ACL ) и политик доступа контейнера |
| Set Container ACL | PUT | Задает ACL и политики доступа контейнера |
| Delete Container | DELETE | Удаление контейнера и всех его blob - объектов |
| List Blobs | GET | Список всех blob - объектов контейнера |
| Put Blob | PUT | Создание нового blob - объекта, или перемещение существующего |
| Get Blob | GET | Чтение или загрузка blob - объекта, включая все свойства и метаданные |
| Get Blob Properties | HEAD | Возвращает все свойства и метаданных blob - объекта |
| Set Blob Properties | PUT | Задает свойства blob - объекта |
| Get Blob Metadata | GET/HEAD | Возвращает заголовки метаданных blob - объекта |
| Set Blob Metadata | PUT | Удаление blob - объекта |
| Delete Blob | DELETE | Установка одноминутной блокировки записи blob-объекта. |
| Lease Blob | PUT | Создание снимка blob - объекта |
| Snapshot Blob | PUT | Копирование исходного blob- объекта в blob - назначения, в рамках одной и той же учетной записи |

**Список материалов для самостоятельного изучения**

**Windows Azure Blob**

1. <http://msdn.microsoft.com/ru-ru/library/ee872420.aspx>

**Windows Azure Blob - блоки и страницы**

1. <http://msdn.microsoft.com/en-us/library/ee691964.aspx>

**Blob Service API**

1. <http://msdn.microsoft.com/ru-ru/library/dd135733.aspx>
2. <http://blogs.msdn.com/b/windowsazurestorage/archive/2011/02/18/windows-azure-blob-md5-overview.aspx>

**Тема 14. Windows Azure Blob, как список блоков.**

**Примеры REST - запросов**

**Блоки и страницы**

Кроме уже рассмотренных нами основ blob - объектов и хранилищ, необходимо так же пояснить термины "блок" и "страница".

"Блок" и "страница" - способы организации хранения бинарных объектов.

Blob -блок

Бинарный объект размером до десятков гигабайт, с целью обеспечения его эффективной загрузки, разбивается на блоки.

Блоки оптимизированы для потоковой передачи информации. Запись блоков осуществляется в два этапа: сначала загружаются отдельные блоки информации, являющиеся частью одного бинарного объекта, затем необходимо подтвердить добавление блоков. В течении процесса подтверждения можно добавлять, удалять и изменять блоки бинарного объекта. При создании блока нет необходимости задавать его размер, блоки просто добавляются бинарному объекту. Также необязательно создавать блоки в строго определенной последовательности, упорядочить и редактировать блоки можно позднее.

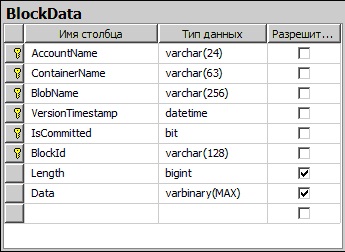


Рисунок 14.1 - Структура и атрибуты блока

Подытоживая вышесказанное. Блок бинарных объектов состоит из частей (блоки блока) , каждый из которых идентифицируется по идентификатору (ID). Создать или изменить blob-блок можно после загрузки его частей. Максимальный размер каждой части блока 4Мб, сам блок бинарного объекта ограничен размером в 200Гб, или до 50000 частей.

Blob - страница

Как blob-блоки состоят из блоков, так и blob-страницы представляют собой коллекцию страниц.

При создании страницы необходимо указать ее предельный размер. Добавление или обновление blob-страницы осуществляется при помощи Put Page операции.

Чтение и запись данных в страницу можно начать с любого произвольного набора данных.

Blob - страница ограничена размером 1Тб.

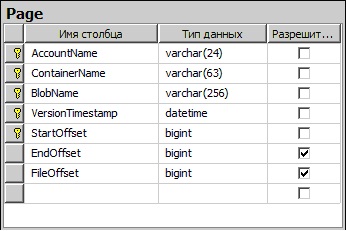


Рисунок 14.2 - Структура и атрибуты страницы

**Windows Azure Blob, как набор блоков**

При загрузке blob - объекта размером в десятки гигабайт реализовывается следующий сценарий [(рис. 14.3)](http://www.intuit.ru/studies/courses/677/533/lecture/6927?page=1#image.19.3):

* загружаемый объект разбивается на блоки, максимальный размер которых 4Мб;
* каждому блоку присваивается уникальный идентификатор в пределах бинарного объекта;
* блоки загружаются в "облако";
* после загрузки всех составных блоков бинарного объекта, определяются список блоков, которые должны использоваться blob - объекте.

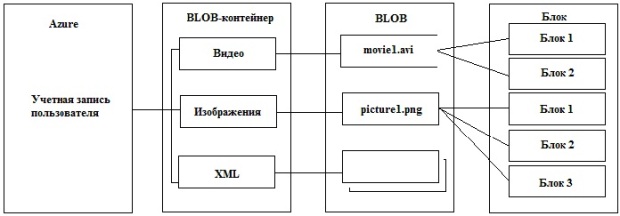


Рисунок 14.3 - Сценарий блоб - объектов

Доступ к blob - объектам, осуществляется при помощи операций PUT и GET.

