* 1. 

Практическая лабораторная работа

Изучая хранилище Windows Azure

Версия: 2.0.0

Обновлено: 1/6/2012



содержание

[обзор 3](#_Toc310346223)

[упражнение 1: работа с таблицами 5](#_Toc310346224)

[Задание 1 – Настройка аккаунта хранилища 5](#_Toc310346225)

[Задание 2 – Создание классов для схемы таблицы 9](#_Toc310346226)

[Задание 3 –Создание пользовательского интерфейса для чата 13](#_Toc310346227)

[Проверка 17](#_Toc310346228)

[упражнение 2: работа с блобами 19](#_Toc310346229)

[Задание 1 – Получение содержимого блоба из хранилища 19](#_Toc310346230)

[Задание 2 – Загрузка блобов в хранилище 28](#_Toc310346231)

[Задание 3 – Получение из хранилища метаданных для блобов 33](#_Toc310346232)

[Задание 4 – Удаление блобов из хранилища 37](#_Toc310346233)

[Задание 5 – Копирование блобов 40](#_Toc310346234)

[Задание 6 – Создание снапшотов блобов 44](#_Toc310346235)

[упражнение 3: работа с очередями 48](#_Toc310346236)

[Задание 1 – Создание решения 48](#_Toc310346237)

[Задание 2 – Помещение сообщений в очередь 52](#_Toc310346238)

[Задание 3 – Получение сообщений из очереди 56](#_Toc310346239)

[Проверка 61](#_Toc310346240)

[упражнение 4: работа с дисками 63](#_Toc310346241)

[Задание 1 – Приложение PhotoAlbum 64](#_Toc310346242)

[Задание 2 –Перенос приложения в Windows Azure с использованием дисков 66](#_Toc310346243)

[Задание 3 –Создание нового диска в «облаке» 80](#_Toc310346244)

[Задание 4 –Создание NTFS VHD локально 89](#_Toc310346245)

[Задание 5 – Развертывание приложения и загрузка диска в Windows Azure 100](#_Toc310346246)

[заключение 108](#_Toc310346247)

Обзор

* 1. Сервисы хранилища предоставляют долговечное надёжное хранилище в эмуляторе вычислений Windows Azure и включают в себя сервисы блобов, таблиц и очередей. Кроме этого, с использованием дисков ваши приложения, запущенные в Windows Azure, могут использовать существующие NTFS API для доступа к долговечному надежному диску, основанному на использовании хранилища блобов.
  2. В этой лабораторной вы рассмотрите основы работы с хранилищем Windows Azure в локальном эмуляторе вычислений и узнаете о некоторой функциональности, доступной разработчикам.

# Цели

* 1. В этой практической лабораторной работе вы научитесь:
  + Использовать сервис таблиц
  + Использовать сервис блобов
  + Использовать сервис очередей
  + Создавать и получать метаданные
  + Использовать диски Windows Azure

# Требования

* 1. Для выполнения лабораторной работы необходимо иметь следующее установленное ПО:
  + [Microsoft .NET Framework 4.0](http://go.microsoft.com/fwlink/?linkid=186916)
  + [Microsoft Visual Studio 2010](http://msdn.microsoft.com/vstudio/products/)
  + [Windows Azure Tools for Microsoft Visual Studio 1.6](http://www.microsoft.com/windowsazure/sdk/)
  + IIS 7 (с ASP.NET, WCF HTTP Activation, Tracing)

* + [SQL Server 2005 Express Edition (or later)](http://www.microsoft.com/sqlserver)
  + [Windows Azure Tools for Microsoft Visual Studio 1.6](http://www.microsoft.com/windowsazure/sdk/)

# Настройка

* 1. Для выполнения упражнений вы должны соответствующим образом настроить среду.
  2. Откройте Windows Explorer и перейдите в папку **Source**.
  3. Щелкните два раза на файле **Setup.cmd** для запуска процесса настройки, который настроит среду и установит необходимые сниппеты кода Visual Studio.
  4. Если во время настройки появится диалоговое окно User Account Control, нажмите OK.

**Примечание: Перед запуском скрипта настройки убедитесь, что у вас установлено всё, что необходимо для лабораторной.**

# Использование сниппетов

* 1. В лабораторной вам необходимо будет вставлять блоки кода. Для вашего удобства большинство кода предоставлено в виде сниппетов Visual Studio, которые вы можете использовать в Visual Studio 2010 во избежание ручного ввода кода.
  2. Если вы не знакомы со сниппетами Visual Studio и хотите научиться работе с ними, вы можете прочесть документ **Setup.docx** в папке **Assets**, содержащий секцию с информацией о работе со сниппетами.

# Упражнения

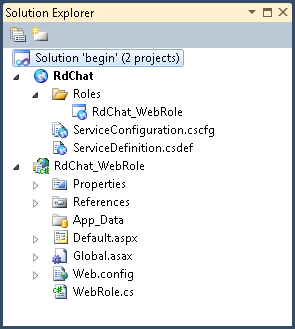
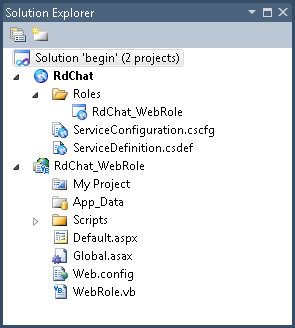
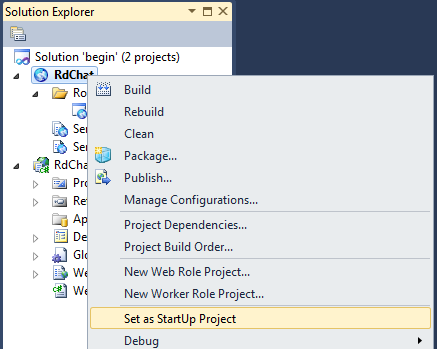
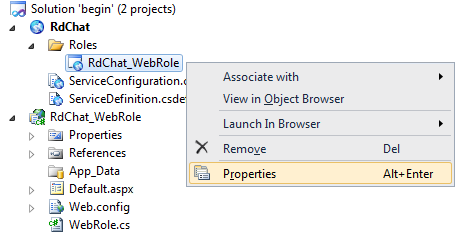
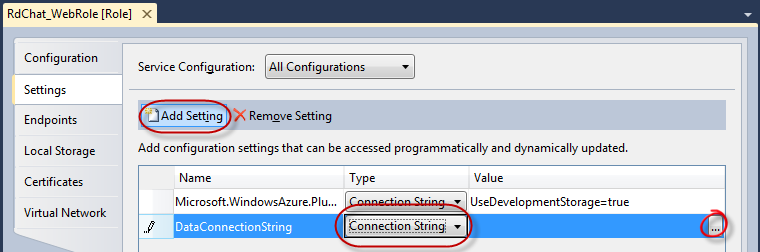
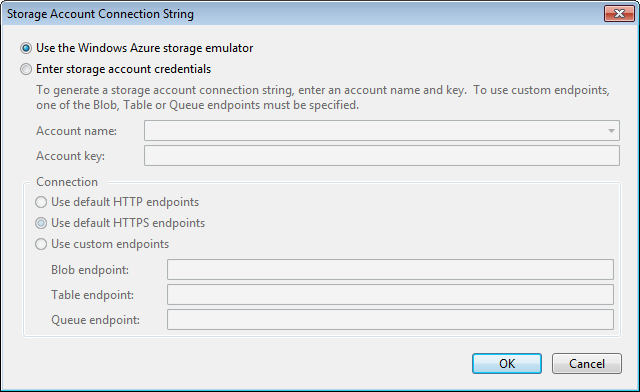
* 1. Лабораторная работа включает в себя упражнения по следующим темам:
  2. Работа с таблицами
  3. Работа с блобами
  4. Работа с очередями
  5. Работа с дисками
  6. Примерное время на выполнение лабораторной работы: **90 минут**.
  7. **Примечание:** При первом запуске Visual Studio вы должны выбрать один из предустановленных наборов настроек. Каждый набор настроек спроектирован с целью использования с различными стилями разработки и определяет расположение окон, поведение редактора, сниппеты кода IntelliSense и опции диалоговых окон. Данная лабораторная подразумевает использование набора настроек **General Development Settings**. Обратите внимание: если вы выберите другой набор настроек, могут быть различия с представленными описаниями.

Упражнение 1: Работа с таблицами

* 1. В этом упражнении вы воспользуетесь Windows Azure Table Service API для создания простого приложения, хранящего и получающего данные из структурированного хранилища и являющегося простым веб-приложением чата с возможностями сохранения, получения и отображения сообщений из таблицы Windows Azure.
  2. Таблицы Windows Azure хранят данные в виде коллекций сущностей. Сущности аналогичны записям. Сущность имеет первичный ключ и набор свойств. Свойства являются парами ключ-значения, что аналогично столбцам.
  3. Для доступа к сервису таблиц вы можете использовать REST API, совместимый с [WCF Data Services](http://msdn.microsoft.com/en-us/library/cc668792.aspx) (ранее ADO.NET Data Services Framework). Для чтения и записи данных в сервис таблиц в упражнении используется [WCF Data Services Client Library](http://msdn.microsoft.com/en-us/library/cc668772.aspx) (ранее .NET Client Library).

Задание 1 – Настройка аккаунта хранилища

В этом задании вы настроите необходимую для установления подключения к сервису таблиц конфигурацию.

* 1. Откройте Microsoft Visual Studio 2010 с повышенными привилегиями. Чтобы сделать это, в **Start** | **All Programs** | **Microsoft Visual Studio 2010** нажмите правой кнопкой мыши на ярлыке **Microsoft Visual Studio 2010** и выберите **Run as Administrator**.
  2. В меню **File** выберите **Open** и **Project/Solution**. В диалоговом окне **Open Project** перейдите в папке **\Source\Ex01-WorkingWithTables\Begin** в папке лабораторной **Source**, выберите **Begin.sln,** исходя из используемого вами языка программирования (Visual C# или Visual Basic) и нажмите **Open**.
     1. 
     2. Изображение
     3. Приложение чата Windows Azure в Solution Explorer (C#)
     4. 
     5. Изображение
     6. Приложение чата Windows Azure в Solution Explorer (VB)
     7. **Примечание:** Решение содержит проект с Web-ролью Windows Azure.
  3. Убедитесь, что **RdChat** является проектом, запускаемым по умолчанию, щелкнув правой кнопкой мыши на нем в **Solution Explorer** и нажав **Set as StartUp Project**.
     1. 
     2. Изображение
     3. Настройка запуска проекта
  4. Измените в описании сервиса настройки конфигурации, необходимые для доступа к сервису таблиц Windows Azure: откройте в проекте **RdChat** папку **Roles**, щелкните правой кнопкой мыши на **RdChat\_WebRole** и выберите **Properties**.
     1. 
     2. Изображение
     3. Запуск редактора конфигурации сервиса
  5. Перейдите на вкладку **Settings**, нажмите **Add Setting** и создайте новую настройку конфигурации с именем *DataConnectionString*. Укажите её тип как **Connection String**, после чего нажмите на кнопку с троеточием и настройте строку подключения к хранилищу на использование локального эмулятора хранилища - *Use storage emulator*.
     1. 
     2. Изображение
     3. Создание строки подключения к хранилищу
     4. 
     5. Изображение
     6. Настройки строки подключения к хранилищу на использование локального эмулятора хранилища
  6. Нажмите **CTRL-S** для сохранения изменений.
     1. **Примечание:** Библиотека StorageClient использует эти настройки для доступа к хранилищу Windows Azure.
     2. **DataConnectionString**: Это строка подключения к аккаунту Windows Azure, программно используя которую, мы можем осуществить доступ к хранилищу данных и другой функциональности Windows Azure. Эта строка подключения может использовать либо аккаунт Windows Azure в «облаке», либо локальный эмулятор.
     3. **Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString**: Это строка подключения к серверу Windows Azure, аналогичная *DataConnectionString*, но отведённая под нужды диагностики.

