

PBC201ILL

Лабораторная работа

*Основы разработки под Windows Azure*

СОДЕРЖАНИЕ

[Обзор 3](#_Toc306005524)

[Упражнение 1: создание вашего первого приложения для Windows Azure 6](#_Toc306005525)

[Задача 1 – создание проекта в Visual Studio 6](#_Toc306005526)

[Задача 2 – Создание модели данных для работы с табличным хранилищем 11](#_Toc306005527)

[Задача 3 – создание веб-роли, позволяющей отображать содержимое гостевой книги и добавлять записи 25](#_Toc306005528)

[Задача 4 – использование очередей для организации фоновой обработки данных 36](#_Toc306005529)

[Проверка 41](#_Toc306005530)

[Упражнение 2: фоновая обработка данных с использованием прикладной роли и очереди 47](#_Toc306005531)

[Задача 1 – создание прикладной роли для фоновой обработки данных 47](#_Toc306005532)

[Проверка 60](#_Toc306005533)

[Упражнение 3: развертывание приложения в Windows Azure 62](#_Toc306005534)

[Задача 1 – создание сервиса хранения данных и вычислительного сервиса 62](#_Toc306005535)

[Задача 2 – развертывание приложения с помощью портала Windows Azure 70](#_Toc306005536)

[Задача 3 – изменение числа экземпляров роли 76](#_Toc306005537)

[Задача 4 – проверка приложения в тестовой среде 78](#_Toc306005538)

[Задача 5 – передача приложения в промышленную среду 79](#_Toc306005539)

[Результат 82](#_Toc306005540)

|  |
| --- |
| **Внимание:**   1. **Для выполнения данной лабораторной работы необходима учетная запись Windows Azure** 2. **Все описанные в данном документе материалы находятся в каталоге C:\WAPTK\Labs\IntroductionToWindowsAzureVS2010** 3. **Пароль для входа в систему: Passw0rd!** |

# Обзор

Размещаемые в Windows Azure сервисы состоят из одной или более веб-ролей (web role) и/или прикладных ролей (worker role). Веб-роль является доступным через конечные точки HTTP и HTTPS веб-приложением ASP.NET и предоставляет пользовательский интерфейс. Прикладные роли, как правило, используются для фоновой обработки данных. Сервисы Windows Azure могут включать любые комбинации ролей любого типа, каждая из которых может существовать в одном или более экземплярах. Экземпляры ролей могут добавляться и удаляться без перезапуска приложения, что позволяет масштабировать сервис по мере необходимости.

Сервисы хранения данных Windows Azure (Windows Azure storage services) включают в себя хранилище двоичных объектов (Blob services), табличное хранилище (Table services) и очереди (Queue services), позволяющие организовывать взаимодействие между службами.

* 1. В данной лабораторной работе вы создадите простое приложение GuestBook, в котором демонстрируется ряд возможностей платформы Windows Azure, в том числе использование веб- и прикладных ролей, хранилища двоичных объектов и табличного, а также очередей.
  2. В приложении GuestBook веб-роль используется для создания пользовательского интерфейса, предоставляющего возможность как просмотра содержимого гостевой книги, так и добавления в нее новых записей. Каждая запись включает имя, текстовое сообщение и изображение. В приложении также используется прикладная роль, генерирующая миниатюры для добавленных пользователями изображений.
  3. Когда пользователь добавляет новую запись, веб-роль загружает соответствующее изображение в хранилище двоичных объектов, после чего добавляет в табличное хранилище новую сущность, содержащую введенную пользователем информацию и ссылку на изображение в хранилище двоичных объектов. При обращении веб-роль форматирует данную информацию таким образом, чтобы пользователь мог просмотреть содержимое гостевой книги.
  4. После сохранения изображения и добавления сущности в табличное хранилище веб-роль помещает в очередь рабочий элемент, указывающий на необходимость обработки изображения. Прикладная роль извлекает рабочий элемент из очереди, извлекает изображение из хранилища двоичных объектов и создает миниатюру – уменьшенный вариант оригинального изображения. Использование очередей является рекомендованным подходом при организации взаимодействия сервисов в облаке. Преимущества организации слабосвязанных элементов заключаются в возможности их раздельного тестирования и масштабирования.

Цели

* 1. В данной лабораторной работе вы научитесь:
  + Создавать приложения для Windows Azure на основе веб-ролей и прикладных ролей
  + Использовать сервисы хранения данных Windows Azure (Windows Azure storage services) – двоичное и табличное хранилища и очереди
  + Выполнять развертывание сервисов в Windows Azure

Технические требования

* 1. Для выполнения данной лабораторной работы потребуется наличие следующих инструментов и компонентов:
  + IIS 7 (в том числе поддержка ASP.NET, WCF HTTP Activation)
  + [Microsoft Visual Studio 2010](http://msdn.microsoft.com/vstudio/products/)
  + Microsoft .NET Framework 4.0
  + [Windows Azure Tools for Microsoft Visual Studio 1.4](http://go.microsoft.com/fwlink/?LinkID=128752)
  + [SQL Server 2005 Express Edition (или более поздняя редакция)](http://www.microsoft.com/express/sql/download/)

Настройка системы

* 1. **[Данная процедура была выполнена для вас]**
  2. Для вашего удобства большая часть кода представлена в виде фрагментов (code snippets), зарегистрированных в Visual Studio. Чтобы проверить наличие требуемых для выполнения работы компонентов и установить фрагменты кода:
  3. Откройте Проводник Windows (Windows Explorer) и перейдите в каталог **Source\Setup**.
  4. Дважды щелкните на файле **Dependencies.dep** для запуска утилиты Dependency Checker, используя которую, установите фрагменты кода и недостающие компоненты.
  5. В случае появления диалога User Account Control, подтвердите необходимость продолжения выполнения.
     1. **Внимание:** данная операция требует повышения полномочий. Расширение .dep используется утилитой Dependency Checker. Для получения дополнительной информации об использовании утилиты обратитесь к документу **Setup.docx**, находящемуся в подкаталоге **Assets** в каталоге с материалами к данной работе.

Использование фрагментов кода (Code Snippets)

* 1. В течение выполнения всей лабораторной работы вы будете добавлять в проект фрагменты кода. Для вашего удобства большая часть таких фрагментов оформлена в виде зарегистрированных в Visual Studio фрагментов, что позволяет добавлять их, а не вводить вручную.
  2. Если вы не знакомы с концепцией использования фрагментов кода в Visual Studio, обратитесь к документу **Setup.docx**, находящемуся в подкаталоге **Assets** в каталоге с материалами к данной работе.

Упражнения

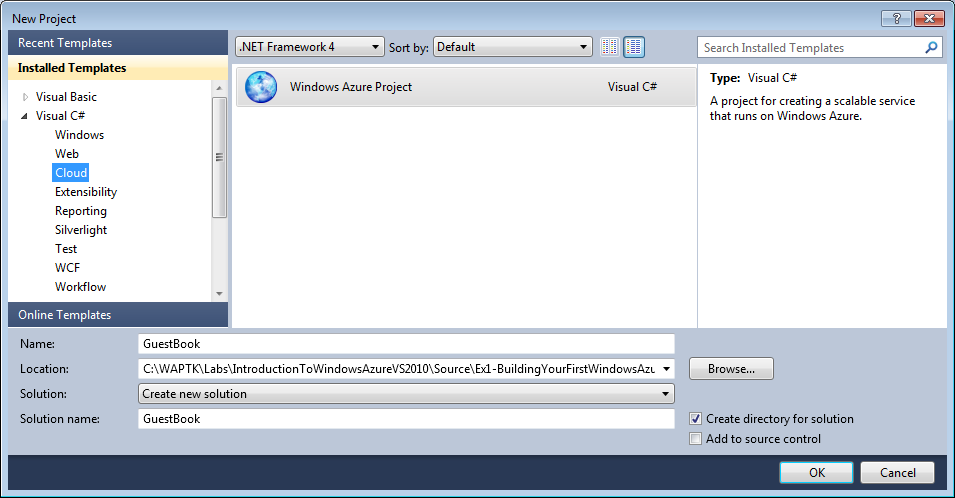
* 1. Данная лабораторная работа содержит описание следующих упраждений:
  + Создание вашего первого приложения для Windows Azure
  + Реализацию фоновой обработки данных с использованием очередей и прикладных ролей
  + Развертывание приложения в Windows Azure
  1. Приблизительное время выполнения данной работы: **60-75 минут**.
  2. **Внимание:** при первом запуске Visual Studio вам будет предложено выбрать один из имеющихся наборов настроек. Каждый набор ориентирован на определенный стиль разработки и определяет схему размещения окон, поведение редакторов, перечень доступных фрагментов кода и другие настройки. Описанные ниже действия расчитаны на использование набора настроек под названием «**General Development Settings**». Если вы выберете другой набор настроек, порядок выполнения тех или иных действий может отличаться от описанного.

# Упражнение 1: создание вашего первого приложения для Windows Azure

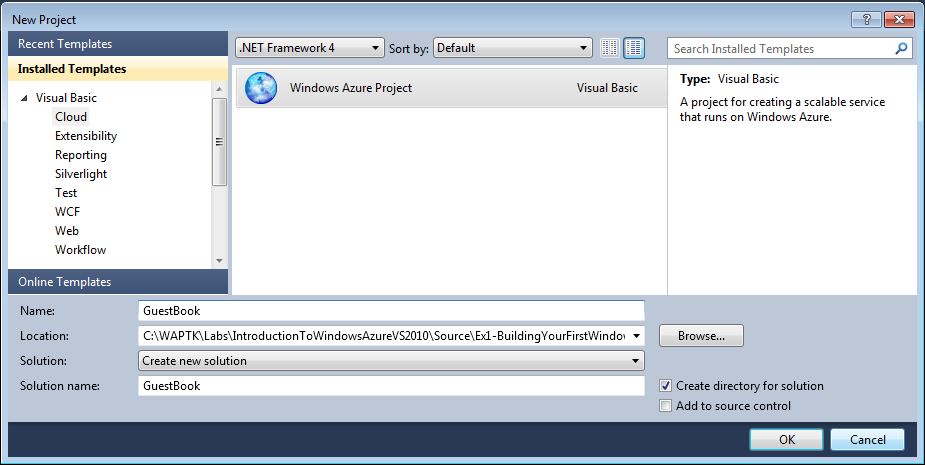
* 1. В этом упражнении вы создадите приложение – гостевую книгу – и запустите его локально. Для разработки вам потребуется набор расширений для Visual Studio 2010 - Windows Azure Tools for Microsoft Visual Studio. Данный набор добавляет в Visual Studio новую функциональность, позволяя создавать приложения и сервисы для Windows Azure, что вы и будете делать в оставшейся части работы.
  2. **Внимание:** чтобы уменьшить объем ручного ввода кода, вы можете использовать установленные фрагменты кода. Для этого щелкните правой кнопкой мыши в необходимом месте кода, выберите в контекстном меню пункт Insert Snippet - My Code Snippets и необходимый для выполнения текущего упражнения фрагмент кода.

## Задача 1 – создание проекта в Visual Studio

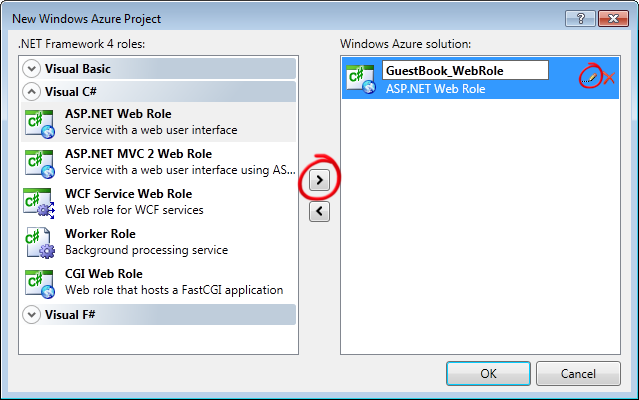
* 1. В данной задаче вы создадите новый проект типа Cloud Service (облачный сервис).
  2. Откройте Visual Studioот имени административной учетной записи: выберите пункты меню **Start | All Programs | Microsoft Visual Studio 2010**, щелкните правой кнопкой мыши на ярлыке **Microsoft Visual Studio 2010** и выберите пункт **Run as administrator**.
  3. В случае появления диалога **User Account Control** нажмите кнопку **Continue**.
  4. В меню **File** выберите пункт **New**, затем **Project**.
  5. В диалоге **New Project** разверните в списке **Installed Templates** соответствующий предпочитаемому вами языку узел (Visual C# или Visual Basic) и выберите пункт **Cloud**. Выберите шаблон проекта **Windows Azure Project**, присвойте проекту имя (поле **Name)** – в нашем случае **GuestBook**, установите расположение (поле **Location**) в **Source\Ex1-BuildingYourFirstWindowsAzureApp\[CS|VB]** в подкаталоге в каталоге с материалами, измените имя решения (**Solution name**) на **GuestBook**, и убедитесь в том, что установлен флажок **Create directory for solution** (Создать каталог для решения). Нажмите кнопку **OK** для создания проекта.



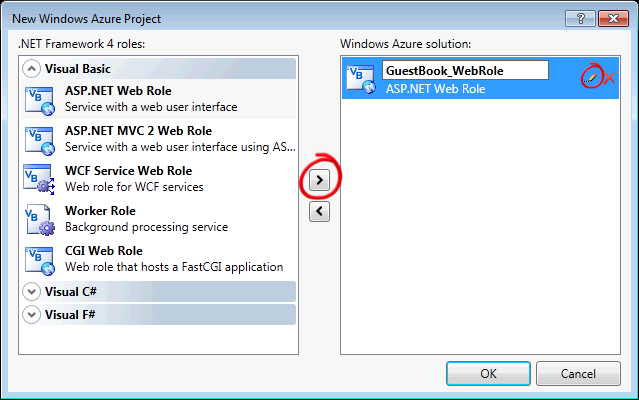
* + 1. Рисунок 1
    2. Создание нового проекта для Windows Azure (C#)



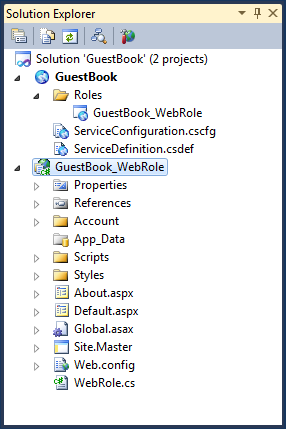
* + 1. Рисунок 2
    2. Создание нового проекта для Windows Azure (Visual Basic)
    3. **Note:** Windows Azure поддерживает .NET Framework 4.0. Если вы работаете с Visual Studio 2010, то можете выбрать данную версию платформы и воспользоваться предоставляемыми ей преимуществами.
  1. В диалоге **New Windows Azure Project** на панели **Roles** раскройте закладку, соответствующую предпочитаемому вами языку (Visual C# или Visual Basic), выберите из списка тип роли **ASP.NET Web Role** и нажмите кнопку с указывающей вправо стрелкой (>), чтобы добавить экземпляр роли данного типа в решение. Перед тем, как закрыть данный диалог, выберите добавленную роль в правой панели, нажмите на пиктограмму с изображением карандаша и переименуйте роль в **GuestBook\_WebRole**. Нажмите кнопку **OK** для создания решения.



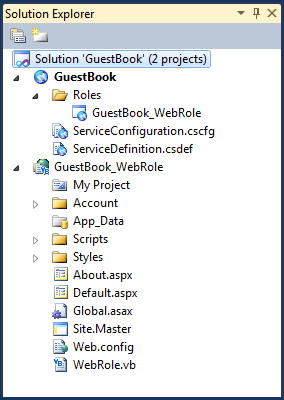
* + 1. Рисунок 3
    2. Добавление ролей в проект для Windows Azure (C#)



* + 1. Рисунок 4
    2. Добавление ролей в проект для Windows Azure (Visual Basic)
  1. В обозревателе решения (**Solution Explorer**)отобразится структура созданного решения.



* + 1. Рисунок 5
    2. Обозреватель решения (Solution Explorer), отображающий структуру решения GuestBook (C#)



* + 1. Рисунок 6
    2. Обозреватель решения (Solution Explorer), отображающий структуру решения GuestBook (Visual Basic)
    3. **Внимание:** В созданном решении содержится два проекта. Первый проект, называющийся **GuestBook**, содержит конфигурацию ролей, образующих облачное приложение. Он включает файл определения службы (service definition file), **ServiceDefinition.csdef**, содержащий используемые фабрикой Windows Azure метаданные: перечень ролей и их уровень доверия, публикуемые ролями конечные точки, используемое локальное хранилище и сертификаты. В этом же файле определяется перечень специфических для приложения настроек. В файле конфигурации службы (service configuration file), **ServiceConfiguration.cscfg**, устанавливается количество экземпляров роли каждого типа, значения настроек, перечень которых содержится в файле определения службы. Разделение настроек между файлами определения и конфигурации позволяет обновлять настройки, в том числе изменять число экземпляров ролей, без необходимости перезапуска приложения.
    4. Узел **Roles** в облачном проекте позволяет настроить перечень ролей (веб, прикладная или их сочетание) и связанные с ними проекты. Добавление и настройка ролей в данный узел обновляет содержимое файлов **ServiceDefinition.csdef** и **ServiceConfiguration.cscfg**.
    5. Второй проект, под названием **GuestBook\_WebRole** – это традиционное ASP.NET приложение, доработанное для функционирования в среде Windows Azure. Оно содержит дополнительный класс, предоставляющий точку входа в приложение и методы, управляющие инициализацией, запуском и остановом роли.

