**Лекция 6:**

**Windows Azure SDK**

*Цель данной лекции – ознакомиться с комплектом средств разработки* *Windows Azure* *SDK.*

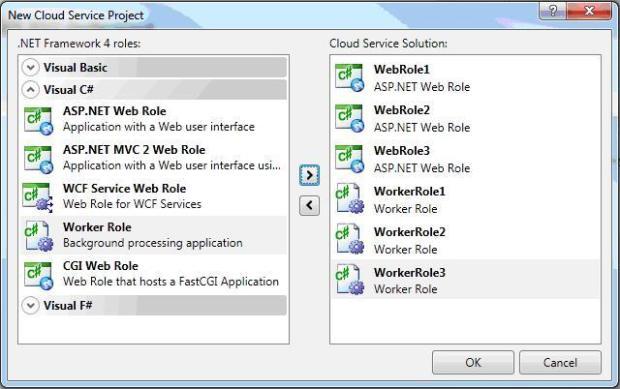
*Windows Azure SDK* предоставляет разработчикам интерфейс программирования приложений, необходимый для разработки, развертывания и управления масштабируемых сервисов в Windows Azure.

Azure Cloud Fabric и службы Azure Storage не поддерживают разработку или отладочные операции в облаке, поэтому Azure SDK позволяет делать это локально в виде приложений Development Fabric (DF) и Development Storage (DS), которые устанавливает Windows Azure SDK. Вместе с SDK также устанавливаются коллекция приложений примеров и библиотеки упакованных классов для облегчения программирования приложений.

Должны быть установлены .NET Framework 3.5 SP1 и SQL Express 2005 или 2008, также необходимо включить ASP.NET и WCF HTTP Activation для IIS 7.0 для Windows Server 2008, Windows Vista SP2, или Windows 7 RC или позже для установки и запуска SDK. Заметки к выпуску включают инструкции для настройки этих опций. Использование SDK не является обязательным, потому что есть возможность пользоваться любыми операционными системами и языками программирования, которые поддерживают HTTP запросы и ответы. Однако, вы увидите что использование SDK интерфейсов прикладного программирования .NET и библиотек для приложений и хранилищ позволяет наиболее просто работать с HTTP напрямую.

После того, как вы установили Azure SDK, Вы должны скачать и установить инструменты Windows Azure для Visual Studio для добавления шаблонов проектов *Web Cloud Sevice, Worker Cloud Service, Web and Worker Cloud Service Workflow Service*. Вы можете скачать текущую версию Windows Azure SDK и Windows Azure Tools для Visual Studio с главной страницы Windows Azure по ссылке *www.microsoft.com/azure/windowsazure.mspx*.

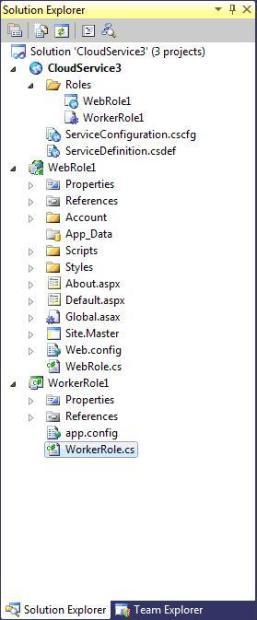
После установки Windows Azure Tools в Visual Studio появляются шаблоны *Cloud Service* в диалоге создания нового проекта. При выборе узла Cloud Service открываются New Cloud Service, который позволяет добавить *ASP.NET Web Roles, Worker Roles или CGI Web Roles* для нового проекта. Windows Azure SDK позволяет добавить более чем одну роль для каждого типа Cloud Service. Каждая роль использует отдельный экземпляр Windows Azure CPU, так минимальная стоимость запуска проекта в облаке будет примерно 4 \* $0.12 = $0.48 в час.



**Рис. 5.1.** Создание нового проекта Cloud Service в Visual Studio

Добавив указанные роли и нажав ОК, откроется новое решение с проектами *WebRole* и *Worker-Role* в Solution Explorer как показано на [рис. 5.2](http://www.intuit.ru/studies/courses/673/529/lecture/11920?page=1#image.5.2).

Узел *Roles* содержит элементы *WebRole*, которые указывают на каждую *WebRole*, которая обеспечивает пользовательский интерфейс ASP.NET для приложения и каждый *WorkerRole* для вычислительный операций, которые не требуют пользовательского интерфейса или используют страниц *ASP.NET WebRole* вместо этого.