Задание 2 – Создание классов для схемы таблицы

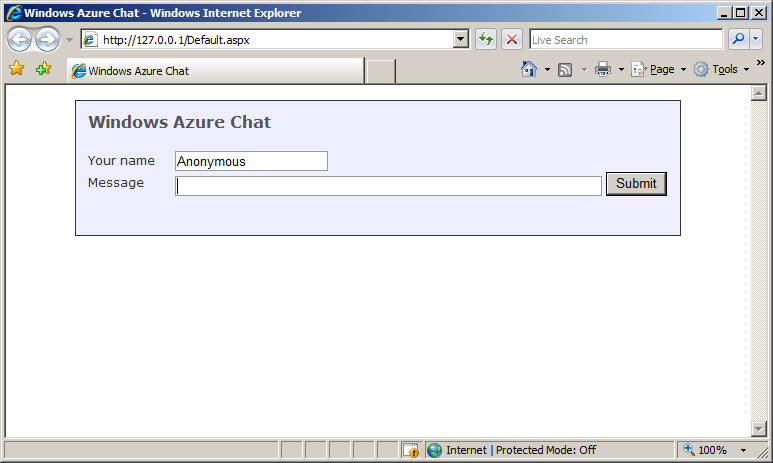
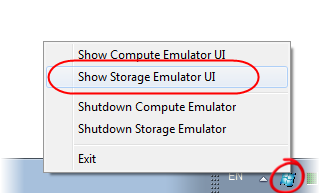
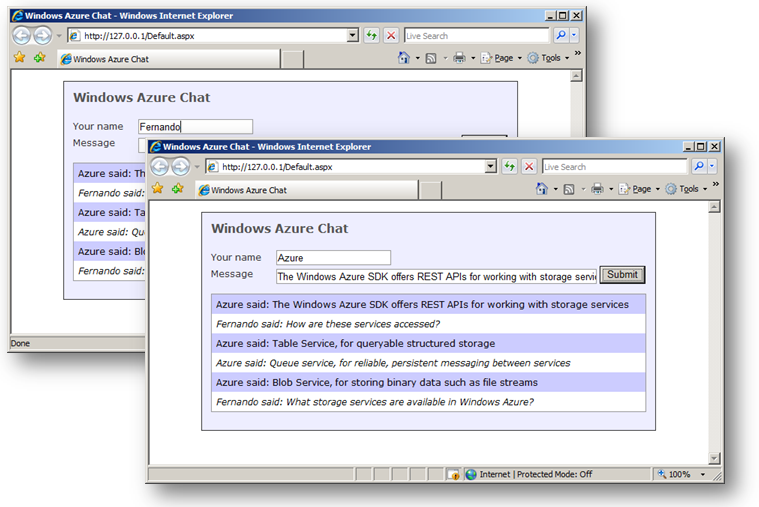
* 1. При локальной работе с сервисом таблиц в эмуляторе хранилища вы будете использовать клиентскую библиотеку WCF Data Services.
  2. Для использования сервиса таблицы Windows Azure в .NET вы создадите класс-модель необходимой схемы. Кроме свойств модели, класс должен содержать также свойства **Timestamp**, **PartitionKey** и **RowKey** и должен иметь особый атрибут **DataServiceKey**(“*PartitionKey”*, “*RowKey”*). Для упрощения работы пространство имен **Microsoft.WindowsAzure.StorageClient** включает класс **TableServiceEntity,** определяющий обязательные свойства и атрибуты и от которого можно легко наследовать ваш собственный класс.
  3. В этом задании вы создадите модель, в которой будут сохраняться данные приложения.
  4. Добавьте в проект Web-роли ссылку на клиентскую библиотеку WCF Data Services: щелкните правой кнопкой мыши на **RdChat\_WebRole** и нажмите **Add Reference**. На вкладке **.NET** выберите сборку **System.Data.Services.Client** и нажмите **OK**.
  5. Добавьте класс в проект Web-роли для модели таблицы сообщений: щелкните правой кнопкой мыши на проекте **RdChat\_WebRole** выберите **Add** и нажмите **Class**. В диалоговом окне **Add New Item** укажите значение имени **Name** как **Message.cs** (для проектов на Visual C#) или **Message.vb** (для проектов на Visual Basic) и добавьте файл в проект, нажав **Add**.
  6. Обновите объявление класса **Message,** унаследовав его от класса **Microsoft.WindowsAzure.StorageClient.TableServiceEntity**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-01-MessageClass-CS*)
     2. C#
     3. public class Message
     4. **: Microsoft.WindowsAzure.StorageClient.TableServiceEntity**
     5. {
     6. }
     7. (Сниппет – *ExploringWindowsAzureStorage-Ex01-01-MessageClass-VB*)
     8. Visual Basic
     9. Public Class Message
     10. **Inherits Microsoft.WindowsAzure.StorageClient.TableServiceEntity**
     11. End Class
     12. **Примечание:** Класс **TableServiceEntity** является частью библиотеки **Microsoft.WindowsAzure.StorageClient**. Он определяет системные свойства **PartititionKey**, **RowKey** и **TimeStamp**, необходимые для каждой сущности в таблице Windows Azure.
     13. **PartitionKey** и **RowKey** формируют **DataServiceKey**, уникально идентифицирующий каждую сущность в таблице.
  7. Добавьте в класс **Message** конструктор по умолчанию, инициализирующий **PartitionKey** и **RowKey**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-02-MessageConstructor-CS*)
     2. C#
     3. **public Message()**
     4. **{**
     5. **PartitionKey = "a";**
     6. **RowKey = string.Format("{0:10}\_{1}", DateTime.MaxValue.Ticks - DateTime.Now.Ticks, Guid.NewGuid());**
     7. **}**
     8. (Сниппет – *ExploringWindowsAzureStorage-Ex01-02-MessageConstructor-VB*)
     9. Visual Basic
     10. **Public Sub New()**
     11. **PartitionKey = "a"**
     12. **RowKey = String.Format("{0:10}\_{1}", (DateTime.MaxValue.Ticks - DateTime.Now.Ticks), Guid.NewGuid)**
     13. **End Sub**
     14. **Примечание:** В этом упражнении вы определяете для свойства **PartitionKey** фиксированное значение. В реальной ситуации лучше выбирать значение, обеспечивающее балансировку нагрузки между узлами хранилища.
  8. Добавьте в класс **Message** два строковых свойства, **Name** и **Body**, в которых будет храниться информация о сообщении в чате.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-03-TableSchemaProperties-CS*)
     2. C#
     3. **public string Name { get; set; }**
     4. **public string Body { get; set; }**
     5. (Сниппет – *ExploringWindowsAzureStorage-Ex01-03-TableSchemaProperties-VB*)
     6. Visual Basic
     7. **Public Property Body As String**
     8. **Public Property Name As String**
  9. Сохраните файл **Message.cs** / **Message.vb**.
  10. Добавьте класс в проект Web-роли для определения контекста WCF Data Services **DataServiceContext,** необходимого для доступа к таблице *Message*: щелкните правой кнопкой мыши на проекте **RdChat\_WebRole**, выберите **Add** и нажмите **Class**. В диалоговом окне **Add New Item** установите значение имени **Name** как **MessageDataServiceContext.cs** (для проектов на Visual C#) или **MessageDataServiceContext.vb** (для проектов на Visual Basic) и нажмите **Add**.
  11. Добавьте пространства имён в новый класс.
      1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-04-Namespace-CS*)
      2. C#
      3. **using Microsoft.WindowsAzure;**
      4. **using Microsoft.WindowsAzure.StorageClient;**
      5. (Сниппет – *ExploringWindowsAzureStorage-Ex01-04-Namespace-VB*)
      6. Visual Basic
      7. **Imports Microsoft.WindowsAzure**
      8. **Imports Microsoft.WindowsAzure.StorageClient**
  12. Замените объявление нового класса, добавив наследование от класса **TableServiceContext ,** и создайте конструктор по умолчанию, инициализирующий базовый класс информацией об аккаунте хранилища.
      1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-05-MessageDataServiceContextClass-CS*)
      2. C#
      3. namespace RdChat\_WebRole
      4. {
      5. **public class MessageDataServiceContext**
      6. **: TableServiceContext**
      7. **{**
      8. **public MessageDataServiceContext(string baseAddress, StorageCredentials credentials)**
      9. **: base(baseAddress, credentials)**
      10. **{**
      11. **}**
      12. **}**
      13. }

(Сниппет – *ExploringWindowsAzureStorage-Ex01-05-MessageDataServiceContextClass-VB*)

* + 1. Visual Basic
    2. **Public Class MessageDataServiceContext**
    3. **Inherits TableServiceContext**
    4. **Public Sub New(ByVal baseAddress As String, ByVal credentials As StorageCredentials)**
    5. **MyBase.New(baseAddress, credentials)**
    6. **End Sub**
    7. **End Class**
  1. Добавьте в класс **MessageDataServiceContext** свойство, возвращающее запрос сервиса данных к таблице *Messages*.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-06-MessagesProperty-CS*)
     2. C#
     3. public class MessageDataServiceContext
     4. : TableServiceContext
     5. {
     6. ...
     7. **public IQueryable<Message> Messages**
     8. **{**
     9. **get**
     10. **{**
     11. **return this.CreateQuery<Message>("Messages");**
     12. **}**
     13. **}**
     14. }
     15. (Сниппет – *ExploringWindowsAzureStorage-Ex01-06-MessagesProperty-VB*)
     16. Visual Basic
     17. Public Class MessageDataServiceContext
     18. Inherits TableServiceContext
     19. ...
     20. **Public ReadOnly Property Messages() As IQueryable(Of Message)**
     21. **Get**
     22. **Return MyBase.CreateQuery(Of Message)("Messages")**
     23. **End Get**
     24. **End Property**
     25. End Class
  2. Добавьте в класс **MessageDataServiceContext** метод добавления новых сообщений в таблице. Вы воспользуетесь этим методом при реализации функциональности чата.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-07-AddMessageMethod-CS*)
     2. C#
     3. public class MessageDataServiceContext
     4. : TableServiceContext
     5. {
     6. ...
     7. **public void AddMessage(string name, string body)**
     8. **{**
     9. **this.AddObject("Messages", new Message { Name = name, Body = body });**
     10. **this.SaveChanges();**
     11. **}**
     12. }
     13. (Сниппет – *ExploringWindowsAzureStorage-Ex01-07-AddMessageMethod-VB*)
     14. Visual Basic
     15. Public Class MessageDataServiceContext
     16. Inherits TableServiceContext
     17. ...
     18. **Public Sub AddMessage(ByVal name As String, ByVal body As String)**
     19. **Dim message As New Message**
     20. **message.Name = name**
     21. **message.Body = body**
     22. **MyBase.AddObject("Messages", message)**
     23. **MyBase.SaveChanges()**
     24. **End Sub**
     25. End Class
  3. В меню **Build** нажмите **Build Solution**.

Задание 3 – Создание пользовательского интерфейса для чата

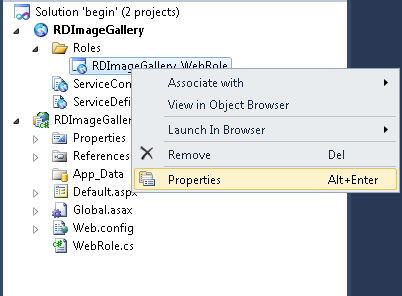
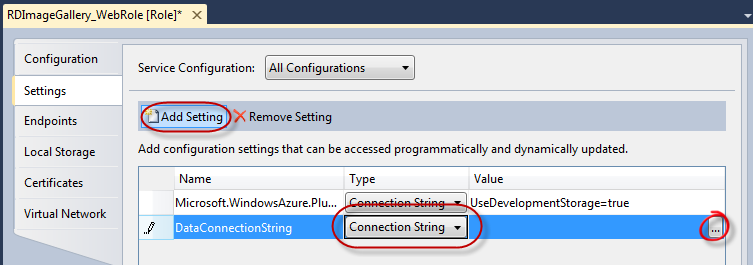
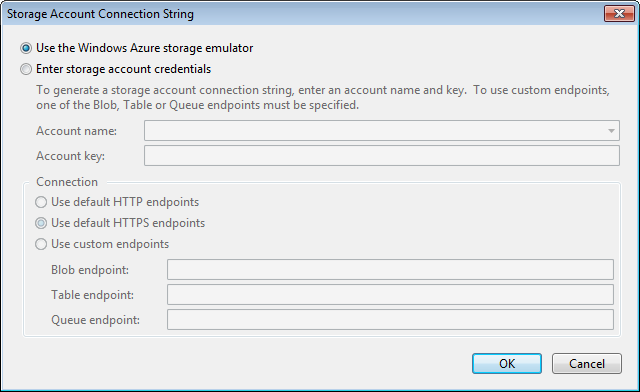
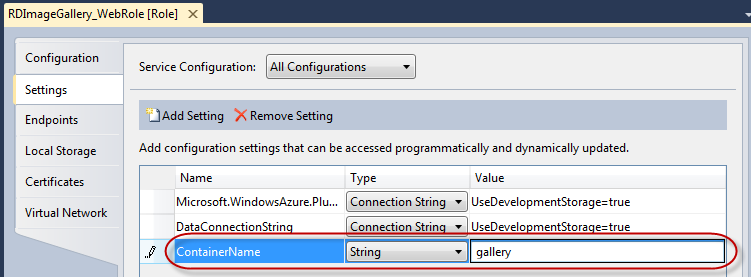
* 1. В этом задании вы добавите код, необходимый для хранения данных в таблице Windows Azure и их отображения на веб-странице.
  2. Найдите в файле **Global.asax.cs** / **Global.asax.vb** метод **Application\_Start** и вставьте следующий код (выделенный **жирным**). В этом коде создаются таблицы в хранилище согласно ранее созданному контексту **MessageDataServiceContext**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-08-ApplicationStartMethod-CS*)
     2. C#
     3. protected void Application\_Start()
     4. {
     5. ...
     6. **/// Создание таблицы данных согласно MessageDataServiceContext**
     7. **/// Рекомендуется создавать таблицы данных только единожды. Обычно этот процесс осуществляется на стадии подготовки, редко – в коде приложения**
     8. **var account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     9. **// динамическое создание таблицы**
     10. **CloudTableClient.CreateTablesFromModel(typeof(MessageDataServiceContext),**
     11. **account.TableEndpoint.AbsoluteUri, account.Credentials);**
     12. }
     13. (Сниппет – *ExploringWindowsAzureStorage-Ex01-08- ApplicationStartMethod-VB*)
     14. Visual Basic
     15. Sub Application\_Start(ByVal sender As Object, ByVal e As EventArgs)
     16. ...
     17. **CloudStorageAccount.SetConfigurationSettingPublisher (Sub(configName, configSetter) configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))**
     18. **' Создание таблицы данных согласно MessageDataServiceContext**
     19. **' Рекомендуется создавать таблицы данных только единожды. Обычно этот процесс осуществляется на стадии подготовки, редко – в коде приложения**
     20. **Dim account As CloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
     21. **' динамическое создание таблиц**
     22. **CloudTableClient.CreateTablesFromModel(GetType(MessageDataServiceContext), account.TableEndpoint.AbsoluteUri, account.Credentials)**
     23. End Sub
     24. **Примечание:** В коде, показанном выше, создаются необходимые таблицы согласно имеющейся модели, определенной в классе контекста **MessageDataServiceContext**.
     25. Обратите внимание на рекомендацию – таблицы данных должны создаваться только один раз. Обычно этот процесс выполняется на стадии подготовки и редко – в коде приложения. **Application\_Start** в классе **Global** является рекомендуемым местом для помещения логики инициализации.
     26. Для получения и отображения сообщений метод создает экземпляр класса **MessageDataServiceContext** и инициализирует его информацией аккаунта, доступной в файле конфигурации сервиса (**ServiceConfiguration.cscfg**). Для отображения он «привязывает» свойство **Messages**, возвращающее запрос сервиса данных таблицы Messages, к контролу ListView на странице.
     27. Объекты типа **CloudStorageAccount** являются отображением аккаунта хранилища, содержащим настройки, необходимые для установления подключения к сервису хранилища. «Привязанными» к аккаунту хранилища являются имя аккаунта, URI аккаунта и ключ доступа, что используется вспомогательным классом **CloudTableClient** для инициализации. Эти настройки хранятся и получаются из файла конфигурации сервиса **ServiceConfiguration.cscfg**.
  3. Убедитесь в присутствии следующих пространств имен в файле **Global.asax.cs** / **Global.asax.vb,** они нужны для полезных классов хранилища и классов из пространства имён **ServiceRuntime**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-09-GlobalNamespace-CS*)
     2. C#
     3. **using Microsoft.WindowsAzure;**
     4. **using Microsoft.WindowsAzure.ServiceRuntime;**
     5. **using Microsoft.WindowsAzure.StorageClient;**
     6. (Сниппет – *ExploringWindowsAzureStorage-Ex01-09- GlobalNamespace -VB*)
     7. Visual Basic
     8. **Imports Microsoft.WindowsAzure**
     9. **Imports Microsoft.WindowsAzure.ServiceRuntime**
     10. **Imports Microsoft.WindowsAzure.StorageClient**
  4. В **Solution Explorer** разверните узел **RDChat\_WebRole,** затем щелкните правой кнопкой мыши на **Default.aspx** и нажмите **View Code** для открытия файла исходников для веб-страницы, содержащей пользовательский интерфейс для приложения чата.
     1. Убедитесь в наличии пространств имён в файле исходников **Default.aspx.cs** / **Default.aspx.vb**.
     2. (Сниппет – *ExploringWindowsAzureStorage-Ex01-10-Namespace-CS*)
     3. C#
     4. **using System.Data.Services.Client;**
     5. **using Microsoft.WindowsAzure;**
     6. (Сниппет – *ExploringWindowsAzureStorage-Ex01-10-Namespace-VB*)
     7. Visual Basic
     8. **Imports System.Data.Services.Client**
     9. **Imports Microsoft.WindowsAzure**
  5. Вставьте в тело обработчика события **SubmitButton\_Click** в **Default.aspx.cs** / **Default.aspx.vb** код, выделенный **жирным**, для реализации сохранения сообщений, введенных пользователей, в сервис таблиц, и «привязки» сообщений из сервиса таблиц к странице. Метод использует другой метод **AddMessage**, который вы создали ранее в лабораторной, для вставки в таблицу новой сущности **Message**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex01-11-SubmitButtonClick-CS*)
     2. C#
     3. protected void SubmitButton\_Click(object sender, EventArgs e)
     4. {
     5. **var statusMessage = String.Empty;**
     6. **try**
     7. **{**
     8. **var account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     9. **var context = new MessageDataServiceContext(account.TableEndpoint.ToString(), account.Credentials);**
     10. **context.AddMessage(this.nameBox.Text, this.messageBox.Text);**
     11. **this.messageList.DataSource = context.Messages;**
     12. **this.messageList.DataBind();**
     13. **}**
     14. **catch (DataServiceRequestException ex)**
     15. **{**
     16. **statusMessage = "Unable to connect to the table storage server. Please check that the service is running.<br>"**
     17. **+ ex.Message;**
     18. **}**
     19. **status.Text = statusMessage;**
     20. }
     21. (Сниппет – *ExploringWindowsAzureStorage-Ex01-11-SubmitButtonClick-VB*)
     22. Visual Basic
     23. Protected Sub SubmitButton\_Click(ByVal sender As Object, ByVal e As EventArgs)
     24. **Dim statusMessage As String = String.Empty**
     25. **Try**
     26. **Dim account As CloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
     27. **Dim context As New MessageDataServiceContext(account.TableEndpoint.ToString, account.Credentials)**
     28. **context.AddMessage(Me.nameBox.Text, Me.messageBox.Text)**
     29. **Me.messageList.DataSource = context.Messages**
     30. **Me.messageList.DataBind()**
     31. **Catch ex As DataServiceRequestException**
     32. **statusMessage = ("Unable to connect to the table storage server. Please check that the service is running.<br>" & ex.Message)**
     33. **End Try**
     34. **Me.status.Text = statusMessage**
     35. End Sub
  6. Сохраните все файлы и нажмите **Build Solution** в меню **Build**.