Таким образом, Blob-блок может быть создан:

1. при размере меньше 64Мб, он может быть загружен при помощи Put Blob операции;
2. при размере более 64Мб, blob разбивается на части размером 4Мб или меньше, которые после загрузки собираются в определенном порядке.

**Примеры REST запросов**

Приведем примеры запросов для размещения блока PUT Block и чтения blob - объекта Get Blob. Примеры других REST-запросов можно найти по ссылкам в списке материалов для самостоятельного изучения.

В данных примерах имя учетной записи обозначено как <account>, работа производится с объектом "BVideo.mp4" в контейнере "video".

Также отметим, что для всех решений Windows Azure Storage введен новый HTTP-заголовок "x-ms-version". Все изменения в API хранилища регистрируются как версии с помощью этого заголовка.

Заголовок x-ms-version должен быть задан для всех запросов к Windows Azure Storage.

PUT Block

Ниже представлен пример REST-запроса для размещения блока размером 4МБ посредством операции PUT block. Параметр запроса "?comp=block" указывает на то, что это операция PUT block. Затем задается BlockID. Параметр Content-MD5 может быть задан для защиты от ошибок передачи по сети и обеспечения целостности. В данном случае, Content-MD5 – это контрольная сумма MD5 данных блока в запросе. Контрольная сумма проверяется на сервере, в случае несовпадения возвращается ошибка. Параметр Content-Length (Длина содержимого) определяет размер содержимого блока. Также в заголовке HTTP-запроса имеется заголовок авторизации, как показано ниже.

PUT <ref src="http://<account>.blob.core.windows.net/video/

BVideo.mp4?comp=block &blockid=BlockId1 &timeout=60 HTTP/1.1

Content-Length: 4194304Content-MD5" type="url">:

HUXZLQLMuI/KZ5KDcJPcOA==Authorization: SharedKey <account>:

<key>= x-ms-date: Mon, 6 Apr 2009 17:00:25 GMTx-ms-version: 2009-04-14

<block data>

GET Blob

Нижеследующий запрос обеспечит извлечение всего содержимого заданного blob. Если для контейнера, которому принадлежит blob (в данном примере "video"), задана политика совместного использования "Private", для получения blob необходимо пройти аутентификацию. Если задана политика совместного использования "Public-Read", аутентификация не требуется, и заголовок аутентификации в заголовке запроса не нужен.

GET <ref src="http://<account>.blob.core.windows.net/video/

BVideo.mp4HTTP/1.1Authorization" type="url"/>: SharedKey

<account>:<key>х-ms-date: Mon, 6 Apr 2009 17:00:25

GMTx-ms-version: 2009-04-14

**Интерфейс перечисления объектов blob - контейнера**

Windows Azure Blob поддерживает перечисление объектов контейнера бинарных объектов. Интерфейс ListBlobs поддерживает параметры префикса (prefix) и разделителя (delimeter), что делает возможным формирование иерархического перечня объектов. К примеру, имеется Blob-контейнер "Photos" бинарных объектов с именами:

Work|MyFirstDay.jpeg

Work|Boss.jpeg

Work|NewProjectStart.jpeg

Friends|Party\_May.jpeg

Friends|NewYear2011.jpeg

Friends|WeekendApril.jpeg

При этом "|" используется как разделитель, для обозначения тематики фотографий, т.е. для создания иерархии имен. Запрос на получение всех имен "каталогов" будет выглядеть следующим образом:

GET <ref src="http://<account>.blob.windows.net/Photos

?comp=list&delimeter=|" type="url"/>

Ответ на запрос:

<BlobPrefix>Work</BlobPrefix>

<BlobPrefix>Friends</BlobPrefix>

Тег < BlobPrefix > указывает на то, что выведенное значение является префиксом имени бинарного объекта, а не именем целиком. Повторяющиеся префиксы возвращаются только один раз. Список бинарных объектов, содержащихся в одном "каталоге", можно получить сочетанием параметров "prefix" и "delimiter", следующим запросом:

GET http://<account>.blob.windows.net/Photos?comp=list

&prefix=Work/&delimeter=|

Ответ на запрос:

<Blob> Work|MyFirstDay.jpeg</Blob>

<Blob> Work|Boss.jheg </Blob>

<Blob> Work|NewProjectStart.jpeg </Blob>

Тег < Blob > указывает на то, что возвращенное имя является полным именем бинарного объекта.

Также интерфейс ListBlobs поддерживает возможность возвращения определенного количества записей, при помощи параметра "maxresult".