## Задача 2 – Создание модели данных для работы с табличным хранилищем

Приложение хранит записи гостевой книги в табличном хранилище Windows Azure. Табличное хранилище предоставляет частично структурированные таблицы, содержащие наборы сущностей. Каждая сущность содержит первичный ключ и набор типизированных свойств в формате свойство-значение.

Дополнительно к используемым вашей моделью свойствам каждая сущность в табличном хранилище должна иметь два ключевых свойства: **PartitionKey** и **RowKey**. Вместе эти два свойства формируют первичный ключ и уникально идентифицируют каждую сущность в таблице. Кроме того, каждая сущность имеет системное свойство **Timestamp**, используемое службой для хранения даты последних изменений. Данное поле предназначено для служебного использования и не может изменяться внутри приложения. Программный интерфейс Table Storage client API предоставляет класс **TableServiceEntity**, в котором уже определены необходимые свойства. Хотя вы можете унаследовать от него ваш собственный класс, это не является обязательным условием.

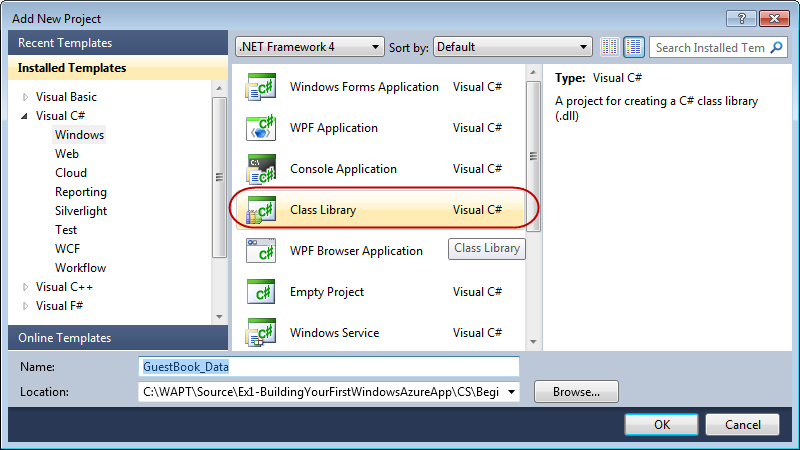
Программный интерфейс Table service API совместим с предоставляемым технологией [WCF Data Services](http://msdn.microsoft.com/en-us/library/cc668792.aspx) (ранее она называлась ADO.NET Data Services Framework) протоколом REST API, что позволяет использовать библиотеку [WCF Data Services Client Library](http://msdn.microsoft.com/en-us/library/cc668772.aspx) (ранее она называлась .NET Client Library) и работать с табличным хранилищем с использованием объектов .NET.

Табличное хранилище не содержит какой-либо информации о схеме таблиц, что позволяет хранить в одной таблице сущности с различными наборами свойств. В то же время этот подход не является рекомендованным и в демонстрационном приложении GuestBook в таблицах хранятся данные с фиксированной схемой.

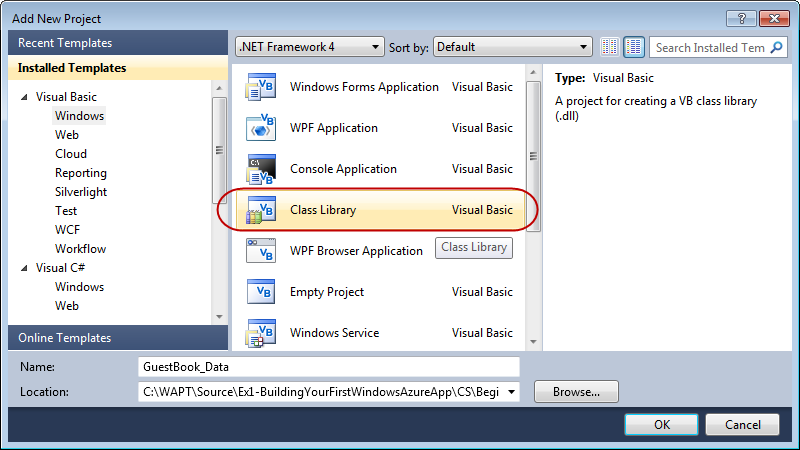
Чтобы работать с данными в хранилище с использованием WCF Data Services Client Library, необходимо определить класс, наследующий от **TableServiceContext**, в свою очередь являющийся потомком описанного в WCF Data Services класса **DataServiceContext**. Интерфейс Table Storage API позволяет приложениям создавать таблицы на основании описанных контекстных классов. Чтобы это стало возможным, каждый контекстный класс должен публиковать таблицы как свойства типа **IQueryable<*SchemaClass*>**, где ***SchemaClass*** – конкретный класс, описывающий хранящуюся в таблице сущность.

В этой задаче вы создадите схему для хранения сущностей, которая потребуется для приложения GuestBook. Затем вы создадите классы, которые позволят использовать WCF Data Services для доступа к табличному хранилищу. В завершение задачи вы создадите объект, который будет использоваться элементами управления ASP.NET и предоставит функциональность чтения, обновления и удаления записей.

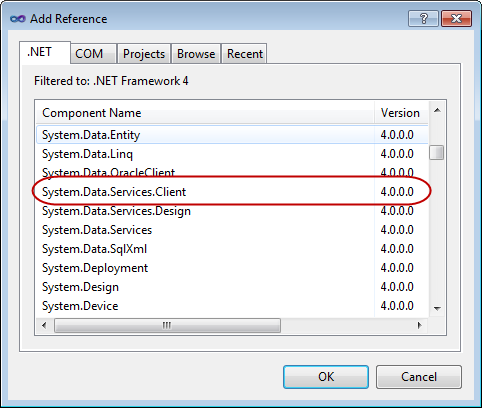
* 1. Создайте проект для размещения в нем классов схемы. Для этого в меню **File** выберите пункты **Add**, затем **New Project**.
  2. В диалоге **Add New Project** в панели **Installed Templates** раскройте соответствующий предпочитаемому языку узел, выберите категорию **Windows**, укажите шаблон проекта **Class Library**. Измените имя на **GuestBook\_Data**, не меняя предлагаемое расположение, и нажмите **OK**.



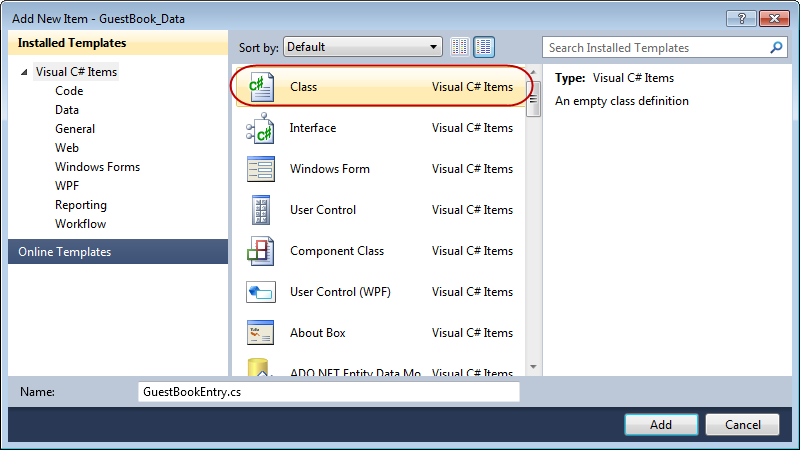
* + 1. Рисунок 7
    2. Создание проекта для хранения сущностных классов (C#)



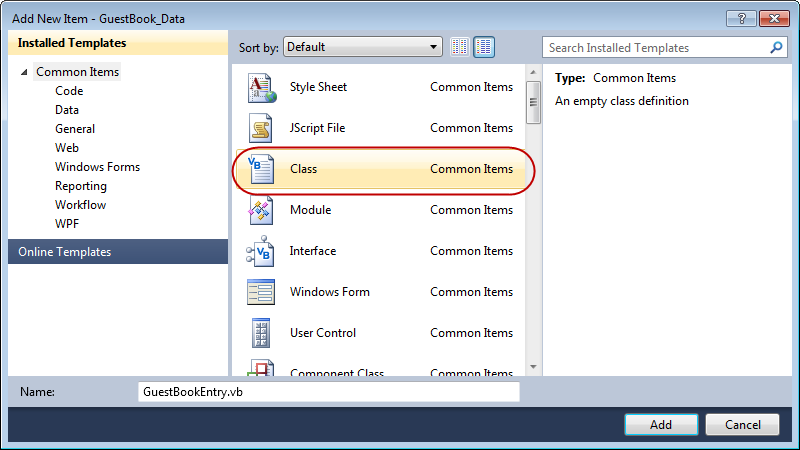
* + 1. Рисунок 8
    2. Создание проекта для хранения сущностных классов (Visual Basic)
  1. Удалите сгенерированный класс. Для этого щелкните правой кнопкой на файле **Class1.cs** (для проекта на Visual C#) или **Class1.vb** (для проекта на Visual Basic) и выберите **Delete**. Нажмите кнопку **OK** в диалоге подтверждения.
  2. Добавьте в проект **GuestBook\_Data** ссылку на библиотеку .NET Client Library for WCF Data Services. На панели **Solution Explorer** щелкните правой кнопкой на узле проекта **GuestBook\_Data**, выберите **Add Reference**, перейдите на закладку **.NET**, выберите компонент **System.Data.Services.Client** и нажмите **OK**.



* + 1. Рисунок 9
    2. Добавление ссылки на компонент System.Data.Service.Client
  1. Повторите предыдущий шаг и добавьте ссылку на сборку Windows Azure storage client API, для чего выберите компонент **Microsoft.WindowsAzure.StorageClient**.
  2. Перед тем, как сохранять сущность в таблице, для нее необходимо описать схему. Чтобы сделать это, щелкните в панели **Solution Explorer** правой кнопкой на проекте **GuestBook\_Data**, выберите **Add**, затем **Class**. В диалоге **Add New Item** укажите имя - **GuestBookEntry.cs** (для проекта на Visual C#) или **GuestBookEntry.vb** (для проекта на Visual Basic) и нажмите кнопку **Add**.



* + 1. Рисунок 10
    2. Добавление класса GuestBookEntry (C#)



* + 1. Рисунок 11
    2. Добавление класса GuestBookEntry (Visual Basic)
  1. В верхней части файла подключите пространство имен **Microsoft.WindowsAzure.StorageClient**.
     1. C#
     2. **using Microsoft.WindowsAzure.StorageClient;**
     3. Visual Basic
     4. **Imports Microsoft.WindowsAzure.StorageClient**
  2. Откройте файл **GuestBookEntry.cs** (для проекта на Visual C#) или **GuestBookEntry.vb** (для проекта на Visual Basic) и обновите объявление класса **GuestBookEntry**, сделав его открытым и наследующим от класса **TableServiceEntity**.
     1. **Внимание:** в проект на Visual Basic класс сразу добавляется как открытый (Public).
     2. C#
     3. **public** class GuestBookEntry
     4. **: Microsoft.WindowsAzure.StorageClient.TableServiceEntity**
     5. {
     6. }
     7. Visual Basic
     8. **Public** Class GuestBookEntry
     9. **Inherits Microsoft.WindowsAzure.StorageClient.TableServiceEntity**
     10. End Class

**Внимание:** класс **TableServiceEntity** объявлен в библиотеке Storage Client API. Он содержит необходимые каждой сущности системные свойства **PartititionKey**, **RowKey** и **TimeStamp**.

* + 1. Свойства **PartitionKey** и **RowKey** вместе определяют ключ **DataServiceKey**, уникально идентифицирующий каждую сущность в таблице.
  1. Добавьте в класс **GuestBookEntry** конструктор, инициализирующий свойства **PartitionKey** и **RowKey**.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookEntry constructor – CS*)
     2. C#
     3. **public GuestBookEntry()**
     4. **{**
     5. **PartitionKey = DateTime.UtcNow.ToString("MMddyyyy");**
     6. **// Row key allows sorting, so we make sure the rows come back in time order.**
     7. **RowKey = string.Format("{0:10}\_{1}", DateTime.MaxValue.Ticks - DateTime.Now.Ticks, Guid.NewGuid());**
     8. **}**
     9. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookEntry constructor – VB*)
     10. Visual Basic
     11. **Public Sub New()**
     12. **PartitionKey = DateTime.UtcNow.ToString("MMddyyyy")**
     13. **' Row key allows sorting, so we make sure the rows come back in time order.**
     14. **RowKey = String.Format("{0:10}\_{1}", DateTime.MaxValue.Ticks - DateTime.Now.Ticks, Guid.NewGuid())**
     15. **End Sub**
     16. **Внимание:** в приложении GuestBook для разбиения по разделам в качестве значения свойства **PartitionKey** используется дата, что приводит к созданию отдельного раздела для сущностей, созданных в каждый день. Выбирать значение для ключа раздела следует таким образом, чтобы оно позволяло воспользоваться преимуществами балансировки нагрузки.
     17. Свойство **RowKey** получает инвертированное значение даты и времени с добавленным для уникальности идентификатором (GUID). Внутри разделов данные отсортированы по значению свойства RowKey, поэтому такой способ формирования сразу отсортирует записи в необходимом порядке – новые записи размещаются выше старых.
  2. Чтобы завершить описание класса **GuestBookEntry**, добавьте свойства **Message**, **GuestName**, **PhotoUrl** и **ThumbnailUrl**,хранящие полезную нагрузку.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Table Schema Properties – CS*)
     2. C#
     3. **public string Message { get; set; }**
     4. **public string GuestName { get; set; }**
     5. **public string PhotoUrl { get; set; }**
     6. **public string ThumbnailUrl { get; set; }**
     7. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Table Schema Properties – VB*)
     8. Visual Basic
     9. **Public Property Message As String**
     10. **Public Property GuestName As String**
     11. **Public Property PhotoUrl As String**
     12. **Public Property ThumbnailUrl As String**
  3. Сохраните файл **GuestBookEntry.cs** (для проекта на Visual C#) или **GuestBookEntry.vb** (для проекта на Visual Basic).
  4. Далее необходимо создать класс, позволяющий взаимодействовать с табличным хранилищем с использованием WCF Data Services. Чтобы сделать это, на панели **Solution Explorer** щелкните правой кнопкой на проекте **GuestBook\_Data**, выберите **Add**, затем **Class**. В диалоге **Add New Item** установите имя (поле **Name)** в **GuestBookDataContext.cs** (для проекта на Visual C#) или **GuestBookDataContext.vb** (для проекта на Visual Basic) и нажмите **Add**.
  5. В созданном классе обновите его объявление таким образом, чтобы он стал открытым и наследовал от класса **TableServiceContext**.
     1. **Внимание:** в проект на Visual Basic класс сразу добавляется как открытый (Public).
     2. C#
     3. **public** class GuestBookDataContext
     4. **: Microsoft.WindowsAzure.StorageClient.TableServiceContext**
     5. {
     6. }
     7. Visual Basic
     8. **Public** Class GuestBookDataContext
     9. **Inherits Microsoft.WindowsAzure.StorageClient.TableServiceContext**
     10. End Class
  6. Теперь добавьте конструктор, инициализирующий базовый класс информацией о реквизитах для доступа к табличному хранилищу.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataContext Class – CS*)
     2. C#
     3. public class GuestBookDataContext
     4. : Microsoft.WindowsAzure.StorageClient.TableServiceContext
     5. {
     6. **public GuestBookDataContext(string baseAddress, Microsoft.WindowsAzure.StorageCredentials credentials)**
     7. **: base(baseAddress, credentials)**
     8. **{ }**
     9. }
     10. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataContext Class – VB*)
     11. Visual Basic
     12. Public Class GuestBookDataContext
     13. Inherits Microsoft.WindowsAzure.StorageClient.TableServiceContext
     14. **Public Sub New(ByVal baseAddress As String, ByVal credentials As Microsoft.WindowsAzure.StorageCredentials)**
     15. **MyBase.New(baseAddress, credentials)**
     16. **End Sub**
     17. End Class
     18. **Внимание:** класс **TableServiceContext** наследует от **DataServiceContext** из библиотеки WCF Data Services, хранит реквизиты для доступа к табличному хранилищу, а также реализует вызов операций и механизм повтора запросов.
  7. Добавьте в класс **GuestBookDataContext** свойство для таблицы сущностей типа **GuestBookEntry**. Чтобы сделать это, вставьте следующий (выделенный) код в класс.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookEntry Property – CS*)
     2. C#
     3. public class GuestBookDataContext
     4. : Microsoft.WindowsAzure.StorageClient.TableServiceContext
     5. {
     6. ...
     7. **public IQueryable<GuestBookEntry> GuestBookEntry**
     8. **{**
     9. **get**
     10. **{**
     11. **return this.CreateQuery<GuestBookEntry>("GuestBookEntry");**
     12. **}**
     13. **}**
     14. }
     15. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookEntry Property – VB*)
     16. Visual Basic
     17. Public Class GuestBookDataContext
     18. Inherits Microsoft.WindowsAzure.StorageClient.TableServiceContext
     19. ...
     20. **Public ReadOnly Property GuestBookEntry() As IQueryable(Of GuestBookEntry)**
     21. **Get**
     22. **Return Me.CreateQuery(Of GuestBookEntry)("GuestBookEntry")**
     23. **End Get**
     24. **End Property**
     25. End Class
     26. **Внимание:** для создания таблиц вы можете воспользоваться методом **CreateTablesFromModel** класса **CloudTableClient**. Когда вы передаете в этот метод в качестве параметра экземпляр класса, наследующего от **DataServiceContext** (или от **TableServiceContext**), он находит свойства, возвращающие **IQueryable<*T*>**, где ***T*** определяет класс таблицы, и создает в хранилище соответствующие таблицы.
  8. Теперь вам понадобится объект, который может быть использован для привязки данных в компонентах ASP.NET. На панели **Solution Explorer** щелкните правой кнопкой на проекте **GuestBook\_Data**, выберите **Add**, затем **Class**. В диалоге **Add New Item** измените имя на **GuestBookDataSource.cs** (для проекта на Visual C#) или **GuestBookDataSource.vb** (для проекта на Visual Basic) и нажмите **Add**.
  9. Во вновь добавленный класс импортируйте пространства имен **Microsoft.WindowsAzure** и **Microsoft.WindowsAzure.StorageClient**.
     1. C#
     2. **using Microsoft.WindowsAzure;**
     3. **using Microsoft.WindowsAzure.StorageClient;**
     4. Visual Basic
     5. **Imports Microsoft.WindowsAzure**
     6. **Imports Microsoft.WindowsAzure.StorageClient**
  10. Сделайтекласс **GuestBookDataSource** открытым (**public**), а также добавьте в него свойства для класса контекста и информации об учетной записи хранилища, как показано ниже.
      1. **Внимание:** в проект на Visual Basic класс сразу добавляется как открытый (Public).
      2. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Fields – CS*)
      3. C#
      4. public class GuestBookDataSource
      5. {
      6. **private static CloudStorageAccount storageAccount;**
      7. **private GuestBookDataContext context;**