1. Проверка
   1. Для проверки работоспособности вашего сервиса, работающего в эмуляторе вычислений:
   2. В Visual Studio нажмите **F5,** чтобы собрать и запустить приложение. Эмулятор вычислений запустит и инициализирует новое развертывание с Web-ролью **RdChat.** Окно браузера откроется для отображения веб-страницы чата в Windows Azure.
      1. 
      2. Изображение
      3. Приложение после успешного подключения к сервису таблиц эмулятора хранилища
      4. При запуске программы в отладчике Visual Studio автоматически запускает эмулятор хранилища. Если приложение не может получить доступ к серверу с сервисом таблицы, вы увидите ошибку. Для просмотра статуса сервиса щелкните правой кнопкой на иконке в системном трее (выглядит как сервер) и нажмите **Show Storage Emulator UI**.
      5. 
      6. Изображение
      7. Просмотр статуса эмулятора хранилища
   3. Протестируйте работу приложения чата, введя несколько сообщений. Наберите ваше имя и текст сообщений и нажмите **Submit**.
   4. В Internet Explorer нажмите **Ctrl + N** для открытия еще одного окна браузера. Введите другое имя и напишите еще несколько сообщений. Обратите внимание, что чат обновится.
      1. 
      2. Изображение
      3. Использование приложения чата Windows Azure

Упражнение 2: Работа с блобами

* 1. В этом упражнении вы воспользуетесь Windows Azure Blob Service API для создания приложения, которое сможет сохранять и получать изображения из блобов хранилища Windows Azure. Приложение будет состоять из простого веб-сайта галереи изображений, на котором можно будет просмотреть, загрузить и удалить изображения, хранящиеся в хранилище Windows Azure, и ввести и отобразить соответствующие изображениям метаданные. Приложение будет использовать один контейнер для хранения изображений-блобов.
  2. При создании блоба в Windows Azure вы «привязываете» к нему тип данных, указывающий формат, в котором API будет возвращать их и позволяющий вам получать изображение напрямую по URL соответствующего блоба.

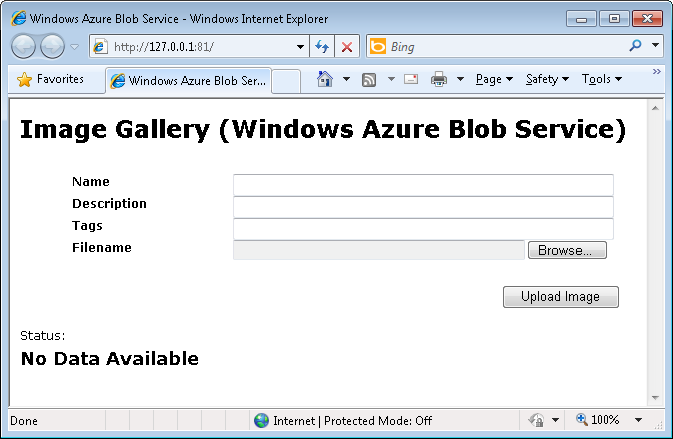
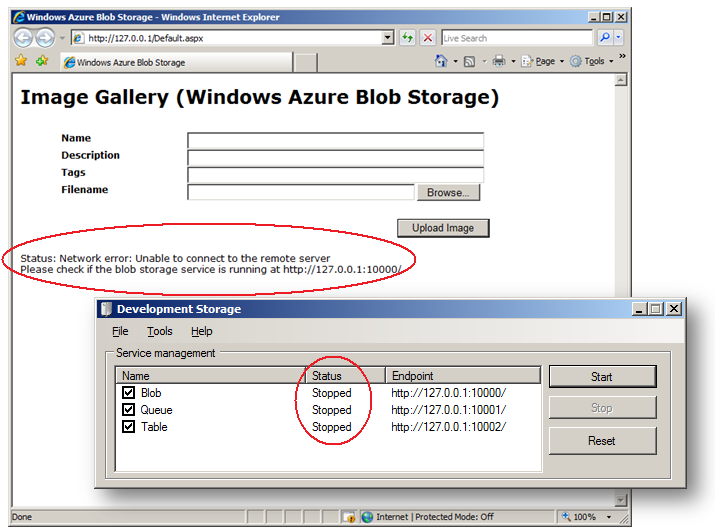
Задание 1 – Получение содержимого блоба из хранилища

* 1. В этом задании вы создадите веб-страницу с галерей, на которой будут отображаться изображения, полученные из хранилища Windows Azure. Предоставленное решение состоит из проекта веб-сайта с одной страницей, которая включает в себя элементы, необходимые для отображения изображений и ввода метаданных. Вы добавите необходимую функциональность, отредактировав файл исходников.
  2. Откройте Microsoft Visual Studio 2010 с повышенными привилегиями. Чтобы сделать это, в **Start** | **All Programs** | **Microsoft Visual Studio 2010** нажмите правой кнопкой мыши на ярлыке **Microsoft Visual Studio 2010** и выберите **Run as Administrator**.
  3. В меню **File** выберите **Open** и **Project/Solution**. В диалоговом окне **Open Project** перейдите в папке **Source\Ex02-WorkingWithBlobs\begin** в папке лабораторной **Source**, выберите **Begin.sln,** исходя из используемого вами языка программирования (Visual C# или Visual Basic) и нажмите **Open**.
  4. Определите настройки конфигурации для доступа к сервису блобов: откройте папку **Roles** в проекте **RDImageGallery**, щелкните правой кнопкой мыши на **RDImageGallery\_WebRole** и выберите **Properties**.
     1. 
     2. Изображение
     3. Setting role configuration settings
  5. Перейдите на вкладку **Settings**, нажмите **Add Setting** и создайте новую настройку конфигурации с именем *DataConnectionString*. Укажите её тип как **Connection String**, после чего нажмите на кнопку с троеточием и настройте строку подключения к хранилищу на использование локального эмулятора хранилища - *Use storage emulator*.
     1. 
     2. Изображение
     3. *Создание строки подключения к хранилищу*
     4. 
     5. Изображение
     6. Настройки строки подключения к хранилищу на использование локального эмулятора хранилища
  6. Создайте еще одну настройку конфигурации *ContainerName* и установите ее значение как ‘*gallery’*.
     1. 
     2. Изображение
     3. Создание настройки для имени контейнера
     4. **Примечание:**  Имя контейнера должно быть корректным DNS-именем, удовлетворяя следующим конвенциям именования:
     5. - Должно начинаться с буквы или цифры, и может содержать только буквы, цифры и символы дефиса.

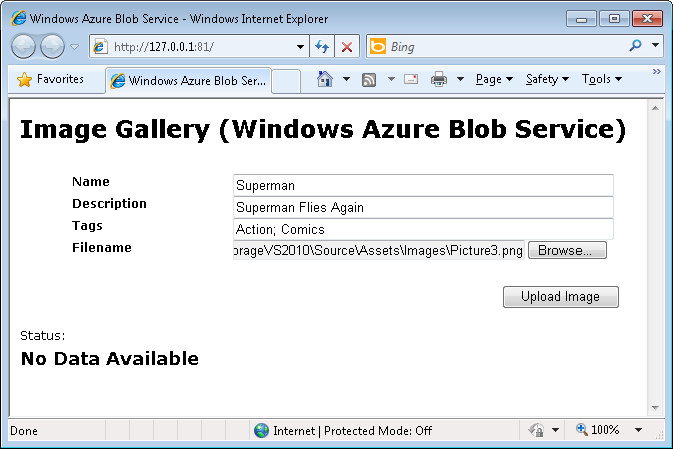
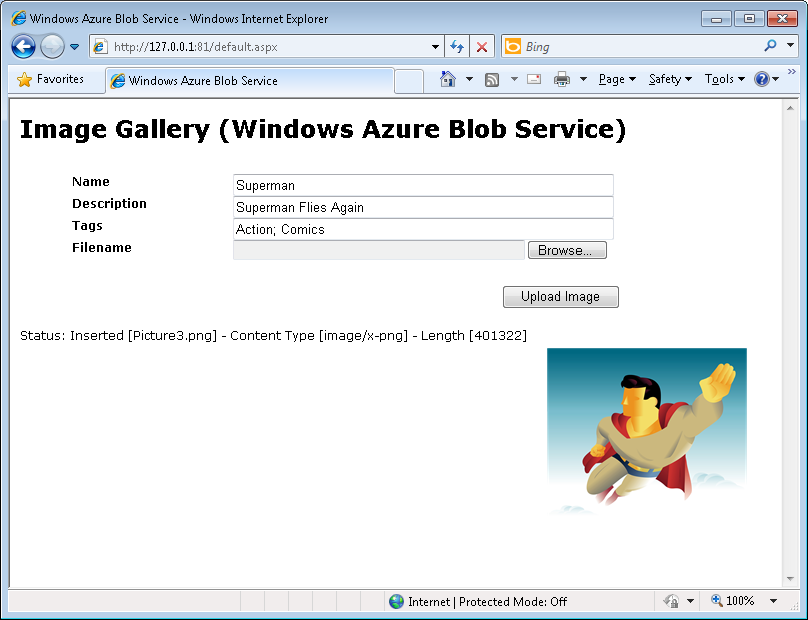
- Все буквы должны быть низшего регистра.