Возвращаясь к нашему примеру, следующий запрос вернет первую набор данных, при установленном параметре maxresult=2:

GET <ref src="http://<account>.blob.windows.net/Photos

?comp=list&prefix=Work&maxresult=2" type="url"/>

Ответ на запрос:

<Blob> Work|MyFirstDay.jpeg</Blob>

<Blob> Work|Boss.jheg </Blob>

<NextMarker> Marker1 </NextMarker>

В теге <NextMarker> возвращается непрозрачный маркер, который может быть использован для возвращения следующих результатов:

GET <ref src="http://<account>.blob.windows.net/Photos

?comp=list&prefix=Work&maxresult=2&marker= "

type="url"/>

Marker1

Ответ на запрос:

<Blob> Work|NewProjectStart.jpeg </Blob>

<NextMarker> </NextMarker>

Если тег <NextMarker> пуст, то получены все результаты.

**Список материалов для самостоятельного изучения**

**Блоки и списки**

1. <http://blog.richard.parker.name/2010/06/30/an-introduction-to-windows-azure-for-busy-people/#BlocksAndPages>
2. <http://msdn.microsoft.com/en-us/library/ee691964.aspx>

**Windows Azure Blob. Операции с Blob - объектами**.

1. <http://msdn.microsoft.com/en-us/library/dd573356.aspx>
2. <http://msdn.microsoft.com/ru-ru/library/ee872420.aspx>
3. <http://docwiki.embarcadero.com/RADStudio/en/Windows_Azure_Blob_API>

**Тема 15. Windows Azure Queue. Введение.**

**Модель данных. REST - интерфейс**

**Общее представлениe**

Windows Azure Queue предоставляет простой и надежные асинхронный механизм доставки сообщений. Это позволяет использовать Queue сервис в качестве инструмента интеграции между различных компонент "облачного" решения с приложениями локальной архитектуры. Azure Queue предоставляет REST - интерфейсы, позволяющие создавать приложения на различных языках программирования, обеспечивая расширяемость, интеграцию и масштабируемость создаваемых решений.

Сервисно - ориентированное приложение, построенное с использованием Azure Queue будет обладать рядом преимуществ:

1. Возможность оценки масштабируемости на основе анализа длины очереди. Длина очереди отражает, по сути, время задержки обработки. Если размер очереди мал, это означает что арендуется больше ресурсов, чем, возможно, необходимо в данном случае. Напротив, большой размер очереди свидетельствует о явной нехватке вычислительных ресурсов и наличии издержек, связанных с задержками в обработке данных.
2. Реализация разделения ролей. Каждая часть приложения может быть реализована на основе более подходящей технологии, без оглядки на технологии остальных компонент, поскольку сообщения в очереди могут быть как в стандартном, так и в XML - формате.
3. Буферизация запросов. В моменты пиковой нагрузи, когда рабочие роли не справляются со всем количеством, поступающих к ним обращений, "лишние" обращения не отсеиваются а остаются в очереди. Кроме того уменьшается влияние сбоя отдельных компонент на систему в целом.

**Модель данных**

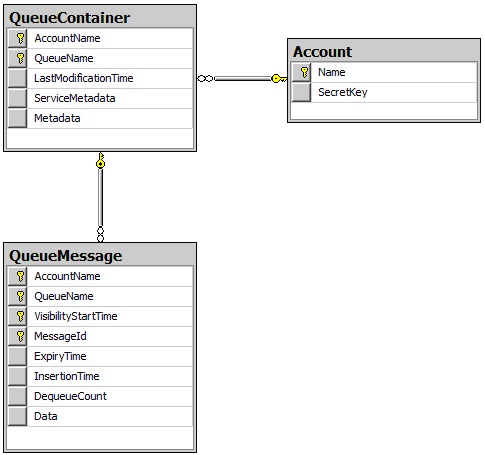


Рисунок 15.1 - Структура данных Windows Azure Queue,

полученная средствами MS SQL Server 2008

Рассмотрим модель данных Windows Azure Queue более детально. Выделим основные термины:

* Учетная запись. Любой доступ к Windows Azure Storage и его сервисам, осуществляется посредством учетной записи. При этом одна учетная запись может иметь несколько очередей.
* Очередь. Очередь содержит множество сообщений, при этом:
  + количество сообщений не ограничено
  + сообщение хранится не дольше одной недели и удаляется по истечении этого срока в процессе "сборка мусора"
  + с очередями могут быть ассоциированы метаданные вида " имя - значение" размером до 8Кб
* Сообщения. Хранятся в очередях. Размер каждого сообщения ограничен 8Кб. При необходимости хранения большего объема данных, сами данные размещаются в табличных или бинарных хранилищах, а сообщение хранит имя бинарного объекта, или сущности.

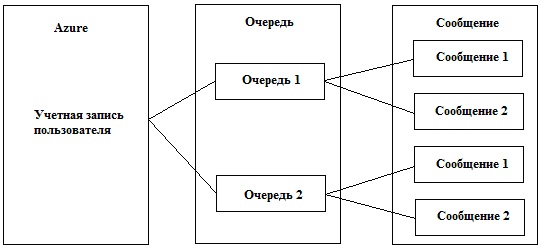


Рисунок 15.2 - Архитектура Queue сервиса Windows Azure

Параметры AzureQueue Services:

MessageID - идентификатор сообщения в очереди

VisibilityTimeout - значение времени ожидания видимости сообщения, т.е. через какой промежуток времени созданное сообщение " увидят". До 2 часов, значение по умолчанию - 30 секунд

PopReceipt -возвращаемая строка для каждого сообщения, наряду с MessageID необходима для удаления строки из очереди

MessageTTL - срок жизни сообщения в сеундах.