}

* + 1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Fields – VB*)
    2. Visual Basic
    3. Public Class GuestBookDataSource
    4. **Private Shared storageAccount As CloudStorageAccount**
    5. **Private context As GuestBookDataContext**
    6. End Class
  1. Теперь добавьте статический конструктор (Shared в Visual Basic), как показано в следующем фрагменте кода (подсвеченная часть). Код в конструкторе создает таблицы на основе описания класса **GuestBookDataContext**.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Static Constructor – CS*)
     2. C#
     3. public class GuestBookDataSource
     4. {
     5. ...
     6. **static GuestBookDataSource()**
     7. **{**
     8. **storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     9. **CloudTableClient.CreateTablesFromModel(**
     10. **typeof(GuestBookDataContext),**
     11. **storageAccount.TableEndpoint.AbsoluteUri,**
     12. **storageAccount.Credentials);**
     13. **}**

}

* + 1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Shared Constructor – VB*)
    2. Visual Basic
    3. Public Class GuestBookDataSource
    4. ...
    5. **Shared Sub New()**
    6. **storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
    7. **CloudTableClient.CreateTablesFromModel(GetType(GuestBookDataContext), storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials)**
    8. **End Sub**
    9. End Class
    10. **Внимание:** статический (Shared в Visual Basic) конструктор инициализирует хранилище реквизитами учетной записи, извлекаемыми из файла конфигурации. Затем он вызывает метод **CreateTablesFromModel** класса **CloudTableClient**,создающий таблицы на основании описания класса **GuestBookDataContext**. Использование статического конструктора позволяет выполнить задачи инициализации единожды.
  1. Добавьте в класс **GuestBookDataSource** конструктор, инициализирующий использующийся для доступа к данным класс.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Constructor – CS*)
     2. C#
     3. public class GuestBookDataSource
     4. {
     5. ...
     6. **public GuestBookDataSource()**
     7. **{**
     8. **this.context = new GuestBookDataContext(storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials);**
     9. **this.context.RetryPolicy = RetryPolicies.Retry(3, TimeSpan.FromSeconds(1));**
     10. **}**

}

* + 1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Constructor – VB*)
    2. Visual Basic
    3. Public Class GuestBookDataSource
    4. ...
    5. **Public Sub New()**
    6. **Me.context = New GuestBookDataContext(storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials)**
    7. **Me.context.RetryPolicy = RetryPolicies.Retry(3, TimeSpan.FromSeconds(1))**
    8. **End Sub**
    9. End Class
  1. Теперь добавьте метод, возвращающий содержимое таблицы *GuestBookEntry*.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Select – CS*)
     2. C#
     3. public class GuestBookDataSource
     4. {
     5. ...
     6. **public IEnumerable<GuestBookEntry> GetGuestBookEntries()**
     7. **{**
     8. **var results = from g in this.context.GuestBookEntry**
     9. **where g.PartitionKey == DateTime.UtcNow.ToString("MMddyyyy")**
     10. **select g;**
     11. **return results;**
     12. **}**

}

* + 1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource Select – VB*)
    2. Visual Basic
    3. Public Class GuestBookDataSource
    4. ...
    5. **Public Function GetGuestBookEntries() As IEnumerable(Of GuestBookEntry)**
    6. **Dim results = From g In Me.context.GuestBookEntry \_**
    7. **Where g.PartitionKey = DateTime.UtcNow.ToString("MMddyyyy") \_**
    8. **Select g**
    9. **Return results**
    10. **End Function**
    11. End Class
    12. **Внимание:** метод **GetGuestBookEntries** возвращает записи за сегодняшний день, для чего использует LINQ-выражение с отбором по значению свойства PartitionKey. Веб-роль использует этот метод для привязки данных и отображения содержимого гостевой книги.
  1. Теперь добавьте метод, добавляющий запись в таблицу *GuestBookEntry*.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource AddGuestBookEntry – CS*)
     2. C#
     3. public class GuestBookDataSource
     4. {
     5. ...
     6. **public void AddGuestBookEntry(GuestBookEntry newItem)**
     7. **{**
     8. **this.context.AddObject("GuestBookEntry", newItem);**
     9. **this.context.SaveChanges();**
     10. **}**

}

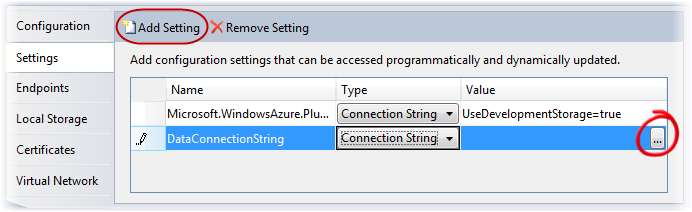
* + 1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource AddGuestBookEntry – VB*)
    2. Visual Basic
    3. Public Class GuestBookDataSource
    4. ...
    5. **Public Sub AddGuestBookEntry(ByVal newItem As GuestBookEntry)**
    6. **Me.context.AddObject("GuestBookEntry", newItem)**
    7. **Me.context.SaveChanges()**
    8. **End Sub**
    9. End Class
    10. **Внимание:** описанный выше метод добавляет объект типа **GuestBookEntry** в контекстный класс, после чего вызывает метод **SaveChanges**, непосредственно записывающий сущность в хранилище.
  1. В заключение добавьте метод, обновляющий URL изображения-миниатюры.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource UpdateImageThumbnail – CS*)
     2. C#
     3. public class GuestBookDataSource
     4. {
     5. ...
     6. **public void UpdateImageThumbnail(string partitionKey, string rowKey, string thumbUrl)**
     7. **{**
     8. **var results = from g in this.context.GuestBookEntry**
     9. **where g.PartitionKey == partitionKey && g.RowKey == rowKey**
     10. **select g;**
     11. **var entry = results.FirstOrDefault<GuestBookEntry>();**
     12. **entry.ThumbnailUrl = thumbUrl;**
     13. **this.context.UpdateObject(entry);**
     14. **this.context.SaveChanges();**
     15. **}**

}

* + 1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 GuestBookDataSource UpdateImageThumbnail – VB*)
    2. Visual Basic
    3. Public Class GuestBookDataSource
    4. ...
    5. **Public Sub UpdateImageThumbnail(ByVal partitionKey As String, ByVal rowKey As String, ByVal thumbUrl As String)**
    6. **Dim results = From g In Me.context.GuestBookEntry \_**
    7. **Where g.PartitionKey = partitionKey AndAlso g.RowKey = rowKey \_**
    8. **Select g**
    9. **Dim entry = results.FirstOrDefault()**
    10. **entry.ThumbnailUrl = thumbUrl**
    11. **Me.context.UpdateObject(entry)**
    12. **Me.context.SaveChanges()**
    13. **End Sub**
    14. End Class
    15. **Внимание:** метод **UpdateImageThumbnail** находит сущность по ключу раздела и ключу строки; обновляет значение свойства, уведомляет контекст об обновлении данных, после чего вызывает процедуру сохранения.
  1. Сохраните файла **GuestBookDataSource.cs** (для проекта на Visual C#) или **GuestBookDataSource.vb** (для проекта на Visual Basic).

## Задача 3 – создание веб-роли, позволяющей отображать содержимое гостевой книги и добавлять записи

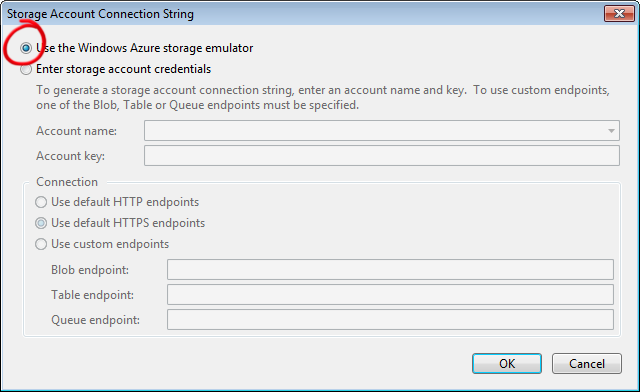
* 1. В этой задаче вы доработаете созданную в задаче 1 веб-роль. Изменения затронут пользовательский интерфейс, после чего он сможет отображать содержимое гостевой книги. Вы не будете настраивать содержимое страницы вручную, вместо этого возьмете существующую страницу, находящуюся в каталоге **Assets** с материалами к данному упражнению. Далее вы добавите код, сохраняющий сущности в таблице, а изображения – в хранилище двоичных объектов.
  2. Добавьте в веб-роль ссылку на проект **GuestBook\_Data**. В панели **Solution Explorer** щелкните правой кнопкой на узле проекта **GuestBook\_WebRole**, выберите **Add Reference**, переключитесь на закладку **Projects**, после чего выберите проект **GuestBook\_Data** и нажмите кнопку **OK**.
  3. При создании веб-роли была сгенерирована страница **Default.aspx**. Вы замените ее другой, предварительно подготовленной для вас. Чтобы удалить страницу, в панели **Solution Explorer** щелкните правой кнопкой на файле **Default.aspx** в проекте **GuestBook\_WebRole** и выберите **Delete**.
  4. Добавьте предварительно подготовленную страницу в веб-роль. Чтобы сделать это, щелкните правой кнопкой на проекте **GuestBook\_WebRole** в **Solution Explorer**, выберите **Add** | **Existing Item**. В диалоге **Add Existing Item** перейдите в каталог **Source\Ex1-BuildingYourFirstWindowsAzureApp\Assets**, выберите соответствующий предпочитаемому языку каталог (Visual C# или Visual Basic); удерживая кнопку **CTRL**, выберите все файлы и нажмите кнопку **Add**.
     1. **Внимание:** в каталоге **Assets** находится пять файлов, которые необходимо добавить в проект - Default.aspx с файлами кода и дизайна, один CSS и одно изображение.
  5. Откройте файл кода для главной страницы проекта **GuestBook\_WebRole**. Чтобы сделать это, щелкните правой кнопкой на файле **Default.aspx** и выберите пункт **View Code**.
  6. Добавьте объявления следующих пространств имен.
     1. (Фрагмент кода *– Introduction to Windows Azure - Ex1 Web Role Namespace Declarations – CS*)
     2. C#
     3. **using System.IO;**
     4. **using System.Net;**
     5. **using Microsoft.WindowsAzure;**
     6. **using Microsoft.WindowsAzure.ServiceRuntime;**
     7. **using Microsoft.WindowsAzure.StorageClient;**
     8. **using GuestBook\_Data;**
     9. (Фрагмент кода *– Introduction to Windows Azure - Ex1 Web Role Namespace Declarations – VB*)
     10. Visual Basic
     11. **Imports System.IO**
     12. **Imports System.Net**
     13. **Imports Microsoft.WindowsAzure**
     14. **Imports Microsoft.WindowsAzure.ServiceRuntime**
     15. **Imports Microsoft.WindowsAzure.StorageClient**
     16. **Imports GuestBook\_Data**
  7. Добавьте в класс **Default** объявления свойств.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Web Role Member Fields – CS*)
     2. C#
     3. public partial class Default : System.Web.UI.Page
     4. {
     5. **private static bool storageInitialized = false;**
     6. **private static object gate = new Object();**
     7. **private static CloudBlobClient blobStorage;**
     9. ...
     10. }
     11. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Web Role Member Fields – VB*)
     12. Visual Basic
     13. Partial Public Class \_Default
     14. Inherits System.Web.UI.Page
     15. **Private Shared storageInitialized As Boolean = False**
     16. **Private Shared gate As New Object()**
     17. **Private Shared blobStorage As CloudBlobClient**
     19. ...
     20. End Class
  8. Добавьте обработчик события **SignButton\_Click** со следующим содержимым.
     1. (Фрагмент кода *– Introduction to Windows Azure - Ex1 SignButton\_Click – CS*)
     2. C#
     3. public partial class Default : System.Web.UI.Page
     4. {
     5. ...
     6. protected void SignButton\_Click(object sender, EventArgs e)
     7. {
     8. **if (FileUpload1.HasFile)**
     9. **{**
     10. **InitializeStorage();**
     11. **// upload the image to blob storage**
     12. **string uniqueBlobName = string.Format("guestbookpics/image\_{0}{1}", Guid.NewGuid().ToString(), Path.GetExtension(FileUpload1.FileName));**
     13. **CloudBlockBlob blob = blobStorage.GetBlockBlobReference(uniqueBlobName);**
     14. **blob.Properties.ContentType = FileUpload1.PostedFile.ContentType;**
     15. **blob.UploadFromStream(FileUpload1.FileContent);**
     16. **System.Diagnostics.Trace.TraceInformation("Uploaded image '{0}' to blob storage as '{1}'", FileUpload1.FileName, uniqueBlobName);**
     17. **// create a new entry in table storage**
     18. **GuestBookEntry entry = new GuestBookEntry() { GuestName = NameTextBox.Text, Message = MessageTextBox.Text, PhotoUrl = blob.Uri.ToString(), ThumbnailUrl = blob.Uri.ToString() };**
     19. **GuestBookDataSource ds = new GuestBookDataSource();**
     20. **ds.AddGuestBookEntry(entry);**
     21. **System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName);**
     22. **}**
     23. **NameTextBox.Text = "";**
     24. **MessageTextBox.Text = "";**
     25. **DataList1.DataBind();**
     26. }
     27. }
     28. (Фрагмент кода – *Introduction to Windows Azure - Ex1 SignButton\_Click – VB*)
     29. Visual Basic
     30. Public Class \_Default
     31. Inherits System.Web.UI.Page
     32. ...
     33. Protected Sub SignButton\_Click(ByVal sender As Object, ByVal e As EventArgs) Handles SignButton.Click
     34. **If FileUpload1.HasFile Then**
     35. **InitializeStorage()**
     36. **' upload the image to blob storage**
     37. **Dim uniqueBlobName As String = String.Format("guestbookpics/image\_{0}{1}", Guid.NewGuid().ToString(), Path.GetExtension(FileUpload1.FileName))**
     38. **Dim blob As CloudBlockBlob = blobStorage.GetBlockBlobReference(uniqueBlobName)**
     39. **blob.Properties.ContentType = FileUpload1.PostedFile.ContentType**
     40. **blob.UploadFromStream(FileUpload1.FileContent)**
     41. **System.Diagnostics.Trace.TraceInformation("Uploaded image '{0}' to blob storage as '{1}'", FileUpload1.FileName, uniqueBlobName)**
     42. **' create a new entry in table storage**
     43. **Dim entry As New GuestBookEntry() With {.GuestName = NameTextBox.Text, .Message = MessageTextBox.Text, .PhotoUrl = blob.Uri.ToString(), .ThumbnailUrl = blob.Uri.ToString()}**
     44. **Dim ds As New GuestBookDataSource()**
     45. **ds.AddGuestBookEntry(entry)**
     46. **System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName)**
     47. **End If**
     48. **NameTextBox.Text = ""**
     49. **MessageTextBox.Text = ""**
     50. **DataList1.DataBind()**
     51. End Sub
     52. End Class
     53. **Внимание:** чтобыдобавить запись обработчик предварительно вызывает метод **InitializeStorage**,позволяющий убедиться в наличии контейнера двоичных объектов и доступа к нему. Данный метод будет создан вами в ближайшее время.
     54. Далее метод получает ссылку на контейнер, создает уникальное имя для объекта и создает сам объект, после чего загружает туда переданное пользователем изображение. Тип содержимого – свойство **ContentType** – устанавливается равным типу загруженного файла. Когда содержимое двоичного объекта будет прочитано, данный тип будет использован для корректного отображения изображения.
     55. В самом конце создается сущность типа **GuestBookEntry**, которую вы описали в предыдущей задаче, инициализируется введенными пользователем данными, после чего сохраняется с использованием методов класса **GuestBookDataSource**.
     56. Последним шагом процедуры выполняется привязка данных для обновления содержимого страницы.
  9. Обновите содержимое метода **Timer1\_Tick** в соответствии с приведенным ниже текстом.
     1. (Фрагмент кода *– Introduction to Windows Azure - Ex1 Timer1\_Tick – CS*)
     2. C#
     3. public partial class Default : System.Web.UI.Page
     4. {
     5. ...
     6. protected void Timer1\_Tick(object sender, EventArgs e)
     7. {
     8. **DataList1.DataBind();**
     9. }
     10. }
     11. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Timer1\_Tick – VB*)
     12. Visual Basic
     13. Public Class \_Default
     14. Inherits System.Web.UI.Page
     15. ...
     16. Protected Sub Timer1\_Tick(ByVal sender As Object, ByVal e As EventArgs) Handles Timer1.Tick
     17. **DataList1.DataBind()**
     18. End Sub
     19. End Class
     20. **Внимание:** таймер периодически принудительно обновляет содержимое страницы.
  10. Найдите обработчик события **Page\_Load** и обновите его содержимое в соответствие с приведенным ниже примером, чтобы включить таймер.
      1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Page\_Load – CS*)
      2. C#
      3. public partial class Default : System.Web.UI.Page
      4. {
      5. ...
      6. protected void Page\_Load(object sender, EventArgs e)
      7. {
      8. **if (!Page.IsPostBack)**
      9. **{**
      10. **Timer1.Enabled = true;**
      11. **}**
      12. }
      13. }
      14. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Page\_Load – VB*)
      15. Visual Basic
      16. Public Class \_Default
      17. Inherits System.Web.UI.Page
      18. ...
      19. Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
      20. **If Not (Page.IsPostBack) Then**
      21. **Timer1.Enabled = True**
      22. **End If**
      23. End Sub
      24. End Class
  11. Добавьте реализацию метода **InitializeStorage** в соответствии с приведенным ниже фрагментом (выделенная часть кода).
      1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 InitializeStorage – CS*)
      2. C#
      3. public partial class Default : System.Web.UI.Page
      4. {
      5. ...
      6. private void InitializeStorage()
      7. {
      8. **if (storageInitialized)**
      9. **{**
      10. **return;**
      11. **}**
      12. **lock (gate)**
      13. **{**
      14. **if (storageInitialized)**
      15. **{**
      16. **return;**
      17. **}**
      18. **try**
      19. **{**
      20. **// read account configuration settings**
      21. **var storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
      22. **// create blob container for images**
      23. **blobStorage = storageAccount.CreateCloudBlobClient();**
      24. **CloudBlobContainer container = blobStorage.GetContainerReference("guestbookpics");**
      25. **container.CreateIfNotExist();**
      26. **// configure container for public access**
      27. **var permissions = container.GetPermissions();**
      28. **permissions.PublicAccess = BlobContainerPublicAccessType.Container;**
      29. **container.SetPermissions(permissions);**
      30. **}**
      31. **catch (WebException)**
      32. **{**
      33. **throw new WebException("Storage services initialization failure. "**
      34. **+ "Check your storage account configuration settings. If running locally, "**
      35. **+ "ensure that the Development Storage service is running.");**
      36. **}**
      37. **storageInitialized = true;**
      38. **}**
      39. }
      40. }
      41. (Фрагмент кода – *Introduction to Windows Azure - Ex1 InitializeStorage – VB*)
      42. Visual Basic
      43. Public Class \_Default
      44. Inherits System.Web.UI.Page
      45. ...
      46. Private Sub InitializeStorage()
      47. **If storageInitialized Then**
      48. **Return**
      49. **End If**
      50. **SyncLock gate**
      51. **If storageInitialized Then**
      52. **Return**
      53. **End If**
      54. **Try**
      55. **' read account configuration settings**
      56. **Dim storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
      57. **' create blob container for images**
      58. **blobStorage = storageAccount.CreateCloudBlobClient()**
      59. **Dim container As CloudBlobContainer = blobStorage.GetContainerReference("guestbookpics")**
      60. **container.CreateIfNotExist()**
      61. **' configure container for public access**
      62. **Dim permissions = container.GetPermissions()**
      63. **permissions.PublicAccess = BlobContainerPublicAccessType.Container**
      64. **container.SetPermissions(permissions)**
      65. **Catch e1 As WebException**
      66. **Throw New WebException("Storage services initialization failure. " \_**
      67. **& "Check your storage account configuration settings. If running locally, " \_**
      68. **& "ensure that the Development Storage service is running.")**
      69. **End Try**
      70. **storageInitialized = True**
      71. **End SyncLock**
      72. End Sub
      73. End Class
      74. **Внимание:** в первую очередь метод **InitializeStorage** проверяет, не был ли он выполнен ранее. Затем он читает из файла конфигурации информацию об учетной записи хранилища, создает контейнер для изображений и настраивает для него возможность общего доступа.
  12. Поскольку веб-роль использует сервисы хранения данных Windows Azure (Windows Azure storage services), необходимо хранить реквизиты для подключения к ним. Чтобы создать новую настройку, в панели **Solution Explorer** разверните узел **Roles**, находящийся в проекте **GuestBook** и дважды щелкните на роли **GuestBook\_WebRole**. В настройках роли перейдите на закладку **Settings**. Нажмите **Add Setting**, укажите в качестве имени строку “*DataConnectionString*” (колонка **Name**), измените тип на *ConnectionString*, после чего нажмите кнопку с многоточием.