- Длина имени должна быть от 3 до 63 символов.

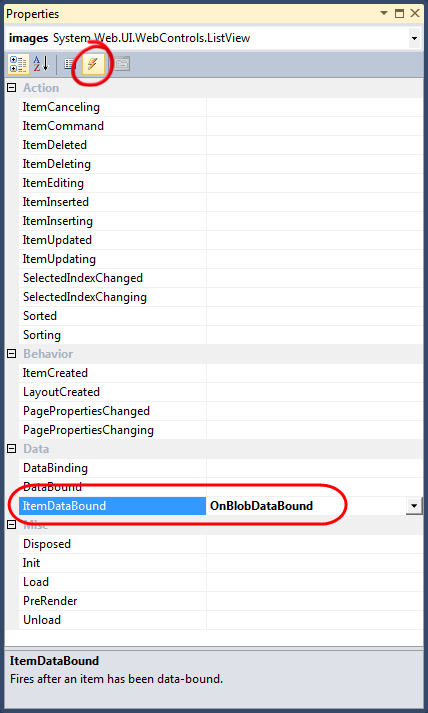
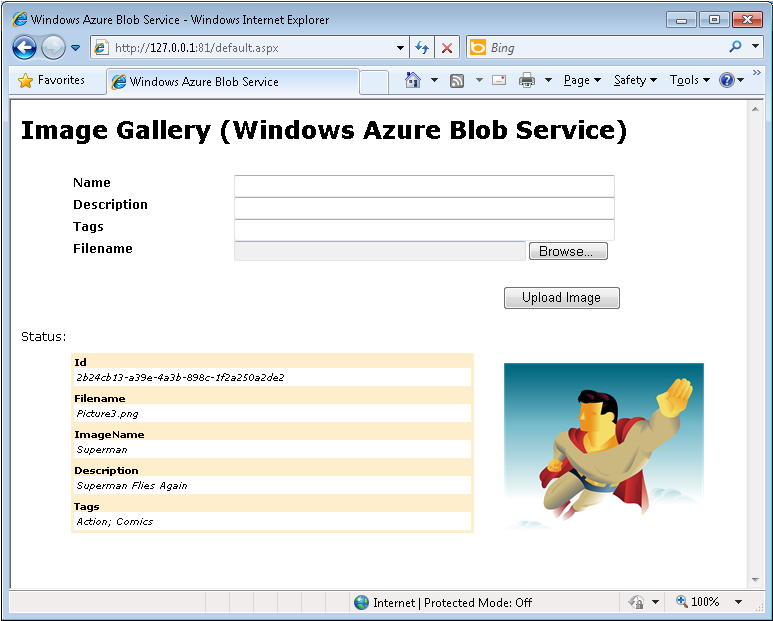
- Имя не может содержат дефис после точки

* 1. В **Solution Explorer** разверните узел **RDImageGallery\_WebRole,** затем щелкните правой кнопкой мыши на **Default.aspx** и нажмите **View Code** для открытия файла исходников для веб-страницы, содержащей пользовательский интерфейс для приложения.
  2. Убедитесь в присутствии следующих пространств имен в файле - они нужны для полезных классов хранилища и классов из пространства имён **ServiceRuntime**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-01-Namespace-CS*)
     2. C#
     3. **using Microsoft.WindowsAzure;**
     4. **using Microsoft.WindowsAzure.ServiceRuntime;**
     5. **using Microsoft.WindowsAzure.StorageClient;**
     6. **using Microsoft.WindowsAzure.StorageClient.Protocol;**
     7. (Сниппет – *ExploringWindowsAzureStorage-Ex02-01-Namespace-VB*)
     8. Visual Basic
     9. **Imports Microsoft.WindowsAzure**
     10. **Imports Microsoft.WindowsAzure.ServiceRuntime**
     11. **Imports Microsoft.WindowsAzure.StorageClient**
     12. **Imports Microsoft.WindowsAzure.StorageClient.Protocol**
  3. В этой лабораторной вы будете хранить блобы в публично доступном блобе – они будут доступны анонимным пользователям. Убедитесь в существовании контейнера, определенного в файле конфигурации сервиса **ServiceConfiguration.cscfg:** добавьте метод в конец **\_Default**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-02-EnsureContainerExistsMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. **private void EnsureContainerExists()**
     7. **{**
     8. **var container = GetContainer();**
     9. **container.CreateIfПримечаниеxist();**
     10. **var permissions = container.GetPermissions();**
     11. **permissions.PublicAccess = BlobContainerPublicAccessType.Container;**
     12. **container.SetPermissions(permissions);**
     13. **}**
     14. }
     15. (Сниппет – *ExploringWindowsAzureStorage-Ex02-02-EnsureContainerExistsMethod-VB*)
     16. Visual Basic
     17. Partial Public Class \_Default
     18. Inherits System.Web.UI.Page
     19. ...
     20. **Private Sub EnsureContainerExists()**
     21. **Dim container = GetContainer()**
     22. **container.CreateIfПримечаниеxist()**
     23. **Dim permissions = container.GetPermissions()**
     24. **permissions.PublicAccess = BlobContainerPublicAccessType.Container**
     25. **container.SetPermissions(permissions)**
     26. **End Sub**
     27. End Class
  4. Создайте метод для получения ссылки на контейнер, созданный в предыдущем шаге. Этот метод будет вызываться практически во всех операциях с блобами, так как работа с блобами непосредственно связана с контейнерами. Добавьте метод для создания контейнера в конец класса **\_Default**. Этот метод будет использовать созданные ранее настройки конфигурации.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-03-GetContainerMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. **private CloudBlobContainer GetContainer()**
     7. **{**
     8. **// Получение информации об аккаунте, создание клиента сервиса блобов и получение прокси контейнера**
     9. **var account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     10. **var client = account.CreateCloudBlobClient();**
     11. **return client.GetContainerReference(RoleEnvironment.GetConfigurationSettingValue("ContainerName"));**
     12. **}**
     13. }
     14. (Сниппет – *ExploringWindowsAzureStorage-Ex02-03-GetContainerMethod-VB*)
     15. Visual Basic
     16. Partial Public Class \_Default
     17. Inherits System.Web.UI.Page
     18. ...
     19. **Private Function GetContainer() As CloudBlobContainer**
     20. **' Получение информации об аккаунте, создание клиента сервиса блобов и получение прокси контейнера**
     21. **Dim account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
     22. **Dim client = account.CreateCloudBlobClient()**
     23. **Return client.GetContainerReference(RoleEnvironment.GetConfigurationSettingValue("ContainerName"))**
     24. **End Function**
     25. End Class
  5. Вставьте код, выделенный **жирным**, в метод **Page\_Load,** чтобы инициализировать контейнер и обновить контрол **asp:ListView,** расположенный на странице с галереей.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-04-PageLoadMethod-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. **try**
     9. **{**
     10. **if (!IsPostBack)**
     11. **{**
     12. **this.EnsureContainerExists();**
     13. **}**
     14. **this.RefreshGallery();**
     15. **}**
     16. **catch (System.Net.WebException we)**
     17. **{**
     18. **status.Text = "Network error: " + we.Message;**
     19. **if (we.Status == System.Net.WebExceptionStatus.ConnectFailure)**
     20. **{**
     21. **status.Text += "<br />Please check if the blob service is running at " +**
     22. **ConfigurationManager.AppSettings["storageEndpoint"];**
     23. **}**
     24. **}**
     25. **catch (StorageException se)**
     26. **{**
     27. **Console.WriteLine("Storage service error: " + se.Message);**
     28. **}**
     29. }
     30. ...
     31. }
     32. (Сниппет – *ExploringWindowsAzureStorage-Ex02-04-PageLoadMethod-VB*)
     33. Visual Basic
     34. Partial Public Class \_Default
     35. Inherits System.Web.UI.Page
     36. ...
     37. Protected Sub Page\_Load(ByVal sender As Object, ByVal e As EventArgs)
     38. **Try**
     39. **If Not IsPostBack Then**
     40. **Me.EnsureContainerExists()**
     41. **End If**
     42. **Me.RefreshGallery()**
     43. **Catch we As System.Net.WebException**
     44. **status.Text = "Network error: " & we.Message**
     45. **If we.Status = System.Net.WebExceptionStatus.ConnectFailure Then**
     46. **status.Text += "<br />Please check if the blob service is running at " & ConfigurationManager.AppSettings("storageEndpoint")**
     47. **End If**
     48. **Catch se As StorageException**
     49. **Console.WriteLine("Storage service error: " & se.Message)**
     50. **End Try**
     51. End Sub
     52. ...
     53. End Class
  6. Добавьте метод в конец класса **\_Default**. Этот метод будет «привязывать» контрол *images*  к списку доступных в контейнере блобов-изображений. Код использует для получения коллекции объектов **IlistBlobItem,** содержащую информацию обо всех блобах, метод **ListBlobs** объекта **CloudBlobContainer.** Контрол *images* **asp:ListView** на странице «привязывается» к этим объектам для отображения их значений.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-05-RefreshGalleryMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. **private void RefreshGallery()**
     7. **{**
     8. **images.DataSource =**
     9. **this.GetContainer().ListBlobs(new BlobRequestOptions()**
     10. **{**
     11. **UseFlatBlobListing = true,**
     12. **BlobListingDetails = BlobListingDetails.All**
     13. **});**
     14. **images.DataBind();**
     15. **}**
     16. }
     17. (Сниппет – *ExploringWindowsAzureStorage-Ex02-05-RefreshGalleryMethod-VB*)
     18. Visual Basic
     19. Partial Public Class \_Default
     20. Inherits System.Web.UI.Page
     21. ...
     22. **Private Sub RefreshGallery()**
     23. **Dim options As BlobRequestOptions = New BlobRequestOptions() With**
     24. **{.BlobListingDetails = BlobListingDetails.All,**
     25. **.UseFlatBlobListing = True}**
     26. **images.DataSource = Me.GetContainer().ListBlobs(options)**
     27. **images.DataBind()**
     28. **End Sub**
     29. End Class
  7. Нажмите **F5** для сборки и запуска приложения. В окне браузера отобразится содержание галереи изображений. Обратите внимание, что в данный момент контейнер пуст и контрол отобразит только сообщение об отсутствии доступных данных. В следующем задании вы реализуете функциональность, необходимую для хранения изображений в качестве блобов в хранилище Windows Azure.
     1. 
     2. Изображение
     3. Приложение галереи изображений отображает пустой контейнер
     4. **Примечание:** Если вы неправильно сконфигурировали настройки хранилища или сервис хранилища не запущен, будет выведена ошибка.
     5. 
     6. Изображение
     7. Ошибка, вызванная неправильной конфигурацией хранилища или проблемой с сервисом
  8. Нажмите **Shift+F5** для остановки отладки и удаления развертывания из эмулятора вычислений.