**REST - интерфейс**

Любой доступ к Windows Azure Queue осуществляется через HTTP - интерфейс REST.

К операциям HTTP\REST на уровне очередей и сообщений относятся:

|  |  |  |
| --- | --- | --- |
| Таблица 15.1 - Queue - операции | | |
| **Операции** | **HTTP метод** | **Описание** |
| List Queues | GET | Выводит список очередей учетной записи |
| Create Queue | PUT | Создает новую очередь в рамках текущей учетной записи |
| Delete queue | DELETE | Удаляет очередь |
| Get Queue Metadata | GET/HEAD | Возвращает свойства очереди, включая определенные пользователем метаданные |
| Set Queue Metadata | PUT | Задает определенные пользователем метаданные очереди |
| Put Message | POST | Добавляет сообщение в очередь |
| Get Messages | GET | Извлекает сообщение из очереди и делает его невидимым для остальных клиентов |
| Peek Messages | GET | Извлекает сообщение из начала очереди без изменения visibility сообщения |
| Delete Message | DELETE | Удаляет определенное сообщение из очереди |
| Clear Messages | DELETE | Удалить все сообщения из очереди |

Операции с очередями осуществляются при помощи следующего URL: http://<account>. queue .core.windows.net/<QueueName>

Операции с сообщениями осуществляются при помощи следующего URL: http://<account>. queue.core.windows.net/<QueueName>/messages

где <account> - имя учетной записи, а <QueueName> - имя очереди.

**Список дополнительных материалов для самостоятельного изучения**

**Windows Azure Queue**

1. <http://msdn.microsoft.com/ru-ru/library/ee872424.aspx>

**Windows Azure Worker Role и Windows Azure Queue Service**

1. <http://social.technet.microsoft.com/wiki/contents/articles/windows-azure-and-sql-azure-tutorials-tutorial-4-using-windows-azure-worker-role-and-windows-azure-queue-service.aspx>

**REST - операции**

1. <http://msdn.microsoft.com/en-us/library/dd179363.aspx>

**Обмен сообщениями**

1. <http://www.oszone.net/13891/Windows-Azure-Storage>

**Обзор Windows Azure Table и Windows Azure Queue (видео)**

1. <http://www.microsoftpdc.com/2009/SVC09>

**Тема 16. Windows Azure Queue: примеры использования, REST - запросы**

**Примеры использования**

Рассмотрим условный пример, демонстрирующий логику использования Azure Queue в приложении [(рис. 16.1)](http://www.intuit.ru/studies/courses/677/533/lecture/6930?page=1#image.22.1):

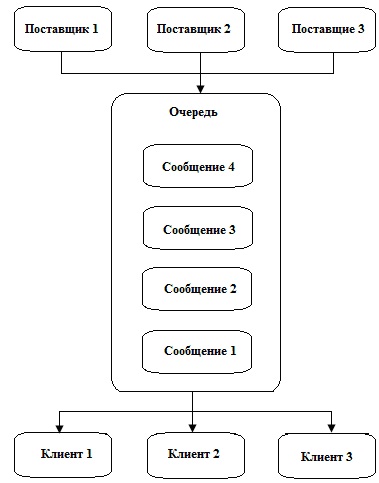


Рисунок 16.1 - Логика использования Azure Queue

Поставщики и клиенты обмениваются информацией посредством очереди сообщений. Поставщики размещают сообщения в очереди, клиенты - их обрабатывают.

1. Клиент 1 забирает из очереди сообщение. Ему возвращается Сообщение 1, при этом оно становится невидимым в очереди на величину времени равную значению VisibilityTimeout.
2. Клиент 2 забирает сообщение из очереди, при этом Сообщение 1 все еще невидимо. Таким образов Клиенту вернется Сообщение 2.
3. По завершении обработки сообщения Клиент 2 удаляет его из очередь.

Отметим, что в случае, если Клиент 2 удалит Сообщение 2, а Клиент 1 не сможет удалить Сообщение 1, то последующие запросы сообщений из очереди вернут Сообщение 1. Таким образом, любое сообщение в очереди будет гарантировано обработано хотя бы единожды.

**Примеры REST - запросов**

Рассмотрим некоторые примеры использования REST - запросов Windows Azure Queue. Учетная запись пользователя будет обозначена, как <account>, имя очереди - <AzureQueue>, само сообщение - <message>

Постановка в очередь

Следующий пример REST - запроса добавит сообщение в очередь, при помощи HTTP - метода PUT. Поясним некоторые моменты: параметр "messagettl" является необязательным и задает время жизни сообщения в секундах до семи дней, что является также сроком жизни по - умолчанию. Параметр Content-MD5 может быть задан для защиты от ошибок передачи по сети и обеспечения целостности. Параметр Content-Length (Длина содержимого) определяет размер содержимого сообщения.