* + 1. Рисунок 12

Добавление в файл конфигурации информации об учетной записи хранилища

* 1. В диалоге **Storage Account Connection String** выберите вариант **Use the Windows Azure storage emulator** (использовать эмулируемое хранилище) и нажмите **OK**.



* + 1. Рисунок 13
    2. Создание строки подключения к эмулируемому хранилищу
    3. **Внимание:** учетная запись хранилища (табличное, двоичных объектов и очереди) создается отдельно, поэтому для использования ее необходимо предварительно создать, используя административный портал. В этом упражнении вы будете использовать эмулируемое хранилище, что позволяет разрабатывать и отлаживать приложения локально.
    4. Чтобы использовать эмулятор хранилища, необходимо в строке подключения указать значение **UseDevelopmentStorage=true**. Для подключения к реально существующему хранилищу необходимо указать протокол, имя хранилища и ключ для доступа:
    5. <Setting name="DataConnectionString" value="DefaultEndpointsProtocol=https;AccountName=YourAccountName;AccountKey=YourAccountKey" />
    6. где *YourAccountName* – имя сервиса хранения данных, а *YourAccountKey* ключ.
  1. Нажмите сочетание клавиш **CTRL + S**, чтобы сохранить изменения в файле конфигурации.
  2. Теперь необходимо предусмотреть, что будет служить в качестве источника информации о конфигурации. В проекте **GuestBook\_WebRole** откройте файл **Global.asax.cs** (для проекта на Visual C#) или **Global.asax.vb** (для проекта на Visual Basic).
  3. Добавьте объявления пространств имен Microsoft.WindowsAzure и Microsoft.WindowsAzure.ServiceRuntime.
     1. C#
     2. using Microsoft.WindowsAzure;
     3. using Microsoft.WindowsAzure.ServiceRuntime;
     4. Visual Basic
     5. Imports Microsoft.WindowsAzure
     6. Imports Microsoft.WindowsAzure.ServiceRuntime
  4. Добавьте в метод **Application\_Start** следующий фрагмент кода.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 SetConfigurationSettingPublisher – CS*)
     2. C#
     3. void Application\_Start(object sender, EventArgs e)
     4. {
     5. **Microsoft.WindowsAzure.CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =>**
     6. **{**
     7. **configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));**
     8. **});**
     9. }
     10. (Фрагмент кода – *Introduction to Windows Azure - Ex1 SetConfigurationSettingPublisher – VB*)
     11. Visual Basic
     12. Sub Application\_Start(ByVal sender As Object, ByVal e As EventArgs)
     13. **CloudStorageAccount.SetConfigurationSettingPublisher(Function(configName, configSetter) configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))**
     14. End Sub

## Задача 4 – использование очередей для организации фоновой обработки данных

* 1. В этом упражнении вы обновите веб-роль таким образом, чтобы она помещала в очередь требующие фоновой обработки рабочие элементы. Эти элементы будут находиться в очереди до тех пор, пока не будут извлечены из нее прикладной ролью. Прикладная роль извлекает из очереди очередной элемент и создает миниатюру для каждого добавленного пользователем изображения.
  2. Откройте файл кода для файла **Default.aspx**. Для этого щелкните на нем правой кнопкой мыши и выберите **View Code**.
  3. Добавьте в класс **Default** экземпляр класса **CloudQueueClient**, используемый для взаимодействия с очередью (выделенный фрагмент).
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 CloudQueueClient member – CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. private static bool storageInitialized = false;
     6. private static object gate = new Object();
     7. private static CloudBlobClient blobStorage;
     8. **private static CloudQueueClient queueStorage;**
     9. ...
     10. }
     11. (Фрагмент кода – *Introduction to Windows Azure - Ex1 CloudQueueClient member – VB*)
     12. Visual Basic
     13. Public Class \_Default
     14. Inherits System.Web.UI.Page
     15. Private Shared storageInitialized As Boolean = False
     16. Private Shared gate As New Object()
     17. Private Shared blobStorage As CloudBlobClient
     18. **Private Shared queueStorage As CloudQueueClient**
     19. ...
     20. End Class
  4. Теперь обновите инициализирующий хранилище код таким образом, чтобы он дополнительно создавал очередь, если не существует, и инициализировал добавленную выше переменную. Чтобы сделать это, найдите метод **InitializeStorage** и добавьте в него приведенный ниже код (выделенный фрагмент) сразу после кода, конфигурирующего контейнер двоичных объектов.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Create Queue – CS*)
     2. C#
     3. public partial class Default : System.Web.UI.Page
     4. {
     5. ...
     6. private void InitializeStorage()
     7. {
     8. ...
     9. try
     10. {
     11. ...
     12. // configure container for public access
     13. var permissions = container.GetPermissions();
     14. permissions.PublicAccess = BlobContainerPublicAccessType.Container;
     15. container.SetPermissions(permissions);
     16. **// create queue to communicate with worker role**
     17. **queueStorage = storageAccount.CreateCloudQueueClient();**
     18. **CloudQueue queue = queueStorage.GetQueueReference("guestthumbs");**
     19. **queue.CreateIfNotExist();**
     20. }
     21. catch (WebException)
     22. {
     23. ...

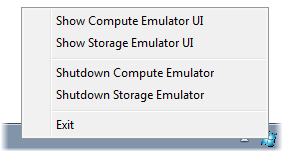
}

* + 1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Create Queue – VB*)
    2. Visual Basic
    3. Partial Public Class \_Default
    4. Inherits System.Web.UI.Page
    5. ...
    6. Private Sub InitializeStorage()
    7. ...
    8. Try
    9. ...
    10. ' configure container for public access
    11. Dim permissions = container.GetPermissions()
    12. permissions.PublicAccess = BlobContainerPublicAccessType.Container
    13. container.SetPermissions(permissions)
    14. **' create queue to communicate with worker role**
    15. **queueStorage = storageAccount.CreateCloudQueueClient()**
    16. **Dim queue As CloudQueue = queueStorage.GetQueueReference("guestthumbs")**
    17. **queue.CreateIfNotExist()**
    18. Catch e1 As WebException
    19. ...
    20. End Class
    21. **Внимание:** обновленный код создает очередь, используемую веб-ролью для передачи заданий прикладной роли.
  1. В завершение добавьте код, помещающий рабочие элементы в очередь. Найдите обработчик нажатия **SignButton\_Click** и вставьте в него приведенный ниже код (выделенный фрагмент) сразу после добавления новой сущности в табличное хранилище.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Queueing work items – CS*)
     2. C#
     3. protected void SignButton\_Click(object sender, EventArgs e)
     4. {
     5. if (FileUpload1.HasFile)
     6. {
     7. ...
     8. // create a new entry in table storage
     9. GuestBookEntry entry = new GuestBookEntry() { GuestName = NameTextBox.Text, Message = MessageTextBox.Text, PhotoUrl = blob.Uri.ToString(), ThumbnailUrl = blob.Uri.ToString() };
     10. GuestBookDataSource ds = new GuestBookDataSource();
     11. ds.AddGuestBookEntry(entry);
     12. System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName);
     13. **// queue a message to process the image**
     14. **var queue = queueStorage.GetQueueReference("guestthumbs");**
     15. **var message = new CloudQueueMessage(String.Format("{0},{1},{2}", blob.Uri.ToString(), entry.PartitionKey, entry.RowKey));**
     16. **queue.AddMessage(message);**
     17. **System.Diagnostics.Trace.TraceInformation("Queued message to process blob '{0}'", uniqueBlobName);**
     18. }
     19. NameTextBox.Text = "";
     20. MessageTextBox.Text = "";
     21. DataList1.DataBind();
     22. }
     23. (Фрагмент кода – *Introduction to Windows Azure - Ex1 Queueing work items – VB*)
     24. Visual Basic
     25. Protected Sub SignButton\_Click(ByVal sender As Object, ByVal e As EventArgs) Handles SignButton.Click
     26. If FileUpload1.HasFile Then
     27. ...
     28. ' create a new entry in table storage
     29. Dim entry As New GuestBookEntry() With {.GuestName = NameTextBox.Text, .Message = MessageTextBox.Text, .PhotoUrl = blob.Uri.ToString(), .ThumbnailUrl = blob.Uri.ToString()}
     30. Dim ds As New GuestBookDataSource()
     31. ds.AddGuestBookEntry(entry)
     32. System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName)
     33. **' queue a message to process the image**
     34. **Dim queue = queueStorage.GetQueueReference("guestthumbs")**
     35. **Dim message = New CloudQueueMessage(String.Format("{0},{1},{2}", blob.Uri.ToString(), entry.PartitionKey, entry.RowKey))**
     36. **queue.AddMessage(message)**
     37. **System.Diagnostics.Trace.TraceInformation("Queued message to process blob '{0}'", uniqueBlobName)**
     38. End If
     39. NameTextBox.Text = ""
     40. MessageTextBox.Text = ""
     41. DataList1.DataBind()
     42. End Sub

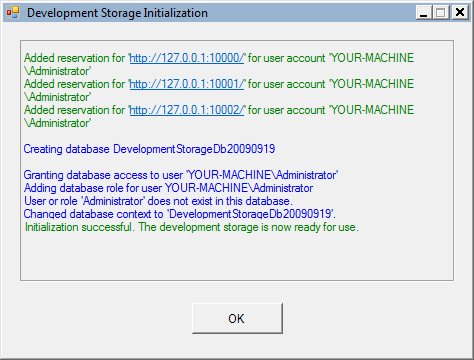
**Внимание:** обновленный код получает ссылку на очередь с именем “*guestthumbs*”. Он создает новое сообщение, состоящее из разделенной запятыми строки, содержащей имя двоичного объекта с изображением, ключ раздела и ключ строки для записи в табличном хранилище. Далее метод добавляет полученное сообщение в очередь.