Задание 2 – Загрузка блобов в хранилище

* 1. В этом задании вы добавите в приложение возможность ввода метаданных и загрузки изображений в хранилище Windows Azure. Страница содержит текстовые контролы, которые вы будете использовать для ввода метаданных для выбранного изображения. Контрол **asp:FileUpload** будет получать изображение с диска и отправлять их на страницу, после чего они будут сохраняться в хранилище блобов.
  2. Откройте в текстовом редакторе файл ***Default.aspx.cs*** */* ***Default.aspx.vb*:**щелкните правой кнопкой мыши на файле Default.aspx и нажмите **View Code**.
  3. Добавьте в конец страницы метод сохранения изображений и метаданных как блобов в хранилище. Для создания блоба из изображения и метаданных используется метод **GetBlobReference** объекта **CloudBlobContainer**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-06-SaveImageMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. **private void SaveImage(string id, string name, string description, string tags, string fileName, string contentType, byte[] data)**
     7. **{**
     8. **// Создание блоба в контейнера и загрузка в него байтов изображения**
     9. **var blob = this.GetContainer().GetBlobReference(name);**
     10. **blob.Properties.ContentType = contentType;**
     11. **// Создание метаданных для блоба**
     12. **var metadata = new NameValueCollection();**
     13. **metadata["Id"] = id;**
     14. **metadata["Filename"] = fileName;**
     15. **metadata["ImageName"] = String.IsNullOrEmpty(name) ? "unknown" : name;**
     16. **metadata["Description"] = String.IsNullOrEmpty(description) ? "unknown" : description;**
     17. **metadata["Tags"] = String.IsNullOrEmpty(tags) ? "unknown" : tags;**
     18. **// Добавление и сохранение метаданных**
     19. **blob.Metadata.Add(metadata);**
     20. **blob.UploadByteArray(data);**
     21. **}**
     22. }
     23. (Сниппет – *ExploringWindowsAzureStorage-Ex02-06-SaveImageMethod-VB*)
     24. Visual Basic
     25. Partial Public Class \_Default
     26. Inherits System.Web.UI.Page
     27. ...
     28. **Private Sub SaveImage(ByVal id As String, ByVal name As String, ByVal description As String, ByVal tags As String, ByVal fileName As String, ByVal contentType As String, \_**
     29. **ByVal data As Byte())**
     30. **' Создание блоба в контейнера и загрузка в него байтов изображения**
     31. **Dim blob = Me.GetContainer().GetBlobReference(name)**
     32. **blob.Properties.ContentType = contentType**
     33. **' Создание метаданных для блоба**
     34. **Dim metadata = New NameValueCollection()**
     35. **metadata("Id") = id**
     36. **metadata("Filename") = fileName**
     37. **metadata("ImageName") = If([String].IsNullOrEmpty(name), "unknown", name)**
     38. **metadata("Description") = If([String].IsNullOrEmpty(description), "unknown", description)**
     39. **metadata("Tags") = If([String].IsNullOrEmpty(tags), "unknown", tags)**
     40. **' Добавление и сохранение метаданных**
     41. **blob.Metadata.Add(metadata)**
     42. **blob.UploadByteArray(data)**
     43. **End Sub**
     44. End Class
  4. Добавьте в обработчик события кнопки **Upload Image upload\_Click** код, выделенный **жирным**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-07-UploadClickMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. protected void upload\_Click(object sender, EventArgs e)
     7. {
     8. **if (imageFile.HasFile)**
     9. **{**
     10. **status.Text = "Inserted [" + imageFile.FileName + "] - Content Type [" + imageFile.PostedFile.ContentType + "] - Length [" + imageFile.PostedFile.ContentLength + "]";**
     11. **this.SaveImage(**
     12. **Guid.NewGuid().ToString(),**
     13. **imageName.Text,**
     14. **imageDescription.Text,**
     15. **imageTags.Text,**
     16. **imageFile.FileName,**
     17. **imageFile.PostedFile.ContentType,**
     18. **imageFile.FileBytes**
     19. **);**
     20. **RefreshGallery();**
     21. **}**
     22. **else**
     23. **status.Text = "No image file";**
     24. }
     25. ...
     26. }
     27. (Сниппет – *ExploringWindowsAzureStorage-Ex02-07-UploadClickMethod-VB*)
     28. Visual Basic
     29. Protected Sub upload\_Click(ByVal sender As Object, ByVal e As EventArgs)
     30. **If imageFile.HasFile Then**
     31. **status.Text = (("Inserted [" & imageFile.FileName & "] - Content Type [") + imageFile.PostedFile.ContentType & "] - Length [") + imageFile.PostedFile.ContentLength.ToString() + "]"**
     32. **Me.SaveImage(Guid.NewGuid().ToString(), imageName.Text, imageDescription.Text, imageTags.Text, imageFile.FileName, imageFile.PostedFile.ContentType, \_**
     33. **imageFile.FileBytes)**
     34. **RefreshGallery()**
     35. **Else**
     36. **status.Text = "No image file"**
     37. **End If**
     38. End Sub
     39. С помощью этого кода получаются метаданные из текстовых контролов и свойств контрола **asp:FileUpload**, включающих тип содержания загружаемого файла, имя файла и массив байт изображения. Затем вызывается метод **SaveImage**, сохраняющий изображение и метаданные в хранилище Windows Azure.
  5. Нажмите **F5** для сборки и запуска приложений и открытия приложения в браузере.
  6. Введите метаданные в поля **Name**, **Description** и **Tags**. Нажмите **Browse,** перейдите в папку **\Source\Assets\Images**, и выберите одно из доступных изображений.
     1. 
     2. Изображение
     3. Ввод метаданных для изображения
  7. Нажмите **Upload Image** для загрузки изображения в приложение. Страница обновится и новое изображение появится в списке изображений с подписанными именем файла, типом содержимого и размером изображения. Обратите внимание, что в данный момент с изображением не ассоциированы метаданные. В следующем задании вы реализуете логику получения и отображения метаданных для блобов Windows Azure.
     1. 
     2. Изображение
     3. Загруженное изображение в галерее
  8. Нажмите **Shift+F5** в Visual Studio, чтобы остановить отладку и удалить развертывание из эмулятора вычислений.

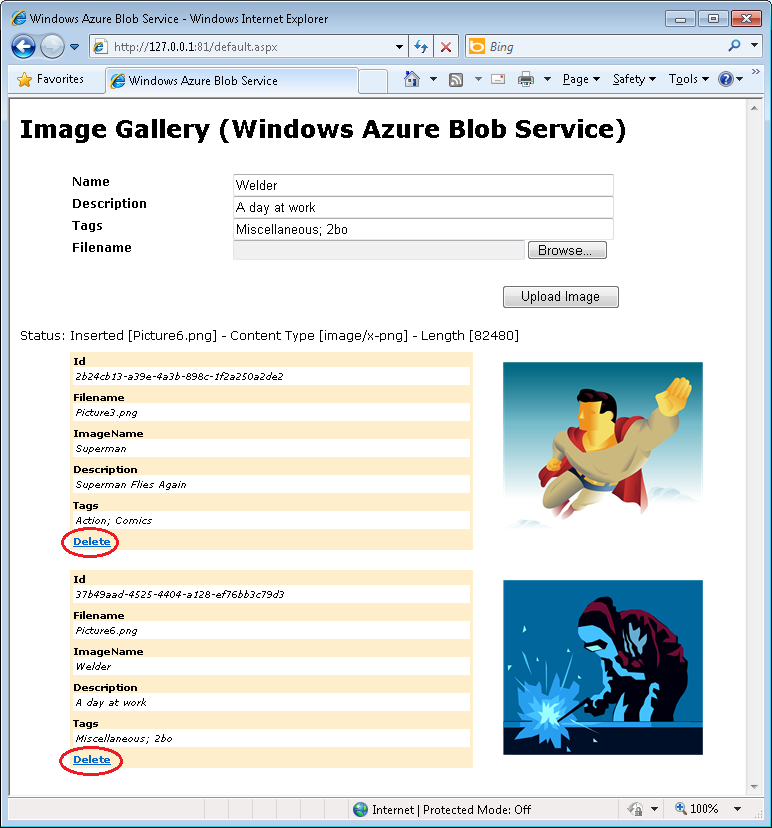
Задание 3 – Получение из хранилища метаданных для блобов

* 1. Блобы могут иметь ассоциированные с ними метаданные. Заголовки метаданных могут быть определены по запросу при создании нового контейнера или блоба, или при «привязке» метаданных к уже существующим ресурсам. В этом задании вы добавите в приложение логику получения и отображения метаданных изображений, хранящихся в контейнере Windows Azure.
  2. Добавьте обработчик события, который будет получать метаданные для каждого блоба в списке: откройте **Default.aspx**, щелкните правой кнопкой мыши на **View Designer**, выберите контрол ***images*** **ListView** и в окне свойств **Properties** (возможно, вам придется сделать это окно видимым, щелкнув правой кнопкой мыши на контроле и выбрав **Properties**) нажмите кнопку **Events**. Найдите в категории *Data* событие **ItemDataBound** *,* наберите **OnBlobDataBound** и нажмите <**Enter**>. Также вы можете отредактировать разметку вручную и вставить соответствующий код для вызова обработчика событий.
     1. 
     2. Изображение
     3. Настройка обработчика событий для отображения метаданных
  3. В файле исходников найдите метод **OnBlobDataBound** и вставьте код, выделенный **жирным**, который осуществляет получение свойств для каждого блоба в списке и создает коллекцию пар ключ-значение для каждой сущности метаданных. Эта коллекция используется как источник данных для контрола **asp:Repeater**, отображающего метаданные для каждого изображения.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-08-OnBlobDataBoundMethod-CS*)
     2. C#
     3. protected void OnBlobDataBound(object sender, ListViewItemEventArgs e)
     4. {
     5. **if (e.Item.ItemType == ListViewItemType.DataItem)**
     6. **{**
     7. **var metadataRepeater = e.Item.FindControl("blobMetadata") as Repeater;**
     8. **var blob = ((ListViewDataItem)(e.Item)).DataItem as CloudBlob;**
     10. **// If this blob is a snapshot, rename button to "Delete Snapshot"**
     11. **if (blob != null)**
     12. **{**
     13. **if (blob.SnapshotTime.HasValue)**
     14. **{**
     15. **var delBtn = e.Item.FindControl("deleteBlob") as LinkButton;**
     17. **if (delBtn != null)**
     18. **{**
     19. **delBtn.Text = "Delete Snapshot";**
     20. **var snapshotRequest = BlobRequest.Get(new Uri(delBtn.CommandArgument), 0, blob.SnapshotTime.Value, null);**
     21. **delBtn.CommandArgument = snapshotRequest.RequestUri.AbsoluteUri;**
     22. **}**
     24. **var snapshotBtn = e.Item.FindControl("SnapshotBlob") as LinkButton;**
     25. **if (snapshotBtn != null) snapshotBtn.Visible = false;**
     26. **}**
     28. **if (metadataRepeater != null)**
     29. **{**
     30. **//bind to metadata**
     31. **metadataRepeater.DataSource = from key in blob.Metadata.AllKeys**
     32. **select new**
     33. **{**
     34. **Name = key,**
     35. **Value = blob.Metadata[key]**
     36. **};**
     37. **metadataRepeater.DataBind();**
     38. **}**
     39. **}**
     40. **}**
     41. }
     42. (Сниппет – *ExploringWindowsAzureStorage-Ex02-08-OnBlobDataBoundMethod-VB*)
     43. Visual Basic
     44. Protected Sub OnBlobDataBound(ByVal sender As Object, ByVal e As ListViewItemEventArgs) Handles images.ItemDataBound
     45. **If e.Item.ItemType = ListViewItemType.DataItem Then**
     46. **Dim metadataRepeater = TryCast(e.Item.FindControl("blobMetadata"), Repeater)**
     47. **Dim blob = TryCast(DirectCast((e.Item), ListViewDataItem).DataItem, CloudBlob)**
     48. **' If this blob is a snapshot, rename button to "Delete Snapshot"**
     49. **If blob IsNot Nothing Then**
     50. **If blob.SnapshotTime.HasValue Then**
     51. **Dim delBtn = TryCast(e.Item.FindControl("deleteBlob"), LinkButton)**
     52. **If delBtn IsNot Nothing Then**
     53. **delBtn.Text = "Delete Snapshot"**
     54. **Dim snapshotRequest = BlobRequest.Get(New Uri(delBtn.CommandArgument), 0, blob.SnapshotTime.Value, Nothing)**
     55. **delBtn.CommandArgument = snapshotRequest.RequestUri.AbsoluteUri**
     56. **End If**
     57. **Dim snapshotBtn = TryCast(e.Item.FindControl("Snapshot"), LinkButton)**
     58. **If snapshotBtn IsNot Nothing Then**
     59. **snapshotBtn.Visible = False**
     60. **End If**
     61. **End If**
     62. **If metadataRepeater IsNot Nothing Then**
     63. **'bind to metadata**
     64. **metadataRepeater.DataSource = From key In blob.Metadata.AllKeys \_**
     65. **Select New With {.Name = key, .Value = blob.Metadata.Get(key)}**
     66. **metadataRepeater.DataBind()**
     67. **End If**
     68. **End If**
     69. **End If**
     70. End Sub
  4. Нажмите **F5** для сборки и запуска приложения. Обратите внимание, что список теперь содержит метаданные для изображения, загруженного в предыдущем упражнении.
     1. 
     2. Изображение
     3. Метаданные изображения
  5. Нажмите **Shift+F5** для остановки отладки и удаления развертывания из эмулятора вычислений.

Задание 4 – Удаление блобов из хранилища

* 1. В этом задании вы добавите в приложение функциональность удаления блобов, содержащих изображения, из хранилища Windows Azure.
  2. Добавьте на представление контрол **asp:LinkButton** для удаления изображений из контейнера: щелкните правой кнопкой мыши на **Default.aspx**, нажмите **View Markup**, найдите **ItemTemplate** для контрола **asp:ListView**. Раскомментируйте разметку ASP.NET, следующую сразу после контрола **blobMetadata** repeater (выделено **жирным**).
     1. HTML
     2. ...
     3. <div class="item">
     4. <ul style="width:40em;float:left;clear:left" >
     5. <asp:Repeater ID="blobMetadata" runat="server">
     6. <ItemTemplate>
     7. <li><%# Eval("Name") %><span><%# Eval("Value") %></span></li>
     8. </ItemTemplate>
     9. </asp:Repeater>
     10. **<li>**
     11. **<asp:LinkButton ID="deleteBlob"**
     12. **OnClientClick="return confirm('Delete image?');"**
     13. **CommandName="Delete"**
     14. **CommandArgument='<%# Eval("Uri")%>'**
     15. **runat="server" Text="Delete" oncommand="OnDeleteImage" />**
     16. ...
     17. **</li>**
     18. </ul>
     19. <img src="<%# Eval("Uri") %>" alt="<%# Eval("Uri") %>" style="float:left"/>
     20. </div>
     21. ...
  3. Добавьте код, выделенный **жирным**, в файл **Default.aspx.cs** / **Default.aspx.vb,** для реализации обработчика контрола *deleteBlob* **asp:LinkButton**. В коде проводится проверка существования блоба в хранилище и последующее его удаление.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-09-OnDeleteImageMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. protected void OnDeleteImage(object sender, CommandEventArgs e)
     7. {
     8. **try**
     9. **{**
     10. **if (e.CommandName == "Delete")**
     11. **{**
     12. **var blobUri = (string)e.CommandArgument;**
     13. **var blob = this.GetContainer().GetBlobReference(blobUri);**
     14. **blob.DeleteIfExists();**
     15. **}**
     16. **}**
     17. **catch (StorageClientException se)**
     18. **{**
     19. **status.Text = "Storage client error: " + se.Message;**
     20. **}**
     21. **catch (Exception) { }**
     22. **RefreshGallery();**
     23. }

...