PUT <ref src="http://<account>.queue.core.windows.net/

<AzureQueue>/messages? messagettl=3600HTTP/1.1"

type="url"/> Content-Length: 3900Content-MD5:

HUXZLQLMuI/KZ5KDcJPcOA== Authorization: SharedKey <account>:<key>

x-ms-date: Thu, 10 May 14:05:56 GMT <mesage>

Извлечение сообщения

Следующий пример REST - запроса извлечет сообщение из очереди, при помощи HTTP - метода GET. Поясним некоторые моменты: "numofmessages" - число сообщение, которое должно быть извлечено (до 32х, по умолчанию только 1); " visibilitytimeout" - определяет время в секундах, которое сообщение будет оставаться невидимым, после его извлечения (от 30секунд до 2 часов, по умолчанию - 30 секунд).

GET http://<account>.queue.core.windows.net/<AzureQueue>

/messages?numofmessages=1 &visibilitytimeout=100HTTP/1.1Authorization:

SharedKey <account>:<key>x-ms-date: Thu, 10 May 14:05:56 GMT

Ответ на запрос будет получен в виде:

HTTP/1.1 200 OK

Transfer-Encoding: chunked

Content-Type: application/xml

Server: Queue Service Version 1.0 Microsoft-HTTPAPI/2.0

x-ms-request-id: 2<requestID>

Date: Thu, 10 May 14:05:56 GMT

<?xml version="1.0" encoding="utf-8"?>

<QueueMessagesList>

<QueueMessage>

<MessageId>MessageID</MessageId>

<InsertionTime>...</InsertionTime>

<ExpirationTime>...</ExpirationTime>

<PopReceipt> PRValue</PopReceipt>

<TimeNextVisible>...</TimeNextVisible>

<MessageText>MessageText</MessageText>

</QueueMessage>

</QueueMessagesList>

Следует пояснить тег <PopReceipt> его значение идентифицирует полученное сообщение, в частности это значение может быть использовано для удаления сообщения.

Удаление сообщения

Следующий пример REST - запроса извлечет сообщение из очереди, при помощи HTTP - метода DELETE. Параметр " popreceipt" определяет сообщение, которое должно быть удалено, его значение получено при помощи запроса на извлечение сообщения.

DELETE /<account>/<AzureQueue>/messages/MessageID?popreceipt=PRValue&timeout=30HTTP/1.1

Content-Type: binary/octet-streamx-ms-date: Thu, 10 May 15:02:04 GMT Authorization: SharedKey <account>:<key>

**Особенности обмена сообщениями Windows Azure Queuе**

Подведем краткий итог и обратим внимание на основные особенности очередей и процесса обмена сообщениями Azure Queue.

При работе с " облаком", необходимо понимать, что физически между различными частями данных вашего приложения могут быть многие десятки и сотни километров. Это касается не только данных, но и рабочих ролей, обрабатывающих эти данные, одна из которых может располагаться на Дальнем Востоке, другая - в Москове. И адаптация к данной особенности - одна из основных проблем разработчиков.

Касательно очередей Windows Azure. На самом деле, все выглядит довольно просто: приложение запрашивает сообщения из очереди и указывает период ожидания, определяющий какое количество времени эти сообщения не будут доступны для остальных клиентов. В случае успешной обработки сообщения, приложение удаляет его из очереди. В случае сбоя, или генерации исключения сообщение возвращается в очередь для последующей обработки и становится видимым для остальных клиентов.

Важно проектировать систему таким образом, чтобы одно сообщение из очереди могло быть обработано несколько раз без рассогласования состояния приложения, даже в случае, если сообщение обрабатывается разными рабочими ролями.

**Список материалов для самостоятельного изучения**

**Windows Azure Queue**

1. <http://msdn.microsoft.com/ru-ru/library/ee872424.aspx>
2. http://www.wintellect.com/CS/blogs/pmehner/archive/2010/02/28/idempotency-for-windows-azure-message-queues.aspx

**Архитектура Windows Azure Queue**

1. <http://mscerts.net/windows/windows%20azure%20%20%20queue%20service%20architecture.aspx>

**Работа с очередями**

1. <http://blogs.infinite-x.net/2010/01/12/working-with-windows-azure-queues/>

**Удаление сообщений**

1. <http://blog.smarx.com/posts/deleting-windows-azure-queue-messages-handling-exceptions>

**Тема 17. Microsoft .Net Access Services.**

**Идентификация на основе утверждений**

Как уже упоминалось в предыдущих Тема, .Net Access Control Services обеспечивает управление доступом к приложениям и сервисам и интеграцию с имеющимися у заказчика средствами авторизации. Поддерживаются стандартные механизмы аутентификации (к примеру Windows Live ID, Active Directory). Основой сервиса Access Control является Windows Identity Foundation.