**Рис. 5.2.** Solution Explorer Cloud Service

В зависимости от типа Cloud Service проекты включают пространство имен *Microsoft.ServiceHosting.ServiceRuntime*, которое содержит классы, указанные в таблице ниже.

|  |  |
| --- | --- |
| **Класс** | **Описание** |
| RoleEntryPoint | Обеспечивает методы для управления инициализацией, запуском и остановкой методов сервиса, так же используется для мониторинга состояния сервиса. |
| RoleException | Сообщает об ошибках когда происходят недопустимые операции внутри роли |
| RoleManager | Обеспечивает методы для журналирования сообщений и поступающих предупреждений, извлекает настройки конфигурации сервиса и возвращает местоположение ресурса |
| RoleStatus | Информирует о текущем статусе роли: Healthy, NonExistent, Started, Starting, Stopped, Stopping или Unhealthy |

Проекты, которые используют шаблон *WebRole* определяют веб страницу ASP.NET Default.aspx как начальную точку для пользовательского интерфейса приложения в облаке.

Это сервис объединяет библиотеку класса *Common* из приложения-образца *HelloFabric* для содействия в журналировании проблем приложения. Журналы приложения – это практичные средства откладки приложений, запущенных в *Cloud Fabric*. Для чтения журналов, вы должны скопировать их в Blob массив использую инструментарий портала.

Образец проекта *StorageClient* включает библиотеку класса *StorageClient*, которая обеспечивает в объединении с библиотекой .NET Client для сервиса данных ADO.NET, интерфейсный класс Microsoft .NET для HTTP операций над *Azure Blob*, *Queue* и *Table Storage* сервисами. Этот проект также включает консольное приложение, которые позволяет Вам тестировать возможности библиотеки. Консольное приложение C# запускается в *Development Fabric с Development Storage*.

При установке Windows Azure SDK не устанавливаются образцы приложений, которые включены в *Program Files\Microsoft Windows Azure SDK\v1.0\samples.zip*. Установите образцы, разархивировав samples.zip в директорию, где Вы имеете права на запись. В таблице ниже можно найти описание некоторых образцов приложений.

Для запуска примера *CloudDrive* необходим PowerShell.

Директория, в которую было извлечено содержимое архива samples.zip также содержит следующие три пакетных файла (cmd), которые можно запустить из командной строки:

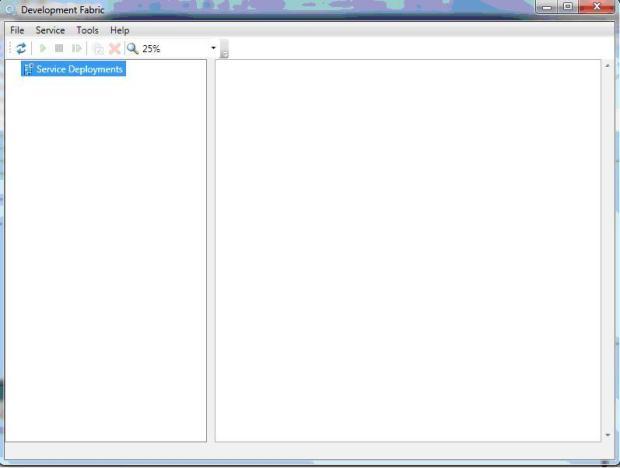
* *buildall.cmd* строит все образцы проектов без использования Visual Studio:
* *createtables.cmd* вызывает *buildall.cmd* и создает базу данных и таблицы, необходимые для образцов, которые используют Table Storage.
* *rundevstore.cmd* вызывает *createtables.cmd* и запускает разработку хранилища, размещая его в базе данных, созданной *createtables.cmd*.

В состав Development Fabric входят следующие исполняемые файлы: *DFAgent.exe , DFLoadBalancer.exe, DFMonitor.exe* и *DFService.exe*, которые по умолчанию устанавливаются программой установки Azure SDK в каталог \Program Files\Windows Azure SDK\v1.0\bin\devfabric. После запуска Development Fabric в диспетчере задач вы можете увидеть эти четыре процесса. Сделать это можно выполнив:

* Выберите *Программы\Windows Azure SDK\Development Fabric* для запуска службы Development Fabric и его пользовательского интерфейса *DFUI.exe*
* Правый щелчок мыши по значку Development Fabric в области уведомлений панели задач и выбрать запуск службы Development Fabric ( [рис. 5.3](http://www.intuit.ru/studies/courses/673/529/lecture/11920?page=1#image.5.3))
* Скомпилировать и запустить приложение Azure в Visual Studio.



**Рис. 5.3.** Сообщения, отображаемые нажатии правой кнопкой мыши по значку Development Fabric в области уведомлений панели задач.



[рис. 5.4](http://www.intuit.ru/studies/courses/673/529/lecture/11920?page=2#image.5.4) показывает пользовательский интерфейс DFUI. Когда вы запускаете или останавливаете отладку, соответствующие приложения появляются или пропадают из пользовательского интерфейса DFUI.