## Проверка

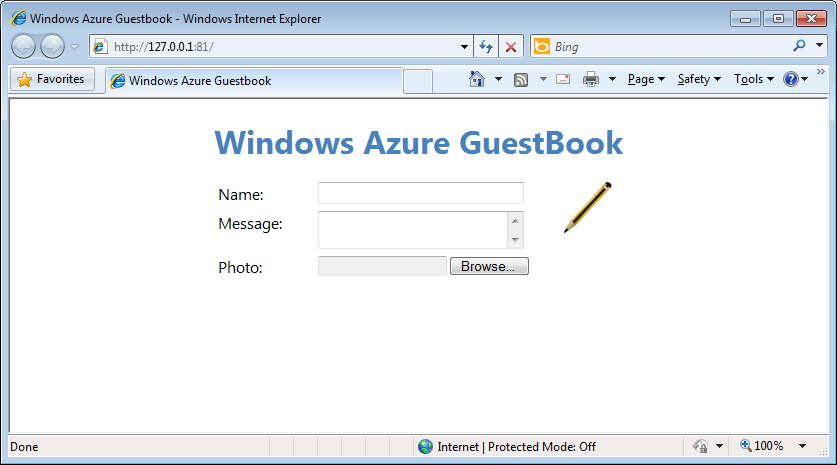
* 1. Эмулятор Windows Azure, (ранее был известен как Development Fabric или devfabric) позволяет локально разрабатывать и тестировать приложения. В этой задаче вы запустите приложение GuestBook локально, после чего добавите в гостевую книгу несколько записей.
  2. Среди прочих функций, предоставляемых набором инструментов Windows Azure Tools for Microsoft Visual Studio, присутствует Windows Azure Storage browser, позволяющий подключаться к учетной записи хранилища и просматривать находящиеся в нем таблицы и двоичные объекты.
  3. Нажмите **F5** для запуска. После завершения построения будет запущен локальный эмулятор Windows Azure. Чтобы отобразить доступный интерфейс, нажмите правой кнопкой на находящейся в системном трее пиктограмме Windows Azure и выберите пункт меню **Show Compute Emulator UI**.



* + 1. Рисунок 14
    2. Отображение интерфейса эмулятора
    3. **Внимание:** первый запуск эмулятора потребует значительно большего времени, чем все последующие, так как при этом выполняется инициализация необходимых баз данных и таблиц. Чтобы увидеть ход выполнения данной процедуры, вы можете воспользоваться диалогом **Development Storage Initialization**.

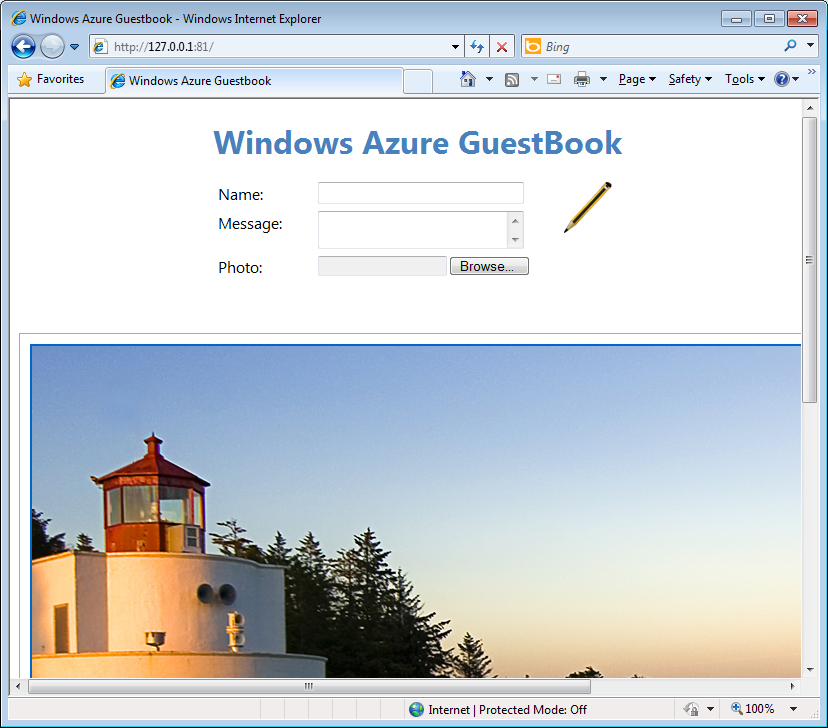


* + 1. Рисунок 15
    2. Процесс инициализации эмулятора хранилища
  1. Переключитесь в Internet Explorer, чтобы увидеть приложение GuestBook.
  2. Добавьте в гостевую книгу новую запись. Для этого введите ваше имя и текст сообщения, выберите рисунок и нажмите кнопку с изображением карандаша.

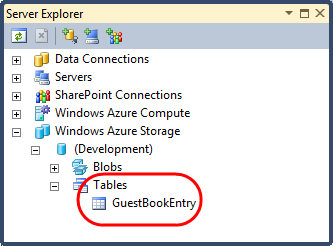


* + 1. Рисунок 16
    2. Главная страница проекта GuestBook
    3. **Внимание:** загрузка изображений в высоком разрешении позволит проверить законченное изображение, так как прикладная роль будет создавать для них миниатюры.

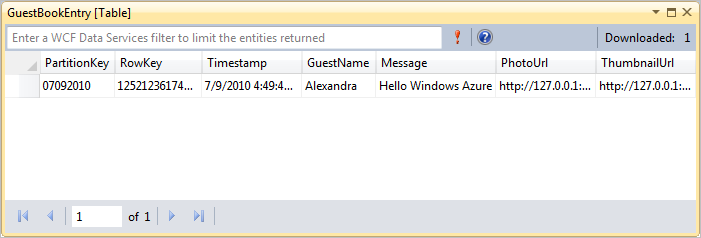
После того, как вы добавили запись, веб-роль создает сущность в табличном хранилище и загружает изображение в хранилище двоичных объектов. Находящийся на странице таймер обновляет ее раз в 5 секунд, поэтому новая запись отобразится в течение этого периода времени. До тех пор, пока миниатюра не создана, на странице будет отображаться оригинальное изображение.



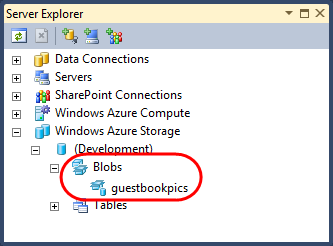
* + 1. Рисунок 17
    2. Приложение GuestBook, отображающее добавленное изображение оригинального размера
    3. Если вы работаете в Visual Studio 2010, то можете увидеть находящиеся в хранилище элементы на панели Windows Azure Storage Explorer. В Visual Studio 2008 эта функциональность недоступна.
  1. Чтобы открыть Storage Explorer в Visual Studio 2010, воспользуйтесь меню **View** | **Server Explorer**, после чего раскройте узел **Windows Azure Storage**.
     1. Узел **Windows Azure Storage** отображает список всех зарегистрированных учетных записей, в том числе локальное хранилище, помеченное как **(Development)**.
     2. **Внимание:** Windows Azure Storage Explorer недоступен в Visual Studio 2008.
  2. Раскройте узел **(Development)**, затем **Tables**. Убедитесь в наличии созданной приложением таблицы *GuestBookEntry*.



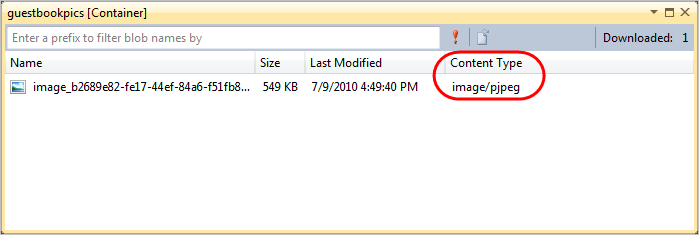
* + 1. Рисунок 18
    2. Просмотр таблиц в локальном хранилище Windows Azure
  1. Дважды щелкните на таблице *GuestBookEntry*, чтобы отобразить ее содержимое. Видно, что она содержит значения как для введенных пользователем свойств (*GuestName*, *Message*, *PhotoUrl* и *ThumbnailUrl*), так и для системных свойств (*PartitionKey*, *RowKey* и *Timestamp*). Свойства *PhotoUrl* и *ThumbnailUrl* в настоящее время ссылаются на один и тот же объект. В следующем упражнении вы создадите прикладную роль, генерирующую миниатюры и обновляющую записи соответствующим URL.



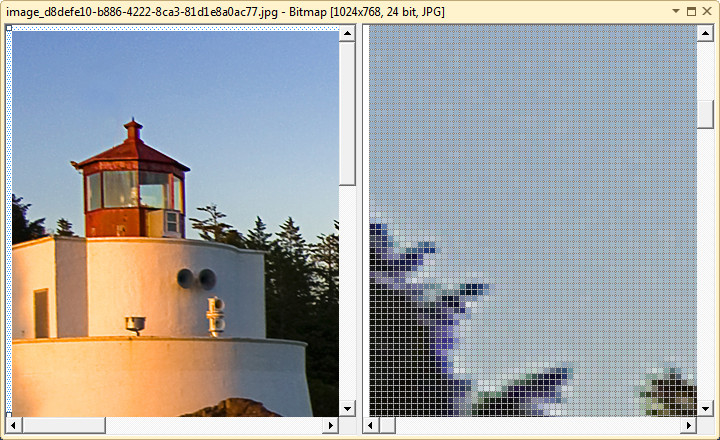
* + 1. Рисунок 19
    2. Просмотр содержимого таблицы с использованием средств Windows Azure Tools for Visual Studio
  1. Теперь разверните узел **Blobs**. Внутри находятся контейнеры – в нашем случае это единственный контейнер *guestbookpics*, в котором размещаются наши изображения.



* + 1. Рисунок 20
    2. Просмотр двоичного хранилища с использованием средств Windows Azure Tools for Visual Studio
  1. Дважды щелкните на контейнере *guestbookpics*. Результат содержит по одной записи для ранее добавленных изображений.



* + 1. Рисунок 21
    2. Просмотр содержимого двоичного контейнера
  1. Для каждой сущности в хранилище двоичных объектов в отдельном поле содержится тип содержимого, что позволяет Visual Studio выбрать соответствующий инструмент для просмотра. Чтобы просмотреть содержимое объекта, дважды щелкните на любой сущности.



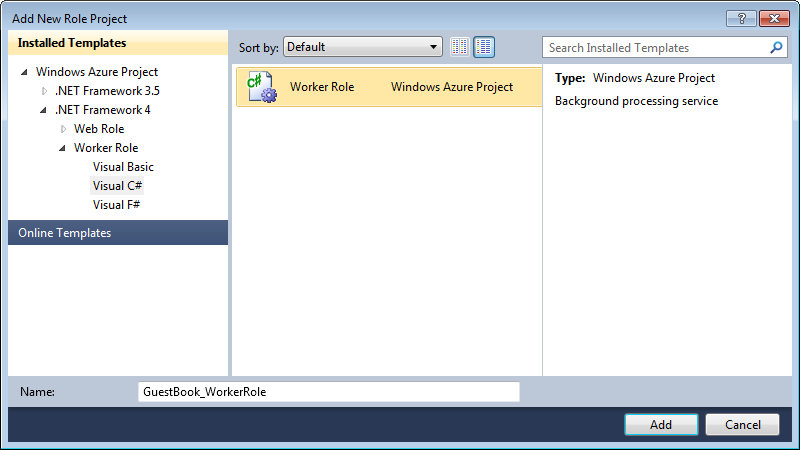
* + 1. Рисунок 22
    2. Просмотр содержимого двоичного объекта в Visual Studio
  1. Нажмите **SHIFT + F5** для остановки процесса отладки.

# Упражнение 2: фоновая обработка данных с использованием прикладной роли и очереди

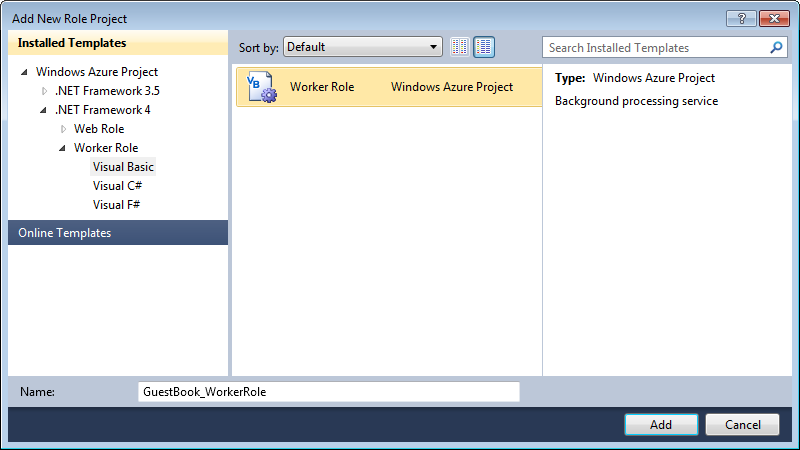
* 1. Прикладная роль работает в фоновом режиме – ее можно сравнить со службой Windows.
  2. В этом упражнении вы создадите прикладную роль, извлекающую сообщения из очереди, куда их помещает веб-роль. Прикладная роль извлекает изображение из хранилища двоичных объектов, создает для него миниатюру, которую затем также сохраняет как двоичный объект. В завершение URL файла-миниатюры помещается в табличное хранилище в соответствующую сущность.

## Задача 1 – создание прикладной роли для фоновой обработки данных

* 1. В этой задаче вы добавите прикладную роль в решение и доработаете ее таким образом, чтобы она могла извлекать сообщения из очереди и обрабатывать их.
  2. Запустите Visual Studioот имени администратора, если это не было сделано ранее: выберите пункт меню **Start | All Programs | Microsoft Visual Studio 2010**. Щелкните правой кнопкой на ярлыке **Microsoft Visual Studio 2010** и выберите элемент **Run as administrator**.
  3. В меню **File** выберите **Open** **|** **Project/Solution**. В диалоге **Open Project** перейдите в подкаталог **Source\Ex2-UsingWorkerRolesAndQueues\Begin** в папке с материалами к работе и укажите файл **Begin.sln** в соответствующем предпочитаемому языку каталоге (Visual C# или Visual Basic), затем нажмите **Open**. Вы также можете продолжить работу с решением, полученным после завершения предыдущего шага.
  4. В панели **Solution Explorer** нажмите правой кнопкой на узле **Roles** проекта **GuestBook**, далее выберите **Add** **|** **New Worker Role Project**.
  5. В диалоге **Add New Role Project** выберите категорию **Worker Role** и шаблон **Worker Role** для предпочитаемого языка (Visual C# или Visual Basic). Измените имя добавленной роли на **GuestBook\_WorkerRole** и нажмите **Add**.



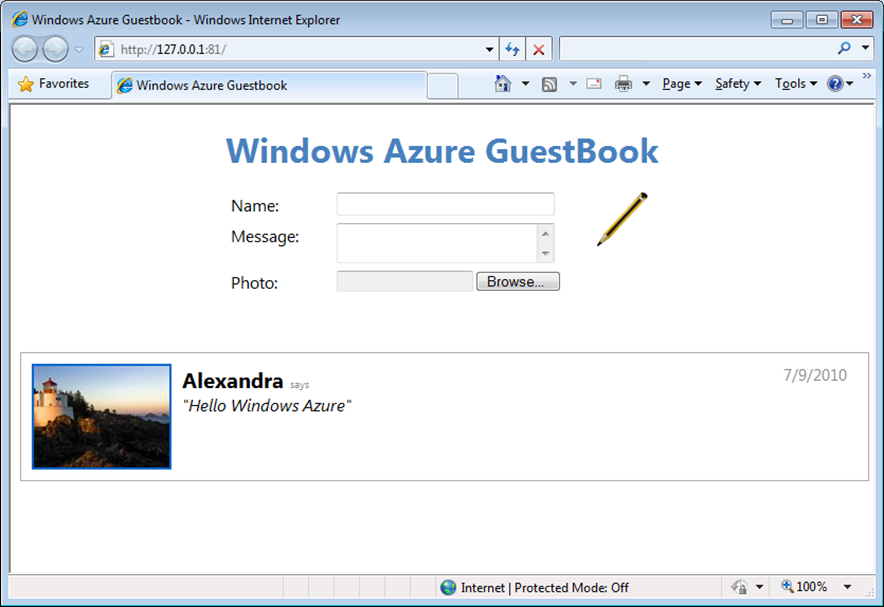
* + 1. Рисунок 23
    2. Добавление прикладной роли в решение (C#)