С .Net Access Control Services "облачные" и локальные приложения могут объединяться (в т.н. федерации) и позволяет использовать сервисы через firewall. Azure - приложение использует ту же систему безопасности каталогов, что и Active Directory, т.е. приложение будет считать, что учетная запись пользователя управляется локально.

Мы не будем подробно затрагивать вопрос об основах идентификации и авторизации и их важности. Отметим только парадокс: несмотря на то, что задача обеспечения безопасности никогда не теряет актуальности и, по сути, любое приложение начинает работу с идентификации пользователя и определения границ его действий, редко данная задача рассматривается подробно в учебных курсах.

Основной целью данной лекции является знакомство с идентификацией на основе утверждений.

**Идентификация на основе утверждения**

Особенностью идентификации на основе утверждений является централизованная реализация сервисов аутентификации и авторизации вне локальной инфраструктуры, вне организации. Собственно, именно эти сервисы и предоставляет .Net Access Control.

При реализации данного вида идентификации, пользователь передает приложению набор утверждений, которыми могут являться имя, должность, e-mail и т.д. Сами утверждения предоставляются службой сертификации, аутентифицирующей пользователя. Таким образом, приложение получает все необходимые данные, при этом клиентское приложение (к примеру, браузер), взаимодействуя со службой сертификации, подтверждает передаваемые утверждения.

Для обеспечения заявленного, приложения используют SAML (Security Assertion Markup Language). Утверждения передаются SAML - маркерами, каждый маркер несет часть информации о пользователе. Маркеры генерируются STS - программой.

STS (security token service), или сервис маркеров доступа - создает, подписывает и выдает маркеры доступа. STS принимает запросы на получение маркера (RST - request for token) и возвращает ответ (RSTR).

По сути, STS является элементом службы сертификации.

Однако, не все так просто, как кажется на первый взгляд. SAML маркер может не содержать утверждений, ожидаемых приложением и сервис, генерирующий маркером может не быть доверенным, с точки зрения приложения. Для решения этой проблемы в процесс идентификации вовлекается еще один STS - сервис, гарантирующий, что SAML - маркеры будут содержать всю требуемую информацию. При этом, .Net Access Control Services, использую федеративные механизмы, устанавливает доверенную связь между новым STS и сервисом, генерирующим маркеры.

[Рис. 17.1](http://www.intuit.ru/studies/courses/677/533/lecture/6932?page=1#image.24.1) показывает, как .Net Access Control Services обеспечивает преобразование утверждений и федеративную идентификацию.

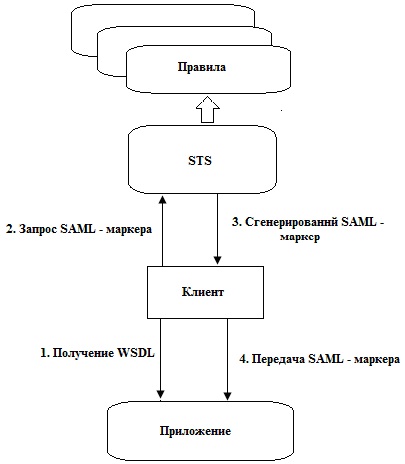


Рисунок 17.1 - .Net Access Control Services

1. На первом шаге клиент знакомится с политикой приложения, т.е. куда необходимо обращаться для аутентификации.
2. Генерируется запрос на получения SAML - маркера к STS службам, формирующим маркеры на основе ряда правил.
3. Маркер передается клиенту.
4. Маркер предается приложению. Приложение принимает только маркеры подписанные службой сертификации, все иные маркеры отклоняются.

[На рисунке 17.1](http://www.intuit.ru/studies/courses/677/533/lecture/6932?page=1#image.24.1) на стороне пользователя подразумевается наличие Smart - клиента, генерирующего запросы к STS и взаимодействующие с приложением. В случае использования пользователем браузера, при обращении к веб - приложению, использующим идентификацию на основе утверждений, шаги процесса идентификации будут следующими:

1. Пользователь посылает HTTP - запрос веб - приложению.
2. Приложение, поскольку пользователь не зарегистрирован, перенаправляет его на веб - страницу, предоставляемую доверенной службой сертификации.
3. Служба сертификации управляет процессом аутентификации пользователя.
4. После аутентификации службой сертификации определяются необходимые утверждения и формируется SAML - маркер.
5. SAML - маркер включается в состав ответа с java - сценарием.
6. Сценарий, вернувшись браузеру пользователя, передает маркер веб - приложению через POST - запрос.

**Используемые стандарты**

Возможность взаимодействия всех этих элементов обеспечивается несколькими WS -стандартами (см п.№8 списка материалов для самостоятельного изучения).

WSDL извлекается с помощью WS-MetadataExchange или простого HTTP-запроса GET, и политика в WSDL структурирована соответственно спецификации WS-Policy.

STS службы сертификации предоставляют конечную точку, реализующую спецификацию WS-Trust, которая описывает процессы запроса и получения маркеров доступа.