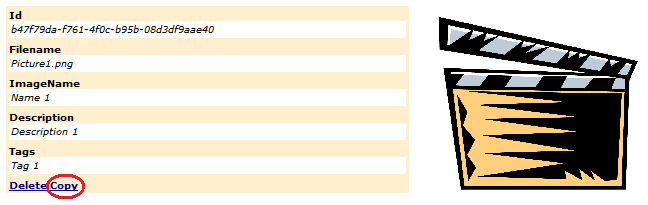
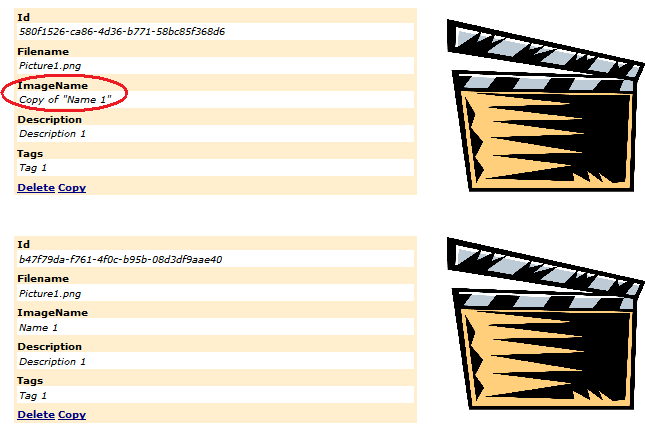
* + 1. }
    2. (Сниппет – *ExploringWindowsAzureStorage-Ex02-09-OnDeleteImageMethod-VB*)
    3. Visual Basic
    4. Partial Public Class \_Default
    5. Inherits System.Web.UI.Page
    6. ...
    7. Protected Sub OnDeleteImage(ByVal sender As Object, ByVal e As CommandEventArgs)
    8. **Try**
    9. **If e.CommandName = "Delete" Then**
    10. **Dim blobUri = DirectCast(e.CommandArgument, String)**
    11. **Dim blob = Me.GetContainer().GetBlobReference(blobUri)**
    12. **blob.DeleteIfExists()**
    13. **End If**
    14. **Catch se As StorageClientException**
    15. **status.Text = "Storage client error: " & se.Message**
    16. **Catch generatedExceptionName As Exception**
    17. **End Try**
    18. **RefreshGallery()**
    19. End Sub
    20. ...
    21. End Class
  1. Нажмите **F5** для сборки и запуска приложения.
  2. Загрузите еще несколько изображений из папки **Source\Assets\Images** и нажмите **Delete** на любом из изображений для его удаления из хранилища.
     1. 
     2. Изображение
     3. Добавление и удаление изображений из хранилища
  3. Нажмите **Shift+F5** для остановки отладки и удаления развертывания из эмулятора вычислений.

Задание 5 – Копирование блобов

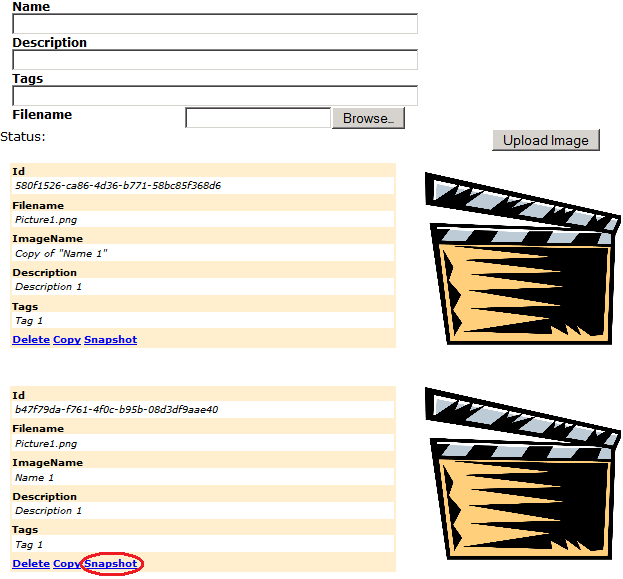
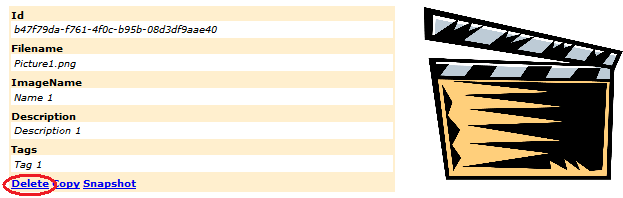
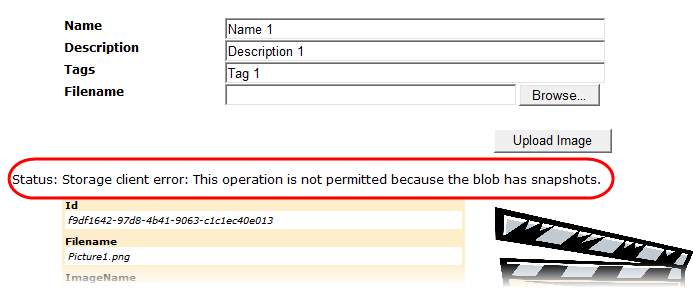
* 1. Сервис блобов Windows Azure имеет поддержку копирования существующих блобов. В этом задании вы реализуете копирование блобов.
  2. Добавьте на представление контрол **asp:LinkButton**, который будет использоваться для копирования блобов из хранилища: откройте страницу **Default.aspx** в режиме Source, найдите **ItemTemplate** для контрола **asp:ListView** и раскомментируйте ASP.NET разметку, расположенную непосредственно после контрола удаления блоба (выделено **жирным**)
     1. HTML
     2. ...
     3. <div class="item">
     4. <ul style="width:40em;float:left;clear:left" >
     5. <asp:Repeater ID="blobMetadata" runat="server">
     6. <ItemTemplate>
     7. <li><%# Eval("Name") %><span><%# Eval("Value") %></span></li>
     8. </ItemTemplate>
     9. </asp:Repeater>
     10. <li>
     11. <asp:LinkButton ID="deleteBlob"
     12. OnClientClick="return confirm('Delete image?');"
     13. CommandName="Delete"
     14. CommandArgument='<%# Eval("Uri")%>'
     15. runat="server" Text="Delete" oncommand="OnDeleteImage" />
     16. **<asp:LinkButton ID="CopyBlob"**
     17. **OnClientClick="return confirm('Copy image?');"**
     18. **CommandName="Copy"**
     19. **CommandArgument='<%# Eval("Uri")%>'**
     20. **runat="server" Text="Copy" oncommand="OnCopyImage" />**

...

* + 1. </li>
    2. </ul>
    3. <img src="<%# Eval("Uri") %>" alt="<%# Eval("Uri") %>" style="float:left"/>
    4. </div>
    5. ...
  1. Добавьте код, выделенный **жирным** , в файл **Default.aspx.cs** / **Default.aspx.vb** для реализации обработчика контрола *copyBlob* **asp:LinkButton**. В коде создается копия существующего блоба и обновляется значение атрибута “*ImageName*” метаданных, отображающее то, что блоб является копией.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-10-OnCopyImageMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. protected void OnCopyImage(object sender, CommandEventArgs e)
     7. {
     8. **if (e.CommandName == "Copy")**
     9. **{**
     10. **// Подготовка ID для блоба-копии**
     11. **var newId = Guid.NewGuid();**
     12. **// Получение блоба-источника**
     13. **var blobUri = (string)e.CommandArgument;**
     14. **var srcBlob = this.GetContainer().GetBlobReference(blobUri);**
     15. **// Создание нового блоба**
     16. **var newBlob = this.GetContainer().GetBlobReference(newId.ToString());**
     17. **// Копирование содержимого блоба-источника**
     18. **newBlob.CopyFromBlob(srcBlob);**
     19. **// Получение метаданных для нового блоба**
     20. **newBlob.FetchAttributes(new BlobRequestOptions { BlobListingDetails = BlobListingDetails.Metadata });**
     21. **// Внесение в метаданные сообщения о том, что новый блоб является копией**
     22. **newBlob.Metadata["ImageName"] = "Copy of \"" + newBlob.Metadata["ImageName"] + "\"";**
     23. **newBlob.Metadata["Id"] = newId.ToString();**
     24. **newBlob.SetMetadata();**
     25. **// Генерация всех блобов**
     26. **RefreshGallery();**
     27. **}**
     28. }
     29. ...

1. }
   * 1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-10-OnCopyImageMethod-VB*)
     2. Visual Basic
     3. Partial Public Class \_Default
     4. Inherits System.Web.UI.Page
     5. ...
     6. Protected Sub OnCopyImage(ByVal sender As Object, ByVal e As CommandEventArgs)
     7. **If e.CommandName = "Copy" Then**
     8. **' Подготовка ID для блоба-копии**
     9. **Dim newId = Guid.NewGuid()**
     10. **' Получение блоба-источника**
     11. **Dim blobUri = DirectCast(e.CommandArgument, String)**
     12. **Dim srcBlob = Me.GetContainer().GetBlobReference(blobUri)**
     13. **' Создание нового блоба**
     14. **Dim newBlob = Me.GetContainer().GetBlobReference(newId.ToString())**
     15. **' Копирование содержимого блоба-источника**
     16. **newBlob.CopyFromBlob(srcBlob)**
     17. **' Получение метаданных для нового блоба**
     18. **newBlob.FetchAttributes(New BlobRequestOptions())**
     19. **' Внесение в метаданные сообщения о том, что новый блоб является копией**
     20. **newBlob.Metadata("ImageName") = "Copy of """ & newBlob.Metadata("ImageName") & """"**
     21. **newBlob.Metadata("Id") = newId.ToString()**
     22. **newBlob.SetMetadata()**
     23. **' Генерация всех блобов**
     24. **RefreshGallery()**
     25. **End If**
     26. End Sub
     27. ...
     28. End Class
   1. Нажмите **F5** для сборки и запуска приложения.
   2. Загрузите еще несколько изображений из папки **Source\Assets\Images** и нажмите **Copy** на одном или нескольких изображениях для осуществления его копирования.
      1. 
      2. Изображение
      3. Копирование блоба из хранилища
   3. Нажмите **OK** для подтверждения операции копирования. Вы должны увидеть копию изображения с соответственно измененными метаданными для атрибута **ImageName**.
      1. 
      2. Изображение
      3. Проверка
   4. Нажмите **Shift+F5** для остановки отладки и удаления развертывания из эмулятора вычислений

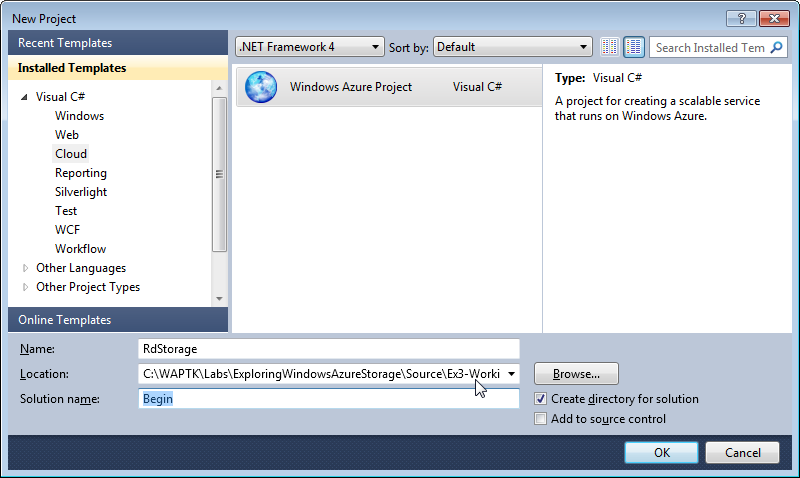
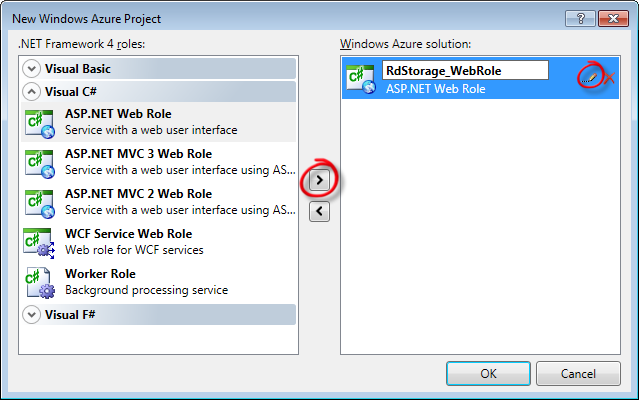
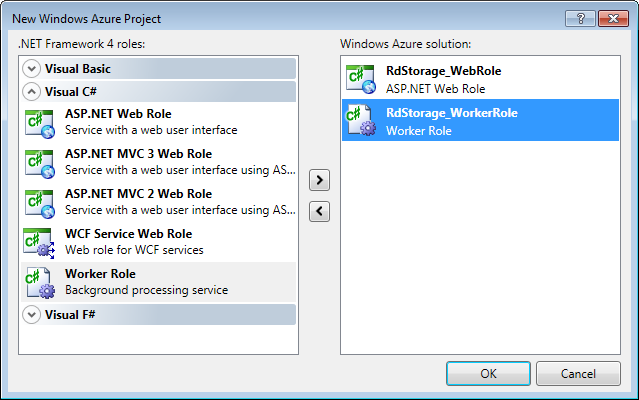
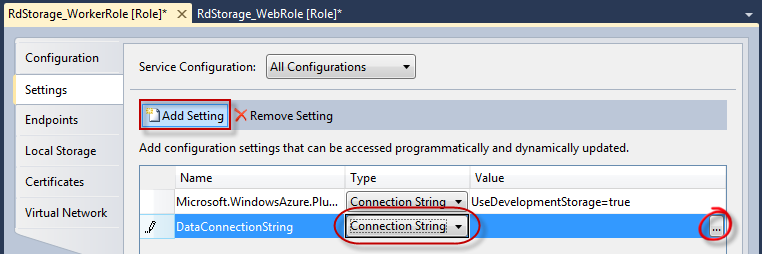
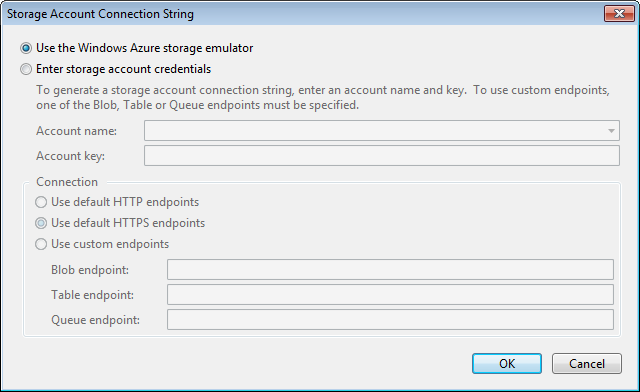
Задание 6 – Создание снапшотов блобов