**Рис. 5.4.** Пользовательский интерфейс приложения Development Fabric

Платформа Windows Azure поддерживает три типа масштабируемых хранилищ:

* Неструктурированные данные (blob)
* Структурированные данные (таблицы)
* Сообщение между приложениями и сервисами (очереди)

Запуская *rundevstore.exe* или собирая и запуская пользовательский код Azure в Visual Studio, запускаются все три сервиса, даже если Ваш проект требует только один сервис и отображается в пользовательском интерфейсе Development Storage

Для защиты от потери данных, облако Azure хранит блобы, таблицы и очереди в минимум трех раздельных контейнерах в одном центре обработки данных. Инструмент геолокации Azure позволяет дублировать данные в нескольких центрах обработки данных Microsoft для уменьшения последствий восстановления после катастроф и для повышения производительности в специфичных географических регионах.

Приложение Azure, которые Вы запускаете в Development Framework, могут иметь доступ к локальным данным в Development Storage или к данным, загруженным в облако Azure. Приложение обращается к определенному порту и данным, расположенным в определенных местах в файле конфигурации проекта *ServiceConfiguration.cscfg*.

Файл конфигурации проекта Azure *ServiceDefinition.csdef* определяет стандартные точки входа и настройки конфигурации, которые хранятся в файле *ServiceConfiguration.cscfg*. В распечатке 5.1 показано содержимое файла ServiceDefinition.csdef по умолчанию, когда Вы создаете проект Azure, используя один из стандартных шаблонов Windows Azure Tools для Visual Studio (отмечены важные значения).

<ServiceDefinition name="SampleWebCloudService"

xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">

<WebRole name="WebRole">

<InputEndpoints>

<!-- Must use port 80 for http and port 443 for https

when running in the cloud -->

<InputEndpoint name="HttpIn" protocol="http" port="80" />

</InputEndpoints>

<ConfigurationSettings>

<Setting name="AccountName"/>

<Setting name="AccountSharedKey"/>

<Setting name="BlobStorageEndpoint"/>

<Setting name="QueueStorageEndpoint"/>

<Setting name="TableStorageEndpoint"/>

</ConfigurationSettings>

</WebRole>

</ServiceDefinition>

Листинг 5.1. Содержимое файла ServiceDefinition.csdef

Значение *InputEndpoint* применяется только для хранилищ в облаке.

Распечатка 5.2 показывает содержимое файла *ServiceConfiguration.cscfg* для веб приложения *SampleWebCloudService* с конфигураций по умолчанию для *Development Storage* (выделено):

<?xml version="1.0"?>

<ServiceConfiguration serviceName="SampleWebCloudService"

xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration">

<Role name="WebRole">

<Instances count="1"/>

<ConfigurationSettings> <Setting name="AccountName" value="devstoreaccount1"/>

<Setting name="AccountSharedKey" value="Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ

1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw=="/>

<Setting name="BlobStorageEndpoint" value="http://127.0.0.1:10000/"/>

<Setting name="QueueStorageEndpoint" value="http://127.0.0.1:10001/"/>

<Setting name="TableStorageEndpoint" value="http://127.0.0.1:10002/"/>

<!--<Setting name="AccountName" value="oakleaf"/>

<Setting name="AccountSharedKey" value="3elV1ndd . . . Coc0AMQA==" />

<Setting name="BlobStorageEndpoint"

value="http://blob.core.windows.net" />

<Setting name="QueueStorageEndpoint"

value="http://queue.core.windows.net" />

<Setting name="TableStorageEndpoint"

value="http://table.core.windows.net" />

</ConfigurationSettings>

</Role>

</ServiceConfiguration>

Листинг 5.2. Содержимое файла ServiceConfiguration.cscfg

Описания элементов файла конфигурации ServiceConfiguration.csfg:

* Instances count – количество экземпляров вашего приложения, которое будет создано в облаке, когда вы развернете его.
* AccountName– имя, ассоциированное с Вашим Hosted Service, с которым вы создавали учетную запись, для Development Storage это devstoreaccount1.
* AccountSharedKey шифрует несколько элементов в HTTP запросе.
* BlobStorageEndpoint– это публичный постоянный Universal Resource Identifier (URI). Для Developer Storage это адрес интерфейса компьютера loopback (localhost = 127.0.0.1) с TCP портом по умолчанию 10000.
* QueueStorageEndpoint для хранилища в облаке это публичный постоянный URI. Для Developer Storage это адрес интерфейса компьютера loopback с TCP портом по умолчанию 10001.
* TableStorageEndpoint публичный постоянный Universal Resource Identifier (URI). Для Developer Storage это адрес интерфейса компьютера loopback с TCP портом по умолчанию 10002.

Значения конечных точек в пользовательском интерфейсе Development Storage представлены на рисунке 6-6. Вы можете настроить собственные номера TCP портов если при использовании значений по умолчанию возникает конфликт с текущей конфигурацией.

### Краткие итоги:

В данной лекции мы получили первоначальные сведения о работе с Windows Azure SDK. Рассмотрели процедуру создание Cloud Service, пользовательский интерфейс Development Fabric