SAML - словарь XML, который может использоваться для представления утверждений в форме, обеспечивающей возможность взаимодействия.

**Список материалов для самостоятельного изучения**

**Идентификация и авторизация пользователей**

1. <http://articles.security-bridge.com/articles/91/11658>/
2. <http://www.intuit.ru/department/security/secbasics/10/>

**.Net Access Control Services**

1. <http://www.techbubbles.com/microsoft/net-access-control-service/>
2. <http://msdn.microsoft.com/ru-ru/library/ee872415.aspx>

**Идентификация на основе утверждений**

1. http://msdn.microsoft.com/ru-ru/library/ee539091.aspx

**SAML**

1. <http://www.ccc.ru/magazine/depot/04_02/read.html?0501.htm>
2. <http://www.pingidentity.com/knowledge-center/SAML-Tutorials-and-Resources.cfm>

**WS - стандарты**

1. <http://www.ccc.ru/magazine/depot/06_10/read.html?0202.htm>
2. **WSDL**
3. <http://xmlhack.ru/texts/03/wsdl.tales/wsdlintralook1.html>
4. <http://www.w3.org/TR/wsdl>

**Тема 18. Отладка и развертывание проекта**

**Развертывание "облачной" службы**

"Облачную" службу можно развернуть непосредственно из среды разработки. В нашем случае, щелкнув правой кнопкой мыши на проекте "облачной" службы, и выбрав пункт "Опубликовать."

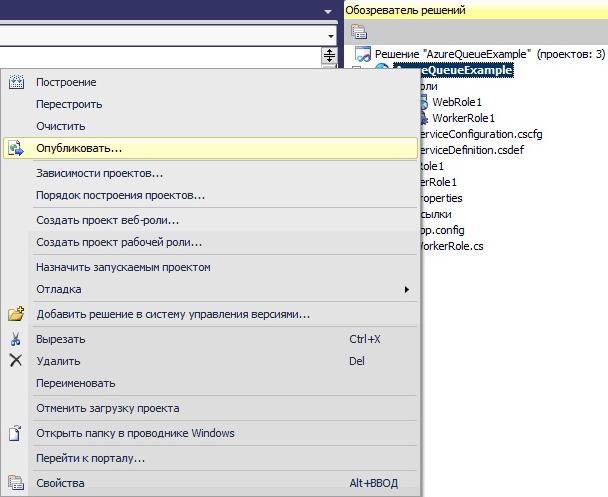


Рисунок 18.1 - Развертывание проекта Windows Azure

В появившемся окне "Развертывание проекта Windows Azure" модно выбрать один из следующих вариантов:

* Только создать пакет службы. Можно указать Visual Studio только создать пакет службы. Когда пакет создан, Visual Studio открывает окно проводника, показывая расположение файла созданного пакета. Теперь вы можете перейти на портал разработчика и развернуть пакет и файл конфигурации в нужный слот развертывания.
* Развернуть "облачную" службу в Windows Azure. В этом случае служба развертывается непосредственно в Windows Azure, для этого необходимо указать идентификатор подписки и сертификат, для проверки подлинности учетных данных

Независимо от выбранного варианта, перед тем, как выполнять развертывание "облачной" службы, необходимо создать на портале разработчика Windows Azure следующее.

* Подписка на Windows Azure. При регистрации в Windows Azure подписка связывается с вашим Live ID. Идентификатор подписки можно найти, перейдя на страницу Account портала разработки. Идентификатор подписки отображается в разделе Support Information в нижней части страницы.
* Размещенная служба Windows Azure. До развертывания "облачной" службы необходимо создать размещенную службу для этого развертывания. Размещенная служба предоставляет два слота развертывания, в которые может быть развернута "облачная" служба: Промежуточный и Производственный.
* Учетная запись хранилища Windows Azure. При развертывании "облачной" службы из Visual Studio пакет службы сначала отправляются в хранилище больших двоичных объектов через заданную учетную запись хранилища, а затем разворачиваются в среде Windows Azure из службы больших двоичных объектов.

При первом развертывании "облачной" службы из Visual Studio необходимо связать ее с размещенной службой, созданной вами на портале разработчика; также необходимо предоставить учетные данные, которые Visual Studio сможет использовать для взаимодействия с вашей подпиской Windows Azure.

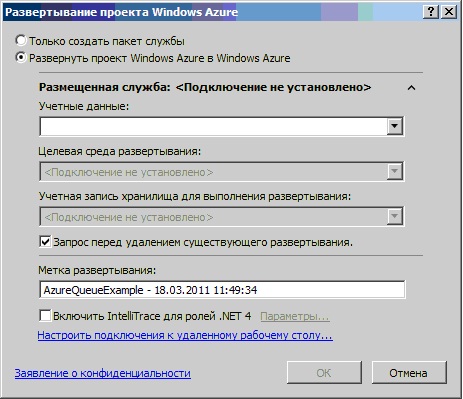


Рисунок 18.2 - Развертывание облачной службы

Также в окне "Развертывание облачной службы" можно включить IntelliTrace.