* 1. Сервис блобов Windows Azure поддерживает создание снапшотов блобов. Разница между снапшотом и копией блоба состоит в том, что снапшоты являются read-only и блоб-источник поддерживает связь со своими снапшотами; копии же можно изменять. После создания снапшота блоба блоб-источник не может быть удален до тех пор, пока существует хотя бы один его снапшот.
  2. В этом задании вы реализуете логику создания снапшота блоба.
  3. Добавьте на представление контрол **asp:LinkButton** для создания снапшотов изображений: откройте страницу **Default.aspx** в режиме Source, **ItemTemplate** для контрола **asp:ListView** и раскомментируйте ASP.NET разметку, расположенную непосредственно после контрола копирования блоба, выделенную **жирным**.
     1. HTML
     2. <div class="item">
     3. <ul style="width:40em;float:left;clear:left" >
     4. <asp:Repeater ID="blobMetadata" runat="server">
     5. <ItemTemplate>
     6. <li><%# Eval("Name") %><span><%# Eval("Value") %></span></li>
     7. </ItemTemplate>
     8. </asp:Repeater>
     9. <li>
     10. <asp:LinkButton ID="deleteBlob"
     11. OnClientClick="return confirm('Delete image?');"
     12. CommandName="Delete"
     13. CommandArgument='<%# Eval("Uri")%>'
     14. runat="server" Text="Delete" oncommand="OnDeleteImage" />
     15. <asp:LinkButton ID="CopyBlob"
     16. OnClientClick="return confirm('Copy image?');"
     17. CommandName="Copy"
     18. CommandArgument='<%# Eval("Uri")%>'
     19. runat="server" Text="Copy" oncommand="OnCopyImage" />
     20. **<asp:LinkButton ID="SnapshotBlob"**
     21. **OnClientClick="return confirm('Snapshot image?');"**
     22. **CommandName="Snapshot"**
     23. **CommandArgument='<%# Eval("Uri")%>'**
     24. **runat="server" Text="Snapshot" oncommand="OnSnapshotImage" />**
     25. </li>
     26. </ul>
     27. <img src="<%# Eval("Uri") %>" alt="<%# Eval("Uri") %>" style="float:left"/>
     28. </div>
  4. Добавьте код, выделенный **жирным**, в файл **Default.aspx.cs** / **Default.aspx.vb** для реализации обработчика контрола *snapshotBlob* **asp:LinkButton**. В коде происходит получение блоба-источника и создание его снапшота.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex02-11-OnSnapshotImageMethod-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. protected void OnSnapshotImage(object sender, CommandEventArgs e)
     7. {
     8. **if (e.CommandName == "Snapshot")**
     9. **{**
     10. **// Получение блоба-источника**
     11. **var blobUri = (string)e.CommandArgument;**
     12. **var srcBlob = this.GetContainer().GetBlobReference(blobUri);**
     13. **// Создание снапшота**
     14. **var snapshot = srcBlob.CreateSnapshot();**
     15. **status.Text = "A snapshot has been taken for image blob:" + srcBlob.Uri + " at " + snapshot.SnapshotTime;**
     16. **RefreshGallery();**
     17. **}**
     18. }
     19. ...
     20. }
     21. (Сниппет – *ExploringWindowsAzureStorage-Ex02-11-OnSnapshotImageMethod-VB*)
     22. Visual Basic
     23. Partial Public Class \_Default
     24. Inherits System.Web.UI.Page
     25. ...
     26. Protected Sub OnSnapshotImage(ByVal sender As Object, ByVal e As CommandEventArgs)
     27. **If e.CommandName = "Snapshot" Then**
     28. **' Получение блоба-источника**
     29. **Dim blobUri = DirectCast(e.CommandArgument, String)**
     30. **Dim srcBlob = Me.GetContainer().GetBlobReference(blobUri)**
     31. **' Создание снапшота**
     32. **Dim snapshot = srcBlob.CreateSnapshot()**
     33. **status.Text = "A snapshot has been taken for image blob:" + srcBlob.Uri.ToString() + " at " + snapshot.SnapshotTime**
     34. **RefreshGallery()**
     35. **End If**
     36. End Sub
     37. ...
     38. End Class
  5. Нажмите F5 для сборки и запуска приложения.
  6. Нажмите **Snapshot** на любом из изображений для создания снапшота.
     1. 
     2. Изображение
     3. Создание снапшота блоба
  7. Нажмите **OK** для подтверждения операции создания снапшота. Вы увидите статус, подтверждающий создание снапшота.
  8. Попытайтесь удалить блоб-источник.
     1. 
     2. Изображение
     3. Ошибка удаления снапшота «Cannot Delete Snapshot»
  9. Вы увидите сообщение о том, что блоб-источник удалить невозможно, так как у него есть снапшоты.
     1. 
     2. Изображение
     3. Ошибка «Cannot Delete Snapshot»
     4. **Примечание:**  Для удаления блоба-источника должны быть сначала удалены все его снапшоты (этой функциональности в приложении нет)

Упражнение 3: Работа с очередями

* 1. В этом упражнении вы создадите простое веб-приложение, отправляющее сообщения в очередь Windows Azure. Worker-роль будет забирать сообщения из очереди и писать их в лог эмулятора вычислений.
  2. Сервис очередей – отличный способ осуществления коммуникаций между Web- и Worker-ролями. Очередь может содержать неограниченное количество сообщений, каждое из которых может быть до 64 Кб в размере. Сообщения помещаются в конец очереди и забираются из ее начала.

Задание 1 – Создание решения

* 1. В этом задании вы создадите и настроите решение для работы с очередями Windows Azure
  2. Откройте в режиме администратора командную строку **Visual Studio Command Prompt**. Для этого: в **Start | All Programs | Microsoft Visual Studio 2010 | Visual Studio Tools | Visual Studio Command Prompt (2010)** щелкните правой кнопкой мыши на ярлыке **Visual Studio 2010 Command Prompt** и выберите **Run as administrator**
  3. В меню **File** нажмите **New** **Project**.
  4. В диалоговом окне **New Project** в списке **Installed Templates** выберите предпочитаемый язык программирования (Visual C# или Visual Basic) и выберите **Cloud.** Выберите **Windows Azure Project**, укажите значение имени проекта **Name** как **RdStorage**, установите расположение проекта в **Source\Ex03-WorkingWithQueues\begin\[CS|VB]** в папке лабораторной, измените имя решения на **begin** и убедитесь, что опция **Create directory for solution** отмечена. Нажмите **OK** для создания проекта.
     1. 
     2. Изображение
     3. Создание проекта Windows Azure
  5. В диалоговом окне **New Windows Azure Project** выберите из списка доступных ролей **ASP.NET Web Role** и нажмите кнопку со стрелкой (>) для добавления экземпляра этой роли в решение. Измените имя роли на **RdStorage\_WebRole**: выберите роль в правой панели, нажмите на иконку ручки и введите новое имя. Не закрывайте диалоговое окно.
     1. 
     2. Изображение
     3. Добавление Web-роли в проект Windows Azure (Visual C#)
     4. **Примечание**: В случае Visual Basic выберите эквивалент.
  6. Добавьте в решение вторую роль: выберите **Worker Role** и измените ее имя на **RdStorage\_WorkerRole**.
     1. 
     2. Изображение
     3. Добавление новой Worker-роли в проект «облачного» сервиса (C#)
     4. **Примечание:** В случае Visual Basic выберите эквивалент.
  7. Нажмите **OK** для закрытия диалогового окна **New Windows Azure Project** и создания решения.
  8. Щелкните правой кнопкой на каждой роли в папке **Roles** в проекте **RdStorage** и выберите **Properties**.
  9. На вкладке **Settings** нажмите **Add Setting** и создайте настройку типа **ConnectionString** с именем *DataConnectionString*. Нажмите кнопку с троеточием и установите опцию **Use storage emulator**. Повторите для каждой роли в проекте.
     1. 
     2. Изображение
     3. Создание строки подключения
     4. 
     5. Изображение
     6. Настройка строки подключения для использования эмулятора вычислений

Задание 2 – Помещение сообщений в очередь

* 1. В этом задании вы создадите веб-приложение **RdStorage\_WebRole,**которое сможет помещать сообщение в очередь.
  2. Откройте в **RdStorage\_WebRole** страницу **Default.aspx.** Удалите содержимое контрола *BodyContent* **Content**.
  3. Добавьте на страницу контрол asp:**TextBox**. Измените **ID** для **TextBox** на **txtMessage**. Сразу после добавленного ранее контрола **TextBox** вставьте контрол asp:**Button**. Измените **ID** контрола кнопки на **btnSend** и определите значение свойства **Text** как **Send message**.
     1. C#
     2. <%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
     3. CodeBehind="Default.aspx.cs" Inherits="RdStorage\_WebRole.\_Default" %>
     4. <asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
     5. </asp:Content>
     6. <asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
     7. **<asp:TextBox ID="txtMessage" runat="server"></asp:TextBox>**
     8. **<asp:Button ID="btnSend" runat="server" Text="Send message" />**
     9. </asp:Content>
     10. Visual Basic
     11. <%@ Page Title="Home Page" Language="vb" MasterPageFile="~/Site.Master" AutoEventWireup="false"
     12. CodeBehind="Default.aspx.vb" Inherits="RdStorage\_WebRole.\_Default" %>
     13. <asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
     14. </asp:Content>
     15. <asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
     16. **<asp:TextBox ID="txtMessage" runat="server"></asp:TextBox>**
     17. **<asp:Button ID="btnSend" runat="server" Text="Send message" />**
     18. </asp:Content>
  4. Откройте файл исходников для страницы **Default.aspx**: щелкните правой кнопкой мыши на **Default.aspx** и нажмите **View Code** (**Default.aspx.cs** / **Default.aspx.vb**).
  5. Добавьте в начало файла пространства имён.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-01-Namespace-CS*)
     2. C#
     3. **using Microsoft.WindowsAzure;**
     4. **using Microsoft.WindowsAzure.StorageClient;**
     5. **using Microsoft.WindowsAzure.ServiceRuntime;**
     6. (Сниппет – *ExploringWindowsAzureStorage-Ex03-01-Namespace-VB*)
     7. Visual Basic
     8. **Imports Microsoft.WindowsAzure**
     9. **Imports Microsoft.WindowsAzure.StorageClient**
     10. **Imports Microsoft.WindowsAzure.ServiceRuntime**
  6. Щелкните правой кнопкой мыши на **Default.aspx**, нажмите **View** **Designer** и щелкните два раза на кнопке **Send message**. Вместо этого вы можете отредактировать разметку вручную и вставить необходимый обработчик событий. Добавьте код, выделенный **жирным**, в обработчик **btnSend\_Click** для инициализации информации аккаунта:
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-02-WebRoleCreateAccount-CS*)
     2. C#
     3. public partial class \_Default : System.Web.UI.Page
     4. {
     5. ...
     6. protected void btnSend\_Click(object sender, EventArgs e)
     7. {
     8. **// инициализация информации аккаунта**
     9. **var storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     10. }
     11. }
     12. (Сниппет – *ExploringWindowsAzureStorage-Ex03-02-WebRoleCreateAccount-VB*)
     13. Visual Basic
     14. Public Class \_Default
     15. Inherits System.Web.UI.Page
     16. ...
     17. Protected Sub btnSend\_Click(ByVal sender As Object, ByVal e As EventArgs) Handles btnSend.Click
     18. **' инициализация информации аккаунта**
     19. **Dim account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
     20. End Sub
     21. End Class
  7. Добавьте выделенный **жирным** код для создания экземпляра **QueueStorage** и очереди сообщений в том случае, если она не существует.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-03-WebRoleCreateQueue-CS*)
     2. C#
     3. protected void btnSend\_Click(object sender, EventArgs e)
     4. {
     5. ...
     6. // Код, вставленный ранее
     7. **// получение ссылки на очередь *messages***
     8. **var queueClient = storageAccount.CreateCloudQueueClient();**
     9. **var queue = queueClient.GetQueueReference("messagequeue");**
     10. **queue.CreateIfNotеxist();**
     11. }
     12. (Сниппет – *ExploringWindowsAzureStorage-Ex03-03-WebRoleCreateQueue-VB*)
     13. Visual Basic
     14. Protected Sub btnSend\_Click(ByVal sender As Object, ByVal e As EventArgs) Handles btnSend.Click

...