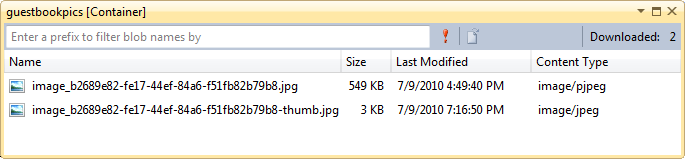
* + 1. Рисунок 24
    2. Добавление прикладной роли в решение (Visual Basic)
  1. В новом проекте добавьте ссылку на модель данных. Для этого в панели **Solution Explorer** нажмитеправой кнопкой на проекте **GuestBook\_WorkerRole**, выберите пункт **Add Reference**, переключитесь на закладку **Projects**, выберите **GuestBook\_Data** и нажмите кнопку **OK**.
  2. Теперь добавьте ссылку на сборку **System.Drawing**, для чего в диалоге **Add Reference** переключитесь на закладку **.NET**, выберите компонент **System.Drawing** и нажмите **OK**.
  3. Откройте файл **WorkerRole.cs** (для проекта на Visual C#) или **WorkerRole.vb** (для проекта на Visual Basic) из проекта **GuestBook\_WorkerRole** и добавьте объявления следующих пространств имен.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole Namespaces – CS*)
     2. C#
     3. **using System.Drawing;**
     4. **using System.Drawing.Drawing2D;**
     5. **using System.Drawing.Imaging;**
     6. **using System.IO;**
     7. **using GuestBook\_Data;**
     8. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole Namespaces – VB*)
     9. Visual Basic
     10. **Imports System.Drawing**
     11. **Imports System.Drawing.Drawing2D**
     12. **Imports System.Drawing.Imaging**
     13. **Imports System.IO**
     14. **Imports GuestBook\_Data**
  4. Добавьте в класс **WorkerRole** экземпляры классов для работы с двоичным хранилищем и очередью.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole Fields – CS*)
     2. C#
     3. public class WorkerRole : RoleEntryPoint
     4. {
     5. **private CloudQueue queue;**
     6. **private CloudBlobContainer container;**
     7. ...
     8. }
     9. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole Fields – VB*)
     10. Visual Basic
     11. Public Class WorkerRole
     12. Inherits RoleEntryPoint
     13. **Private queue As CloudQueue**
     14. **Private container As CloudBlobContainer**
     15. ...
     16. End Class
  5. Добавьте приведенный ниже фрагмент в метод **OnStart** перед вызовом метода **OnStart** базового класса.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole OnStart – CS*)
     2. C#
     3. public class WorkerRole : RoleEntryPoint
     4. {
     5. ...
     6. public override bool OnStart()
     7. {
     8. // Set the maximum number of concurrent connections
     9. ServicePointManager.DefaultConnectionLimit = 12;
     10. **// read storage account configuration settings**
     11. **CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =>**
     12. **{**
     13. **configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));**
     14. **});**
     15. **var storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     16. **// initialize blob storage**
     17. **CloudBlobClient blobStorage = storageAccount.CreateCloudBlobClient();**
     18. **container = blobStorage.GetContainerReference("guestbookpics");**
     19. **// initialize queue storage**
     20. **CloudQueueClient queueStorage = storageAccount.CreateCloudQueueClient();**
     21. **queue = queueStorage.GetQueueReference("guestthumbs");**
     22. **Trace.TraceInformation("Creating container and queue...");**
     23. **bool storageInitialized = false;**
     24. **while (!storageInitialized)**
     25. **{**
     26. **try**
     27. **{**
     28. **// create the blob container and allow public access**
     29. **container.CreateIfNotExist();**
     30. **var permissions = container.GetPermissions();**
     31. **permissions.PublicAccess = BlobContainerPublicAccessType.Container;**
     32. **container.SetPermissions(permissions);**
     33. **// create the message queue(s)**
     34. **queue.CreateIfNotExist();**
     35. **storageInitialized = true;**
     36. **}**
     37. **catch (StorageClientException e)**
     38. **{**
     39. **if (e.ErrorCode == StorageErrorCode.TransportError)**
     40. **{**
     41. **Trace.TraceError("Storage services initialization failure. "**
     42. **+ "Check your storage account configuration settings. If running locally, "**
     43. **+ "ensure that the Development Storage service is running. Message: '{0}'", e.Message);**
     44. **System.Threading.Thread.Sleep(5000);**
     45. **}**
     46. **else**
     47. **{**
     48. **throw;**
     49. **}**
     50. **}**
     51. **}**
     52. return base.OnStart();
     53. }
     54. ...
     55. }
     56. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole OnStart – VB*)
     57. Visual Basic
     58. Public Class WorkerRole
     59. Inherits RoleEntryPoint
     60. ...
     61. Public Overrides Function OnStart() As Boolean
     62. ' Set the maximum number of concurrent connections
     63. ServicePointManager.DefaultConnectionLimit = 12
     64. **' read storage account configuration settings**
     65. **CloudStorageAccount.SetConfigurationSettingPublisher(Function(configName, configSetter) configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))**
     66. **Dim storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
     67. **' initialize blob storage**
     68. **Dim blobStorage = storageAccount.CreateCloudBlobClient()**
     69. **container = blobStorage.GetContainerReference("guestbookpics")**
     70. **' initialize queue storage**
     71. **Dim queueStorage = storageAccount.CreateCloudQueueClient()**
     72. **queue = queueStorage.GetQueueReference("guestthumbs")**
     73. **Trace.TraceInformation("Creating container and queue...")**
     74. **Dim storageInitialized = False**
     75. **Do While (Not storageInitialized)**
     76. **Try**
     77. **' create the blob container and allow public access**
     78. **container.CreateIfNotExist()**
     79. **Dim permissions = container.GetPermissions()**
     80. **permissions.PublicAccess = BlobContainerPublicAccessType.Container**
     81. **container.SetPermissions(permissions)**
     82. **' create the message queue(s)**
     83. **queue.CreateIfNotExist()**
     84. **storageInitialized = True**
     85. **Catch e As StorageClientException**
     86. **If (e.ErrorCode = StorageErrorCode.TransportError) Then**
     87. **Trace.TraceError("Storage services initialization failure. " \_**
     88. **& "Check your storage account configuration settings. If running locally, " \_**
     89. **& "ensure that the Development Storage service is running. Message: '{0}'", e.Message)**
     90. **System.Threading.Thread.Sleep(5000)**
     91. **Else**
     92. **Throw**
     93. **End If**
     94. **End Try**
     95. **Loop**
     96. Return MyBase.OnStart()
     97. End Function
     98. ...
     99. End Class
  6. Замените тело метода **Run** приведенным ниже фрагментом.
     1. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole Run – CS*)
     2. C#
     3. public class WorkerRole : RoleEntryPoint
     4. {
     5. ...
     6. public override void Run()
     7. {
     8. **Trace.TraceInformation("Listening for queue messages...");**
     9. **while (true)**
     10. **{**
     11. **try**
     12. **{**
     13. **// retrieve a new message from the queue**
     14. **CloudQueueMessage msg = queue.GetMessage();**
     15. **if (msg != null)**
     16. **{**
     17. **// parse message retrieved from queue**
     18. **var messageParts = msg.AsString.Split(new char[] { ',' });**
     19. **var imageBlobUri = messageParts[0];**
     20. **var partitionKey = messageParts[1];**
     21. **var rowkey = messageParts[2];**
     22. **Trace.TraceInformation("Processing image in blob '{0}'.", imageBlobUri);**
     23. **string thumbnailBlobUri = System.Text.RegularExpressions.Regex.Replace(imageBlobUri, "([^\\.]+)(\\.[^\\.]+)?$", "$1-thumb$2");**
     24. **CloudBlob inputBlob = container.GetBlobReference(imageBlobUri);**
     25. **CloudBlob outputBlob = container.GetBlobReference(thumbnailBlobUri);**
     26. **using (BlobStream input = inputBlob.OpenRead())**
     27. **using (BlobStream output = outputBlob.OpenWrite())**
     28. **{**
     29. **ProcessImage(input, output);**
     30. **// commit the blob and set its properties**
     31. **output.Commit();**
     32. **outputBlob.Properties.ContentType = "image/jpeg";**
     33. **outputBlob.SetProperties();**
     34. **// update the entry in table storage to point to the thumbnail**
     35. **GuestBookDataSource ds = new GuestBookDataSource();**
     36. **ds.UpdateImageThumbnail(partitionKey, rowkey, thumbnailBlobUri);**
     37. **// remove message from queue**
     38. **queue.DeleteMessage(msg);**
     39. **Trace.TraceInformation("Generated thumbnail in blob '{0}'.", thumbnailBlobUri);**
     40. **}**
     41. **}**
     42. **else**
     43. **{**
     44. **System.Threading.Thread.Sleep(1000);**
     45. **}**
     46. **}**
     47. **catch (StorageClientException e)**
     48. **{**
     49. **Trace.TraceError("Exception when processing queue item. Message: '{0}'", e.Message);**
     50. **System.Threading.Thread.Sleep(5000);**
     51. **}**
     52. **}**
     53. }
     54. ...
     55. }
     56. (Фрагмент кода – *Introduction to Windows Azure - Ex2 WorkerRole Run – VB*)
     57. Visual Basic
     58. Public Class WorkerRole
     59. Inherits RoleEntryPoint
     60. ...
     61. Public Overrides Sub Run()
     62. **Trace.TraceInformation("Listening for queue messages...")**
     63. **Do**
     64. **Try**
     65. **' retrieve a new message from the queue**
     66. **Dim msg As CloudQueueMessage = queue.GetMessage()**
     67. **If msg IsNot Nothing Then**
     68. **' parse message retrieved from queue**
     69. **Dim messageParts = msg.AsString.Split(New Char() {","c})**
     70. **Dim imageBlobUri = messageParts(0)**
     71. **Dim partitionKey = messageParts(1)**
     72. **Dim rowKey = messageParts(2)**
     73. **Trace.TraceInformation("Processing image in blob '{0}'.", imageBlobUri)**
     74. **Dim thumbnailBlobUri As String = System.Text.RegularExpressions.Regex.Replace(imageBlobUri, "([^\\.]+)(\\.[^\\.]+)?$", "$1-thumb$2")**
     75. **' download original image from blob storage**
     76. **Dim inputBlob As CloudBlockBlob = container.GetBlockBlobReference(imageBlobUri)**
     77. **Dim outputBlob As CloudBlockBlob = container.GetBlockBlobReference(thumbnailBlobUri)**
     78. **Using input As BlobStream = inputBlob.OpenRead()**
     79. **Using output As BlobStream = outputBlob.OpenWrite()**
     80. **ProcessImage(input, output)**
     81. **' commit the blob and set its properties**
     82. **output.Commit()**
     83. **outputBlob.Properties.ContentType = "image/jpeg"**
     84. **outputBlob.SetProperties()**
     85. **' update the entry in table storage to point to the thumbnail**
     86. **Dim ds = New GuestBookDataSource()**
     87. **ds.UpdateImageThumbnail(partitionKey, rowKey, thumbnailBlobUri)**
     88. **' remove message from queue**
     89. **queue.DeleteMessage(msg)**
     90. **Trace.TraceInformation("Generated thumbnail in blob '{0}'.", thumbnailBlobUri)**
     91. **End Using**
     92. **End Using**
     93. **Else**
     94. **System.Threading.Thread.Sleep(1000)**
     95. **End If**
     96. **Catch e As StorageClientException**
     97. **Trace.TraceError("Exception when processing queue item. Message: '{0}'", e.Message)**
     98. **System.Threading.Thread.Sleep(5000)**
     99. **End Try**
     100. **Loop**
     101. End Sub
     102. ...
     103. End Class
  7. В завершение добавьте в класс **WorkerRole** метод, создающий миниатюру для указанного изображения.
     1. (Фрагмент кода – *Introduction to Windows Azure -* *Ex2 ProcessImage – CS*)
     2. C#
     3. public class WorkerRole : RoleEntryPoint
     4. {
     5. ...
     6. **public void ProcessImage(Stream input, Stream output)**
     7. **{**
     8. **int width;**
     9. **int height;**
     10. **var originalImage = new Bitmap(input);**
     11. **if (originalImage.Width > originalImage.Height)**
     12. **{**
     13. **width = 128;**
     14. **height = 128 \* originalImage.Height / originalImage.Width;**
     15. **}**
     16. **else**
     17. **{**
     18. **height = 128;**
     19. **width = 128 \* originalImage.Width / originalImage.Height;**
     20. **}**
     21. **var thumbnailImage = new Bitmap(width, height);**
     22. **using (Graphics graphics = Graphics.FromImage(thumbnailImage))**
     23. **{**
     24. **graphics.InterpolationMode = InterpolationMode.HighQualityBicubic;**
     25. **graphics.SmoothingMode = SmoothingMode.AntiAlias;**
     26. **graphics.PixelOffsetMode = PixelOffsetMode.HighQuality;**
     27. **graphics.DrawImage(originalImage, 0, 0, width, height);**
     28. **}**
     29. **thumbnailImage.Save(output, ImageFormat.Jpeg);**
     30. **}**
     31. }
     32. (Фрагмент кода – *Introduction to Windows Azure - Ex2 ProcessImage – VB*)
     33. Visual Basic
     34. Public Class WorkerRole
     35. Inherits RoleEntryPoint
     36. ...
     37. **Private Sub ProcessImage(ByVal input As Stream, ByVal output As Stream)**
     38. **Dim width As Integer**
     39. **Dim height As Integer**
     40. **Dim originalImage As New Bitmap(input)**
     41. **If originalImage.Width > originalImage.Height Then**
     42. **width = 128**
     43. **height = 128 \* originalImage.Height / originalImage.Width**
     44. **Else**
     45. **height = 128**
     46. **width = 128 \* originalImage.Width / originalImage.Height**
     47. **End If**
     48. **Dim thumbnailImage As New Bitmap(width, height)**
     49. **Using graphic = Graphics.FromImage(thumbnailImage)**
     50. **graphic.InterpolationMode = InterpolationMode.HighQualityBicubic**
     51. **graphic.SmoothingMode = SmoothingMode.AntiAlias**
     52. **graphic.PixelOffsetMode = PixelOffsetMode.HighQuality**
     53. **graphic.DrawImage(originalImage, 0, 0, width, height)**
     54. **End Using**
     55. **thumbnailImage.Save(output, ImageFormat.Jpeg)**
     56. **End Sub**
     57. End Class
     58. **Внимание:** в примере выше для простоты использованы типы из сборки System.Drawing. Данная сборка была разработана для использования в приложениях Windows Forms и не поддерживается другими типами приложений. Если вы решите использовать данную сборку в промышленных решениях для Windows Azure, тщательно протестируйте данную функциональность.
  8. Прикладная роль также использует сервисы хранения данных Windows Azure, поэтому вам потребуется добавить соответствующие настройки в файл конфигурации, точно так же, как это было сделано в веб-роли. Чтобы добавить настройку, разверните узел **Roles** проекта **GuestBook**, дважды щелкните на **GuestBook\_WorkerRole** и перейдите на закладку **Settings**. Нажмите **Add Setting**, дайте настройке имя “*DataConnectionString*”, измените **Type** на *ConnectionString*, и нажмите кнопку с многоточием. В диалоге **Storage Account Connection String** выберите вариант **Use the Windows Azure storage emulator** и нажмите **OK**. Нажмите **CTRL + S**, чтобы сохранить изменения.

## Проверка

* 1. Теперь локально запустим обновленное приложение для проверки новой функциональности. Проверим, что прикладная роль генерирует миниатюры для предъявляемых изображений.
  2. Нажмите **F5** для локального запуска приложения.
  3. Переключитесь в Internet Explorer. Если вы запускали приложение ранее, то увидите предыдущие записи, в том числе изображения в оригинальном размере. Одновременно с добавлением сущностей в табличное хранилище в очередь добавлялись сообщения, которые также никуда не делись.
  4. Подождите некоторое время, которое потребуется прикладной роли на обработку сообщений в очереди и создание миниатюр. После завершения обработки изображений и обновления страницы вы увидите вместо оригинального изображения его миниатюру.



* + 1. Рисунок 25
    2. Страница, отображающая сгенерированную прикладной ролью миниатюру
  1. Если вы работаете с Visual Studio 2010, раскройте узел **Windows Azure Storage\Blobs** в панели **Server Explorer** и дважды щелкните на контейнере *guestbookpics*. Теперь контейнер содержит дополнительные объекты, соответствующие созданным миниатюрам.



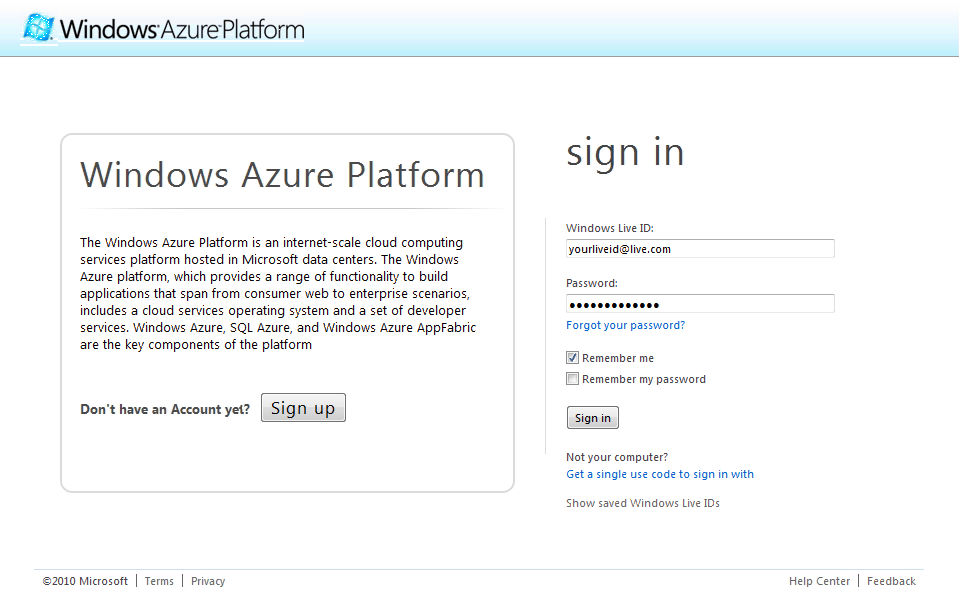
* + 1. Рисунок 26
    2. Контейнер, содержащий оригинальное изображение и созданную миниатюру
  1. Добавьте еще несколько записей в гостевую книгу. Убедитесь в том, что через несколько секунд записи обновляются, что свидетельствует об успешной работе фонового процесса.
  2. Нажмите **SHIFT + F5** для остановки процесса отладки.