**Просмотр состояния развертывания**

Состояние развертывания также можно просматривать средствами среды разработки в обозревателе серверов, где можно добавить развертывания для отслеживания. При просмотре состояния развертывания работу в Visual Studio прерывать не требуется.

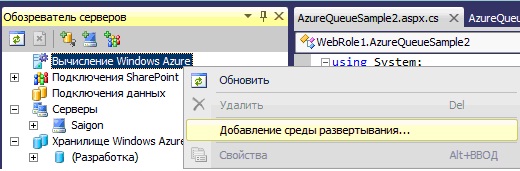


Рисунок 18.3 - Обозреватель серверов

Обозреватель вычислений Windows Azure позволяет просматривать и отслеживать только ваши собственные развертывания; из обозревателя вычислений Windows Azure нельзя запускать и останавливать экземпляры ролей.

Просматривать службу в обозревателе вычислений Windows Azure можно только после того, как она развернута в слот развертывания.

Чтобы добавить развертывание в обозреватель вычислений Windows Azure:

* Откройте обозреватель серверов.
* Щелкните правой кнопкой мыши в обозревателе вычислений Windows Azure и выберите "Добавление среды развертывания", для отображения окна "Добавление среды развертывания".

Чтобы добавить слот развертывания для отслеживания щелкните правой кнопкой мыши узел "Учетные данные Windows Azure " и выберите "Создать".

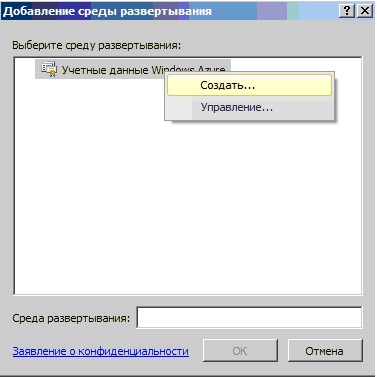


Рисунок 18.4 - Добавление среды развертывания

Эти учетные данные используются Visual Studio для взаимодействия с Windows Azure для управления вашими размещенными службами.

После создания и сохранения ваших именованных учетных данных выберите размещенные службы и слот развертывания для отображения в обозревателе вычислений Windows Azure. Обозреватель вычислений Windows Azure будет показывать отслеживаемый слот, включен ли отладчик IntelliTrace для этого развертывания и состояние каждого экземпляра роли.

**Отладка с помощью IntelliTrace**

Отладка с помощью IntelliTrace доступная в Microsoft Visual Studio 2010 Ultimate, и позволяет получить более подробное представление о приложении.

С помощью IntelliTrace можно увидеть события, произошедшие в прошлом, а также контекст, в котором они происходили.

Отладку с помощью IntelliTrace можно включить только для "облачной" службы, развертывание которой выполняется из Visual Studio. Необходимо настроить отладку с помощью IntelliTrace для "облачной" службы перед ее развертыванием в среду Windows Azure. Если отладка для службы не настроена, и вы решили сделать это, необходимо заново выполнить развертывание службы через Visual Studio.

Когда все готово к развертыванию "облачной" службы, убедитесь, что для целевых объектов построения задано значение Отладка. Затем щелкните правой кнопкой мыши по проекту "облачной" службы в обозревателе решений и выберите "Опубликовать". Для включения IntelliTrace установите флажок "Включить IntelliTrace для ролей .NET 4" в диалоговом окне "Развертывание проекта Windows Azure ".

Журнал IntelliTrace представляет собой кольцевой файл журнала, максимальный размер которого указывается в настройках IntelliTrace (размер по умолчанию 250 МБ). Журналы IntelliTrace записываются в файл в файловой системе виртуальной машины. В момент, когда вы запрашиваете журнал, делается снимок, который загружается на ваш локальный компьютер.

Загрузить журналы IntelliTrace для экземпляра роли можно с помощью обозревателя вычислений Windows Azure.

**Список вспомогательных материалов и материалов для самостоятельного изучения**

**Развертывание облачной службы**.

1. <http://msdn.microsoft.com/ru-ru/library/ff683671.aspx>
2. <http://msdn.microsoft.com/ru-ru/library/dd264915.aspx>
3. <http://blogs.msdn.com/b/jnak/archive/2010/05/27/using-intellitrace-to-debug-windows-azure-cloud-services.aspx>
4. <http://msdn.microsoft.com/ru-ru/library/ff683672.aspx>

**IntelliTrace - общая информация**

1. <http://www.interface.ru/home.asp?artId=23551>
2. <http://msdn.microsoft.com/en-us/library/dd264915.aspx>
3. <http://habrahabr.ru/blogs/vs/73335/>
4. <http://msug.vn.ua/Posts/Details/3809>
5. <http://andir-notes.blogspot.com/2010/02/intellitrace-new-tool-for-debugging-in.html>

**Отладка с помощью IntelliTrace**

1. <http://msdn.microsoft.com/ru-ru/library/dd572114.aspx>