# Упражнение 3: развертывание приложения в Windows Azure

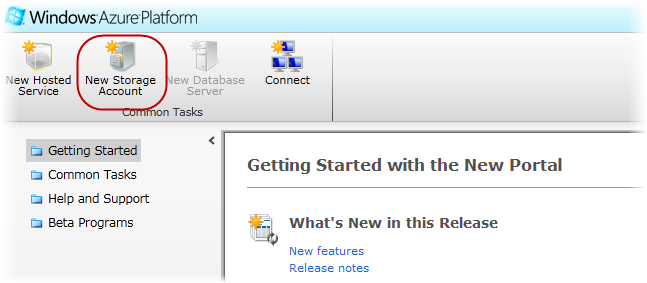
* 1. В этом упражнении вы развернете созданное приложение в Windows Azure с использованием административного портала. Сначала вы создадите необходимые компоненты на стороне Windows Azure, затем загрузите туда созданное приложение и сконфигурируете его. После проверки работоспособности приложения в тестовой области (staging area) вы переместите приложение в промышленную среду.
  2. **Внимание:** чтобывыполнить описанные упражнения вам потребуется учетная запись и подписка Windows Azure.

## Задача 1 – создание сервиса хранения данных и вычислительного сервиса

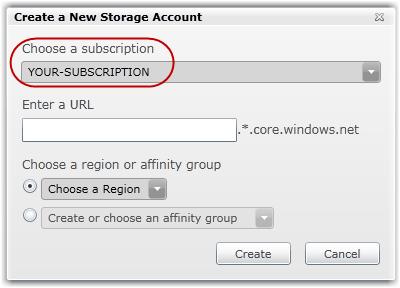
* 1. Развертываемое приложение использует как вычислительные ресурсы, так и хранилище данных. В этой задаче вы создадите учетную запись хранилища, что позволит приложению хранить данные. Дополнительно вам потребуется вычислительный сервис, предназначенный для выполнения кода приложения.
  2. Откройте браузер и перейдите на страницу [http://windows.azure.com](http://windows.azure.com/). Введите реквизиты учетной записи Windows Live ID, связанной с учетной записью Windows Azure.



* + 1. Рисунок 27
    2. Вход на административный портал Windows Azure
  1. Создайте учетную запись хранилища данных. Нажмите кнопку **New Storage Account** напанели инструментов.



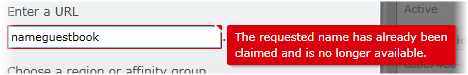
* + 1. Рисунок 28
    2. Создание учетной записи хранилища
  1. В диалоге **Create a New Storage Account** выберите подписку в раскрывающемся списке **Choose a subscription**.



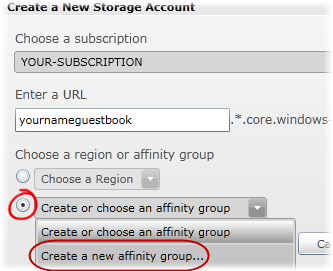
* + 1. Рисунок 29
    2. Выбор подписки для сервиса хранения данных
  1. В текстовом поле **Enter a URL** введите имя создаваемого сервиса, например ***<yourname>guestbook***, где *<yourname>* - уникальное имя. Windows Azure использует это значение для генерации URL конечных точек сервиса.



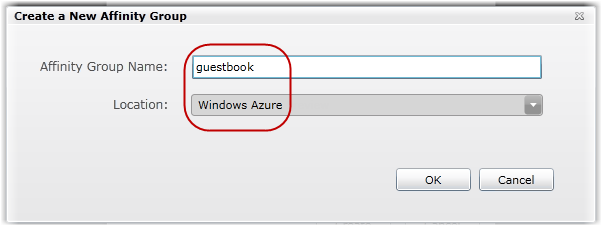
* + 1. Рисунок 30
    2. Указание URL для создаваемого хранилища
    3. **Внимание:** Windows Azure требует уникальности создаваемого имени и соответствия правилам формирования URL. В противном случае вы увидите сообщение о недопустимости введенных данных.



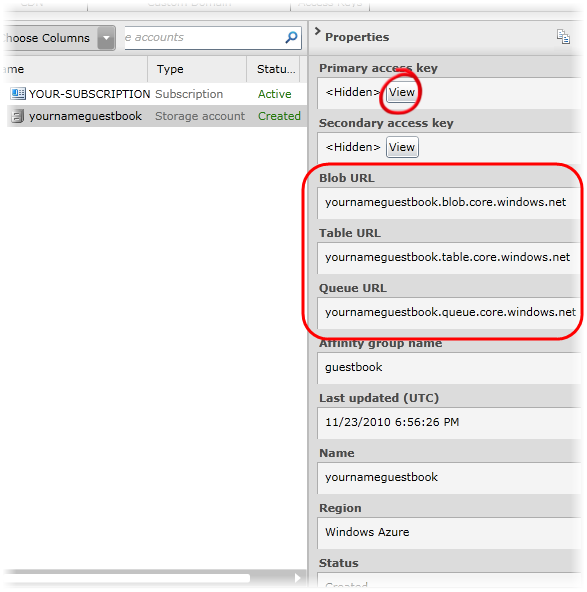
* 1. Выберите вариант **Create or choose an affinity group**, затем из раскрывающегося списка значение **Create a new affinity group**.



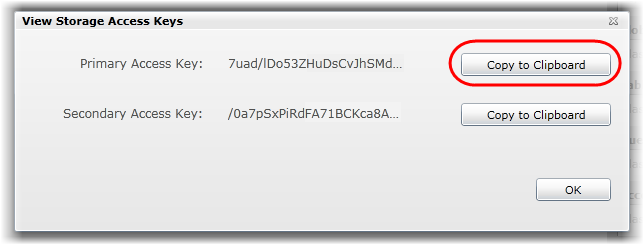
* + 1. Рисунок 31
    2. Выбор варианта создания новой географической группы
    3. **Внимание:** необходимость создания новой географической группы диктуется целесообразностью размещения сервиса хранения данных и вычислительного в одном ЦОДе.
  1. В диалоге **Create a New Affinity Group** введите в поле **Affinity Group Name** имя “guestbook”, выберите местоположение (**Location**) и нажмите кнопку **OK**.



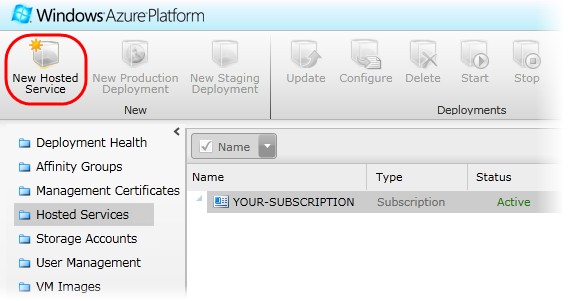
* + 1. Рисунок 32
    2. Создание новой географической группы
  1. Вернувшись в диалог **Create a New Storage Account**, нажмите кнопку **Create** для создания хранилища. Дождитесь завершения процесса создания и обновления списка **Storage Accounts**. Убедитесь в том, что панель **Properties** отображает соответствующие каждому из сервисов **URL**. Запишите имя учетной записи – первый фрагмент назначенного конечным точкам адреса.



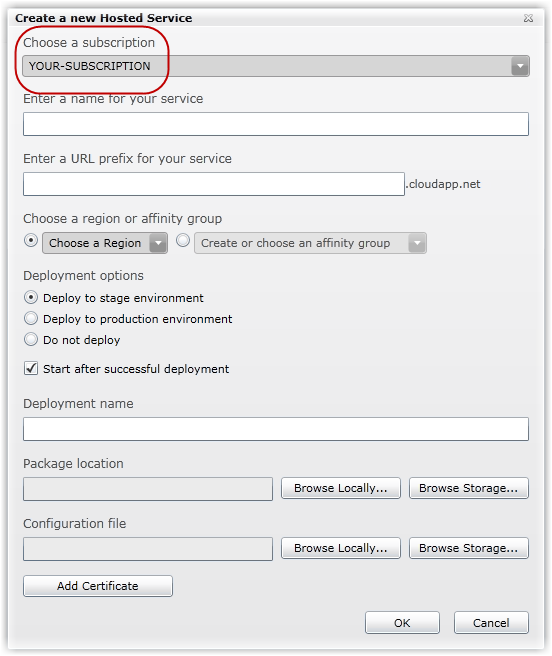
* + 1. Рисунок 33
    2. Учетная запись хранилища данных успешно создана
  1. Теперь нажмите кнопку **View** напротив поля **Primary access key** на панели **Properties**. В диалоге **View Storage Access Keys**, нажмите кнопку **Copy to Clipboard** напротив поля **Primary Access Key**. Это значение потребуется позже для конфигурирования приложения.



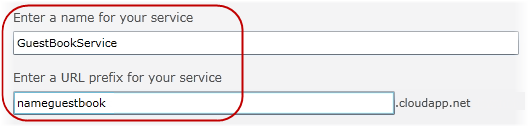
* + 1. Рисунок 34
    2. Получение ключа доступа
    3. **Внимание:** ключи **Primary Access Key** и **Secondary** **Access** **Key** позволяют получить доступ к хранилищу. Второй (secondary) ключ дает те же права доступа и используется как дублирующий. Вы можете заменить любой из ключей, если он был скомпрометирован.
  1. Теперь создайте вычислительный сервис для запуска исполняемого кода приложения. Выберите раздел **Hosted Services** в левой панели. Нажмите кнопку **New Hosted Service** на панели инструментов.



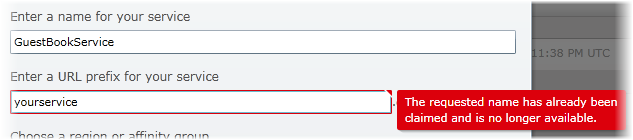
* + 1. Рисунок 35
    2. Создание вычислительного сервиса
  1. В диалоге **Create a new Hosted Service** выберите из раскрывающегося списка **Choose a subscription** необходимую подписку.



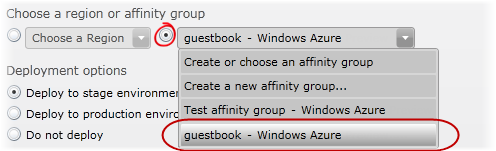
* + 1. Рисунок 36
    2. Выбор подписки
  1. Введите имя сервиса в текстовое поле **Enter a name for your service** и выберите для него URL, введя его префикс в текстовое поле **Enter a URL prefix for your service**, например, ***<yourname>guestbook***, где *<yourname>* - уникальное имя. Windows Azure использует это значение для генерации URL конечных точек сервиса.



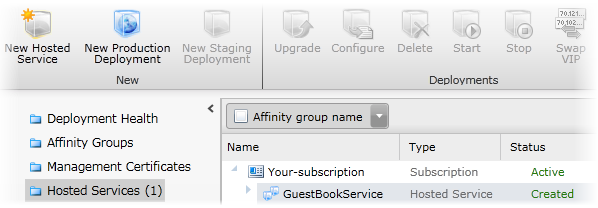
* + 1. Рисунок 37
    2. Указание имени сервиса и URL
    3. **Внимание:** есливозможно, укажите одно и то же имя для хранилища данных и вычислительного сервиса. Тем не менее, ничто не мешает выбрать различные имена, если одно из них уже занято.
    4. **Внимание:** Windows Azure требует уникальности создаваемого имени и соответствия правилам формирования URL. В противном случае вы увидите сообщение о недопустимости введенных данных.



* 1. Укажите вариант **Create or choose an affinity group** и выберите из раскрывающегося списка созданную ранее группу **guestbook**.



* + 1. Рисунок 38
    2. Выбор созданной ранее географической группы
    3. **Внимание:** выбрав из списка группу **guestbook**, вы заставите фабрику развернуть вычислительный сервис в том же ЦОДе, что и хранилище данных.
  1. Выберите вариант **Do not Deploy**.
     1. **Внимание:** создание сервиса и развертывание приложения в Windows Azure может быть выполнено в одной операции (группа свойств **Deployment Options**), но в этой лабораторной работы вы выполните развертывание отдельно чуть позже.
  2. Нажмите кнопку **OK**, чтобы создать сервис и дождитесь завершения операции.



* + 1. Рисунок 39
    2. Вычислительный сервис успешно создан
  1. Не закрывайте окно браузера. Портал потребуется вам для выполнения следующей задачи.

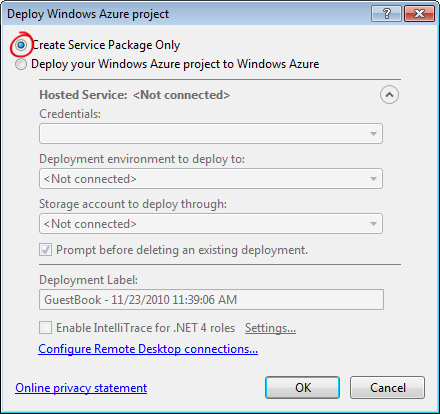


## Задача 2 – развертывание приложения с помощью портала Windows Azure

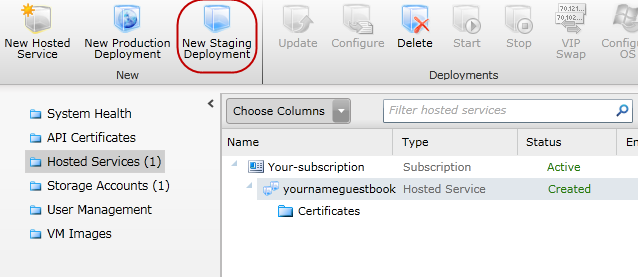
* 1. Существует несколько способов развернуть приложение в Windows Azure. Так, набор средств Windows Azure Tools for Visual Studio позволяет создавать и развертывать пакеты непосредственно из Visual Studio. Другой вариант развертывания связан с использованием сценариев [Windows Azure Service Management PowerShell Cmdlets](http://code.msdn.microsoft.com/azurecmdlets) и подходит для автоматизации данной процедуры. Наконец, административный портал Windows Azure предоставляет веб-интерфейс, позволяющий выполнить развертывание с помощью браузера.
  2. В этой задаче вы развернете приложение в тестовой (staging) среде с использованием портала, но сначала потребуется сгенерировать пакет средствами Visual Studio.
  3. Запустите Visual Studioот имени администратора, если это не было сделано ранее: выберите пункт меню **Start | All Programs | Microsoft Visual Studio 2010**. Щелкните правой кнопкой на ярлыке **Microsoft Visual Studio 2010** и выберите элемент **Run as administrator**.
  4. В случае появления диалога **User Account Control** нажмите кнопку **Continue**.
  5. В меню **File** выберите **Open** **|** **Project/Solution**. В диалоге **Open Project** перейдите в подкаталог **Source\Ex3-WindowsAzureDeployment\Begin** в папке с материалами к работе и укажите файл **Begin.sln** в соответствующем предпочитаемому языку каталоге (Visual C# или Visual Basic), затем нажмите **Open**. Вы также можете продолжить работу с решением, полученным после завершения предыдущего шага.
  6. Чтобы настроить хранилище, откройте находящийся в проекте **GuestBook** файл **ServiceConfiguration.cscfg**. Замените символы [YOUR\_ACCOUNT\_NAME] именем учетной записи хранилища, которую вы создали в задаче 1. Если вы следовали рекомендации, то имя записано в формате ***<yourname>guestbook***, где *<yourname>* - уникальное имя. Выполните замену в двух местах – в строке подключения *DataConnectionString* и в строке *Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString*.
  7. Теперь замените символы [YOUR\_ACCOUNT\_KEY] значением ключа доступа **Primary Access Key**, также созданным в задаче 1. Аналогично предыдущему пункту, выполните замену в двух местах.



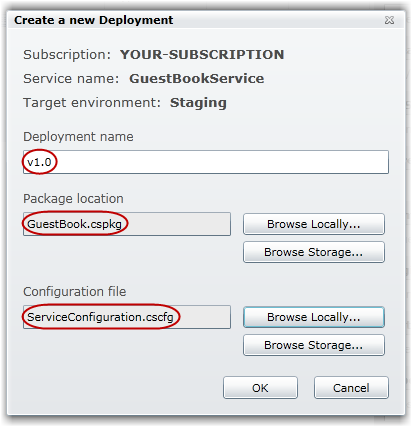
* + 1. Рисунок 40
    2. Настройка строк подключения к хранилищу данных
  1. Сформируйте пакет для развертывания. Для этого нажмите правой кнопкой на проекте **GuestBook** и выберите пункт **Publish**. В диалоге **Deploy Windows Azure project** выберите вариант **Create Service Package Only** и нажмите кнопку **OK**.
     1. После завершения построения проекта и генерации пакета откроется окно Windows Explorer, отображающего содержимое каталога с собранным пакетом.



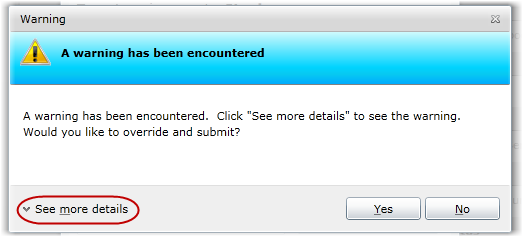
* + 1. Рисунок 41
    2. Создание пакета для развертывания в Visual Studio
    3. **Внимание:** хотя процедура развертывание непосредственно из Visual Studio здесь не описана, для этого потребуется указать реквизиты использующей сертификат учетной записи. Сертификат необходимо предварительно загрузить на портал.
  1. Переключитесь обратно в административный портал.
  2. Выберите созданный на предыдущем шаге вычислительный сервис и нажмите кнопку **New Staging Deployment** на панели инструментов.
     1. **Внимание:** вычислительный сервис используется для запуска вашего кода в среде Windows Azure. В нем присутствуют два различных слота – тестовый (staging) и промышленный (production). Тестовый слот позволяет выполнить проверку работоспособности приложения до передачи в эксплуатацию.



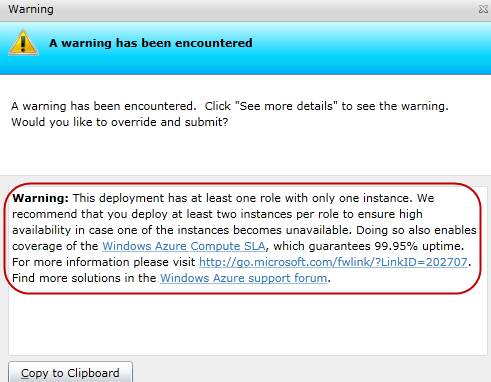
* + 1. Рисунок 42
    2. Сводная информация о вычислительном сервисе
  1. В диалоге **Create a new Deployment** для выбора местоположения пакета (**Package location**) нажмите на кнопку **Browse Locally**, перейдите в каталог с пакетом и выберите **GuestBook.cspkg**.
  2. Теперь выберите расположение файла конфигурации (поле **Configuration File**) – нажмите кнопку **Browse Locally** и выберите файл **ServiceConfiguration.cscfg** из того же каталога, что и в предыдущем шаге.
     1. **Внимание:** файл *.cscfg* содержит настройки приложения, в том числе количество экземпляров.
  3. Укажите название развертывания (поле **Deployment name**) – строку, позволяющую идентифицировать его, например, **v1.0**.
     1. **Внимание:** портал отображает название развертывания в тестовой и промышленной средах, что позволяет отличать версии.



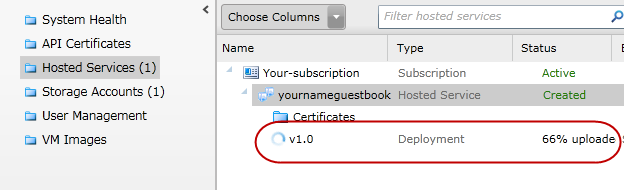
* + 1. Рисунок 43
    2. Настройка свойств развертывания
  1. Нажмите кнопку **OK** для начала развертывания. Убедитесь в том, что портал отобразил диалог с предупреждением. Нажмите **See more details** для просмотра сообщения.



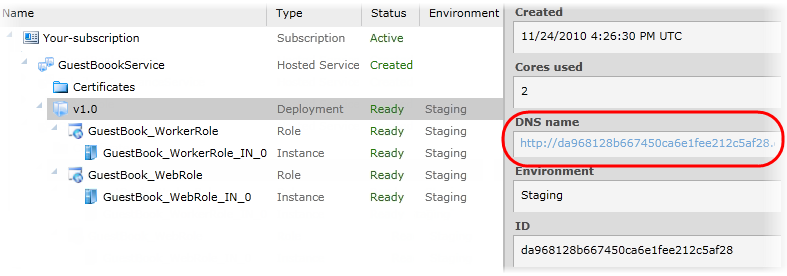
* + 1. Рисунок 44
    2. Просмотр предупреждений
    3. **Внимание:** в данном случае предупреждение связано с развертыванием ролей в единственном экземпляре. Развертывание менее двух экземпляров не позволяет гарантировать высокую доступность сервиса. В следующей задаче вы увеличите количество экземпляров, чтобы устранить данную проблему.



* 1. Нажмите **Yes** чтобы продолжить развертывание. Убедитесь в том, что пакет начал загружаться, для чего на портале отобразился индикатор состояния.



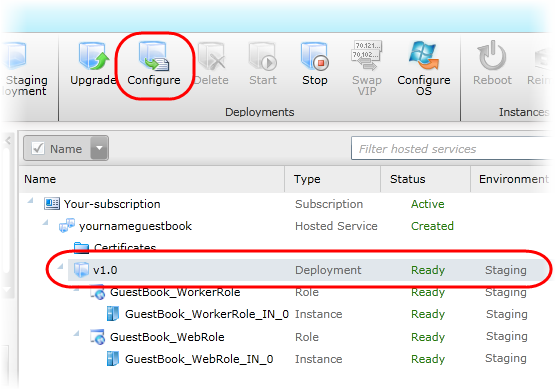
* + 1. Рисунок 45
    2. Загрузка пакета на портал Windows Azure
  1. Дождитесь завершения развертывания, что может занять несколько минут. Служба развернута и находится в состоянии **Ready**. Убедитесь в том, что портал назначил службе **DNS**, представленный в форме уникального идентификатора. Если коротко – это тестовый URL для доступа к службе.
     1. **Внимание:** в процессе развертывания в Windows Azure на основе информации из конфигурационного файла сервис был развернут на необходимом числе машин. Одновременно настраиваются балансировщики нагрузки и необходимое сетевое оборудование.



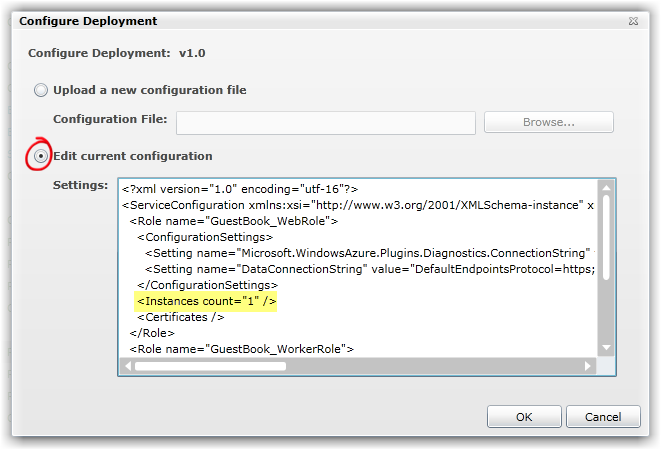
* + 1. Рисунок 46
    2. Пакет успешно развернут и готов к работе

## Задача 3 – изменение числа экземпляров роли

* 1. Перед началом тестирования приложения вы сконфигурируете его.
  2. В разделе **Hosted Services** выберите сервис GuestBook и нажмите кнопку **Configure**.



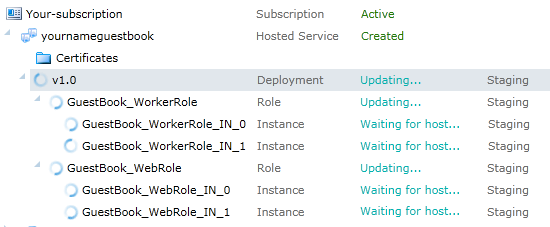
* + 1. Рисунок 47
    2. Настройка приложения
  1. В диалоге **Configure Deployment** выберите вариант **Edit current configuration**, найдите внутри конфигурации роли **GuestBook\_WebRole** элемент **Instances** и измените значение атрибута **count** на *2*. Выполните такую же операцию для прикладной роли **GuestBook\_WorkerRole**.



* + 1. Рисунок 48
    2. Настройка числа экземпляров

**Внимание:** настройки представлены в виде XML документа. Его начальные значения содержатся в файле **ServiceConfiguration.cscfg**, использованном при развертывании.

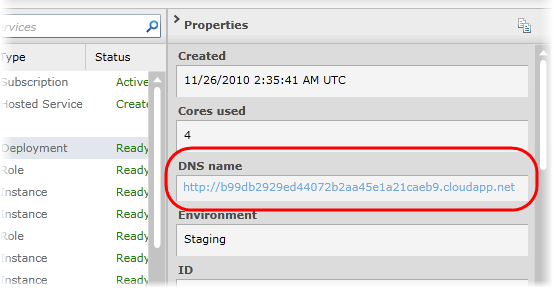
* + 1. **Внимание:** настройка **Instances** управляет количеством экземпляров каждой роли. Для отдельных видов подписки максимальное количество может быть ограничено, например, двумя. Для коммерческих подписок можно установить любое количество экземпляров, за которые вы готовы платить.
  1. Нажмите кнопку **OK** для обновления настроек и дождитесь их вступления в силу.



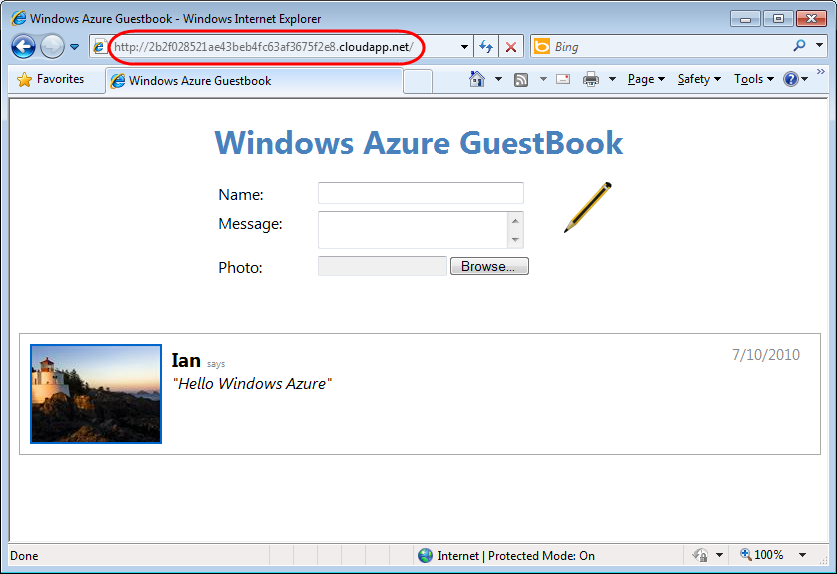
* + 1. Рисунок 49
    2. Изменение числа экземпляров
    3. **Внимание:** во время обновления настроек отображается статус "Updating deployment...".

## Задача 4 – проверка приложения в тестовой среде

* 1. В этой задаче вы запустите приложение в тестовой среде и проверите его работоспособность.
  2. В разделе **Hosted Services** выберите сервис GuestBook и нажмите на ссылке **DNS name** в правой панели**.**



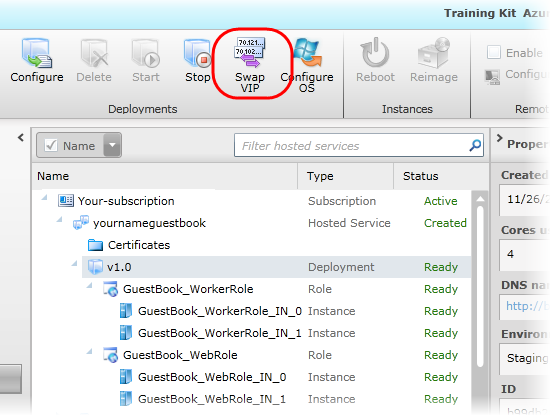
* + 1. Рисунок 50
    2. Запуск приложения для тестирования
    3. **Внимание:** ссылка в поле DNS name образована по правилу *<guid>.cloudapp.net*, где *<guid>* - уникальный идентификатор. Это позволяет разделить тестовую и промышленную среды – других отличий между ними нет.
  1. Если хотите, вы можете протестировать приложение и добавить запись в гостевую книгу.



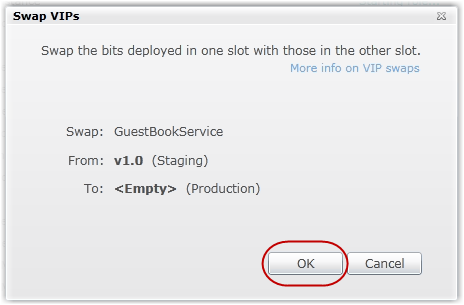
* + 1. Рисунок 51
    2. Приложение в тестовой среде

## Задача 5 – передача приложения в промышленную среду

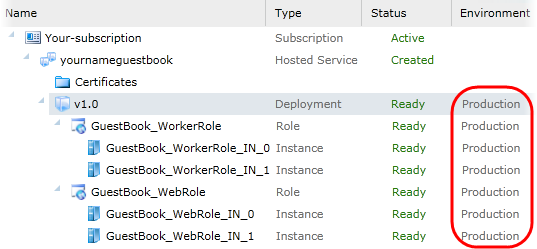
* 1. Теперь, когда вы убедились в работоспособности сервиса, вы можете передать его в промышленную эксплуатацию. Этот процесс связан с реконфигурированием балансировщиков нагрузки, после чего приложение становится доступным по «реальному» адресу.
  2. В разделе **Hosted Services** выберите сервис GuestBook и нажмите кнопку **Swap VIP** на панели инструментов.



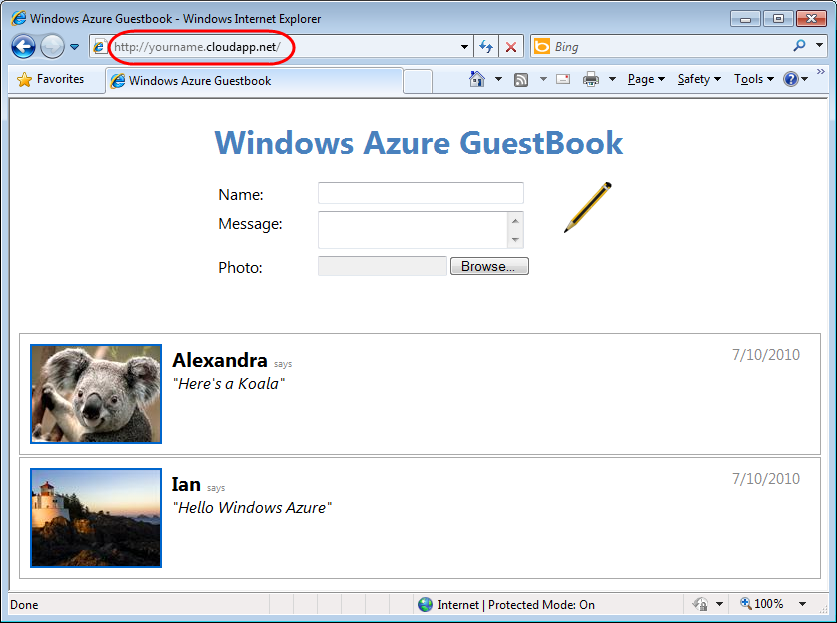
* + 1. Рисунок 52
    2. Перевод приложения в промышленный слот
  1. В диалоге **Swap VIPs** нажмите кнопку **OK** для замены тестовой и промышленной сред.



* + 1. Рисунок 53
    2. Перевод приложения в промышленный слот
  1. Дождитесь завершения процесса. Обычно на это требуется несколько секунд.



* + 1. Рисунок 54
    2. Приложение переведено в промышленную эксплуатацию
  1. Нажмите ссылку **DNS name** для открытия браузера и убедитесь в том, что адрес соответствует промышленному слоту.



* + 1. Рисунок 55
    2. Приложение запущено в промышленной среде
    3. **Внимание:** иногда перевод приложения в промышленный слот занимает некоторое время. Обращение к сервису в этот момент может привести к возникновению ошибки DNS (404).
  1. Даже когда сервис находится в остановленном состоянии (suspended), он потребляет ресурсы, а вы платите за них. После завершения тестирования не забывайте удалить неиспользуемые ресурсы. Чтобы удалить сервис, перейдите в раздел **Hosted Services**, выберите тип развертывания (staging или production)и остановите его, нажав кнопку **Stop** на панели инструментов. После останова сервиса нажмите кнопку **Delete**.

# Результат

1. В ходе данной работы вы познакомились с основными типами сервисов Windows Azure. Вы увидели, что сервис состоит из одной или более ролей; познакомились с сервисами хранения данных Windows Azure (Windows Azure storage services) – хранилищем двоичных объектов, табличным хранилищем и очередями. Наконец, вы рассмотрели один из базовых шаблонов построения приложений для Windows Azure – организацией взаимодействия пользовательского интерфейса и фоновых служб на основе очередей.