' Код, вставленный ранее

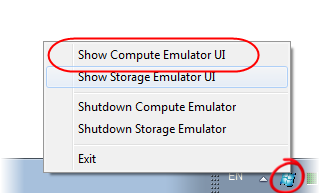
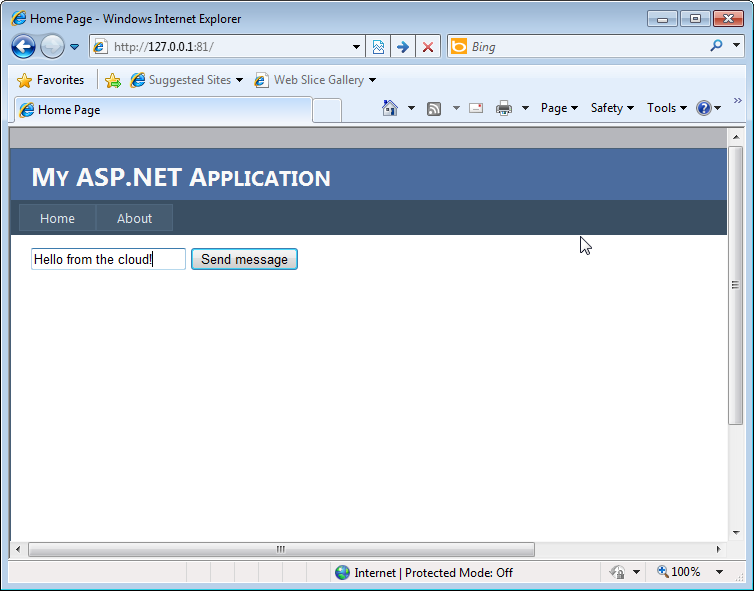
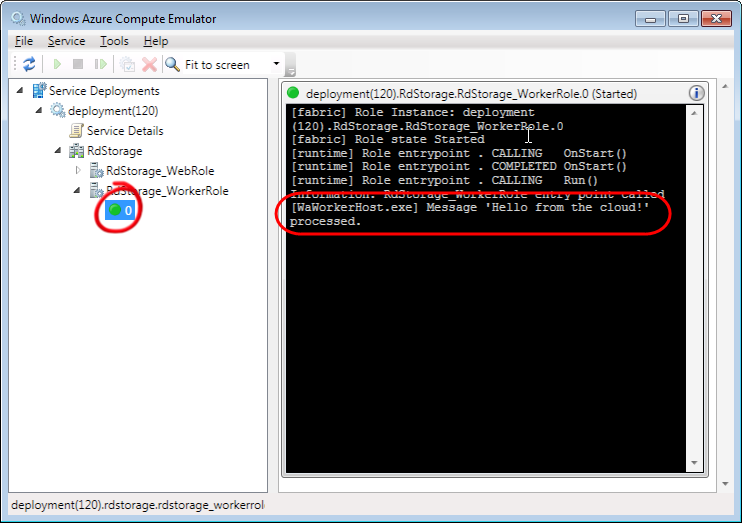
**' получение ссылки на очередь *messages***

**Dim queueClient = account.CreateCloudQueueClient()**

* + 1. **Dim queue As CloudQueue = queueClient.GetQueueReference("messagequeue")**
    2. **queue.CreateIfПримечаниеxist()**
    3. End Sub
  1. Добавьте выделенный **жирным** код для добавления возможности помещения сообщения в очередь.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-04-WebRoleAddMessage-CS*)
     2. C#
     3. protected void btnSend\_Click(object sender, EventArgs e)
     4. {
     5. ...
     6. // Код, вставленный ранее
     7. **// добавление сообщения в очередь**
     8. **var msg = new CloudQueueMessage(txtMessage.Text);**
     9. **queue.AddMessage(msg);**
     10. **txtMessage.Text = string.Empty;**
     11. }
     12. (Сниппет – *ExploringWindowsAzureStorage-Ex03-04-WebRoleAddMessage-VB*)
     13. Visual Basic
     14. Protected Sub btnSend\_Click(ByVal sender As Object, ByVal e As EventArgs) Handles btnSend.Click
     15. ...
     16. ' Код, вставленный ранее
     17. **' добавление сообщения в очередь**
     18. **Dim msg As New CloudQueueMessage(Me.txtMessage.Text)**
     19. **queue.AddMessage(msg)**
     20. **Me.txtMessage.Text = String.Empty**
     21. End Sub
  2. Откройте файл **WebRole.cs** / **WebRole.vb** из проекта **RDStorage\_WebRole**.
  3. Добавьте необходимые пространства имён в начало файла.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-05-Namespace-CS*)
     2. C#
     3. **using Microsoft.WindowsAzure.StorageClient;**
     4. (Сниппет – *ExploringWindowsAzureStorage-Ex03-05-Namespace-VB*)
     5. Visual Basic
     6. **Imports Microsoft.WindowsAzure.StorageClient**

Задание 3 – Получение сообщений из очереди

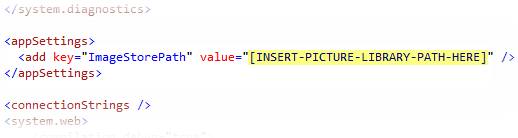
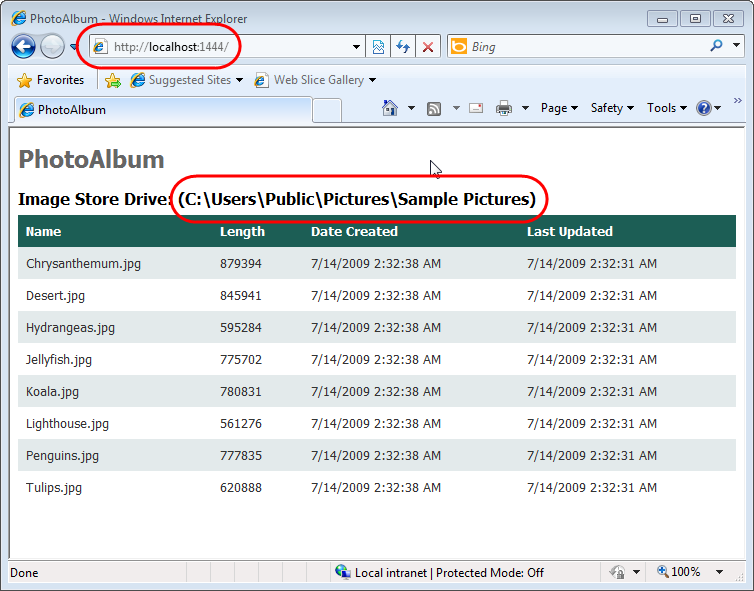
* 1. В этом задании вы добавите в worker-роль возможность «забора» сообщений из очереди и записи их в лог эмулятора вычислений.
  2. Откройте файл **Global.asax.cs** / **Global.asax.vb** из проекта **RdStorage\_WebRole**.
  3. Добавьте выделенный **жирным** код в метод **Application\_Start** для инициализации информации аккаунта.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-06-InitializeAccount-CS*)
     2. C#
     3. void Application\_Start(object sender, EventArgs e)
     4. {
     5. **CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =>**
     6. **{**
     7. **configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));**
     8. **});**
     9. }
     10. (Сниппет – *ExploringWindowsAzureStorage-Ex03-06-InitializeAccount-VB*)
     11. Visual Basic
     12. Sub Application\_Start()
     13. **CloudStorageAccount.SetConfigurationSettingPublisher(Sub(configName, configSetter)**
     14. **configSetter(RoleEnvironment.GetConfigurationSettingValue(configName))**
     15. **End Sub)**
     16. End Sub
  4. Убедитесь в наличии необходимых пространств имен в начале файлов.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-07-Namespace-CS*)
     2. C#
     3. **using Microsoft.WindowsAzure;**
     4. **using Microsoft.WindowsAzure.ServiceRuntime;**
     5. (Сниппет – *ExploringWindowsAzureStorage-Ex03-07-Namespace*-VB)
     6. Visual Basic
     7. **Imports Microsoft.WindowsAzure**
     8. **Imports Microsoft.WindowsAzure.ServiceRuntime**
  5. Откройте файл **WorkerRole.cs** / **WorkerRole.vb** из проекта **RdStorage\_WorkerRole**.
  6. Добавьте выделенный **жирным** код в метод **OnStart** для инициализации информации аккаунта.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-08-InitializeAccount-CS*)
     2. C#
     3. public override bool OnStart()
     4. {
     5. // Определение максимального количества одновременных соединений
     6. ServicePointManager.DefaultConnectionLimit = 12;
     7. // Подробная информация по обработке изменений конфигурации http://go.microsoft.com/fwlink/?LinkId=166357.
     9. **CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =>**
     10. **{**
     11. **configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));**
     12. **});**
     13. return base.OnStart();
     14. }
     15. (Сниппет – *ExploringWindowsAzureStorage-Ex03-08-InitializeAccount-VB*)
     16. Visual Basic
     17. Public Overrides Function OnStart() As Boolean
     18. ' Определение максимального количества одновременных соединений
     19. ServicePointManager.DefaultConnectionLimit = 12
     20. ' Подробная информация по обработке изменений конфигурации http://go.microsoft.com/fwlink/?LinkId=166357.
     22. **CloudStorageAccount.SetConfigurationSettingPublisher(Sub(configName, configSetter)**
     23. **configSetter(RoleEnvironment.GetConfigurationSettingValue(configName))**
     24. **End Sub)**
     25. Return MyBase.OnStart()
     26. End Function
  7. Убедитесь в присутствии необходимых пространств имен в начале файла.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-09-Namespace-CS*)
     2. C#
     3. **using System.Diagnostics;**
     4. **using System.Threading;**
     5. **using Microsoft.WindowsAzure;**
     6. **using Microsoft.WindowsAzure.Diagnostics;**
     7. **using Microsoft.WindowsAzure.ServiceRuntime;**
     8. **using Microsoft.WindowsAzure.StorageClient;**
     9. (Сниппет – *ExploringWindowsAzureStorage-Ex03-09-Namespace*-VB)
     10. Visual Basic
     11. **Imports System.Diagnostics**
     12. **Imports System.Threading**
     13. **Imports Microsoft.WindowsAzure**
     14. **Imports Microsoft.WindowsAzure.Diagnostics**
     15. **Imports Microsoft.WindowsAzure.ServiceRuntime**
     16. **Imports Microsoft.WindowsAzure.StorageClient**
  8. В методе **Run** создайте экземпляр **QueueStorage** и получите ссылку на очередь *messages:* добавьте выделенный **жирным** код и удалите ~~перечеркнутый~~ код, симулирующий задержку worker-роли.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-10-WorkerGetQueue-CS*)
     2. C#
     3. public override void Run()
     4. {
     5. // Пример реализации Worker-роли. Замените на вашу логику.
     6. Trace.WriteLine("RdStorage\_WorkerRole entry point called", "Information");
     7. ~~while (true)~~
     8. ~~{~~
     9. ~~Thread.Sleep(10000);~~
     10. ~~Trace.WriteLine("Working", "Information");~~
     11. ~~}~~
     12. **// инициализация информации аккаунта**
     13. **var storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     14. **// получение ссылки на очередь messages**
     15. **var queueClient = storageAccount.CreateCloudQueueClient();**
     16. **var queue = queueClient.GetQueueReference("messagequeue");**
     17. }
     18. (Сниппет – *ExploringWindowsAzureStorage-Ex03-10-WorkerGetQueue-VB*)
     19. Visual Basic
     20. Public Overrides Sub Run()
     21. ' Пример реализации Worker-роли. Замените на вашу логику.
     22. Trace.WriteLine("RdStorage\_WorkerRole entry point called.", "Information")
     23. ~~While (True)~~
     24. ~~Thread.Sleep(10000)~~
     25. ~~Trace.WriteLine("Working", "Information")~~
     26. ~~End While~~
     27. **' инициализация информации аккаунта**
     28. **Dim storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
     29. **' получение ссылки на очередь messages**
     30. **Dim queueClient As CloudQueueClient = storageAccount.CreateCloudQueueClient()**
     31. **Dim queue = queueClient.GetQueueReference("messagequeue")**
     32. End Sub
  9. Добавьте выделенный **жирным** код, реализующий логику получения сообщений и записи их в лог эмулятора вычислений. После записи в лог сообщение удаляется из очереди.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex03-11-WorkerGetMessages-CS*)
     2. C#
     3. public override void Run()
     4. {
     5. ...
     6. // Код, вставленный ранее
     7. **// получение сообщений и запись их в лог эмулятора вычислений**
     8. **while (true)**
     9. **{**
     10. **Thread.Sleep(10000);**
     11. **if (queue.Exists())**
     12. **{**
     13. **var msg = queue.GetMessage();**
     14. **if (msg != null)**
     15. **{**
     16. **Trace.TraceInformation(string.Format("Message '{0}' processed.", msg.AsString));**
     17. **queue.DeleteMessage(msg);**
     18. **}**
     19. **}**
     20. **}**
     21. }
     22. (Сниппет – *ExploringWindowsAzureStorage-Ex03-11-WorkerGetMessages-VB*)
     23. Visual Basic
     24. Public Overrides Sub Run()
     25. ...
     26. ' Код, вставленный ранее
     27. **' получение сообщений и запись их в лог эмулятора вычислений**
     28. **While (True)**
     29. **Thread.Sleep(10000)**
     30. **If queue.Exists() Then**
     31. **Dim msg = queue.GetMessage()**
     32. **If (msg IsNot Nothing) Then**
     33. **Trace.TraceInformation(String.Format("Message '{0}' processed.", msg.AsString))**
     34. **queue.DeleteMessage(msg)**
     35. **End If**
     36. **End If**
     37. **End While**
     38. End Sub
     39. **Примечание:** Процесс Worker-а пытается получить сообщение из очереди каждые 10 секунд, используя метод **GetMessage**. Если в очереди содержатся сообщения, они забираются и записываются в лог эмулятора вычислений.

1. Проверка
   1. Для проверки вашего сервиса, работающего в эмуляторе вычислений:
   2. В Visual Studio нажмите F5 для сборки и запуска приложения.
   3. Откройте пользовательский интерфейс эмулятора вычислений: щелкните правой кнопкой мыши на иконке эмулятора в системном трее и **Show Compute Emulator UI** (иконка лазурного цвета).
      1. 
      2. Изображение
      3. Пользовательский интерфейс эмулятора вычислений
   4. Разверните дерева и найдите лог экземпляра **Worker**.
   5. Переключитесь в Windows Internet Explorer. Убедитесь, что открылась стандартная страница, введите сообщение и нажмите **Send message**.
      1. 
      2. Изображение
      3. Стандартная веб-страница
   6. Переключитесь в пользовательский интерфейс эмулятора вычислений. Вы должны увидеть в логе Worker-роли введенное только что сообщение.
      1. 
      2. Изображение
      3. Лог Worker-роли с появившимся сообщением
      4. **Примечание:** Появление сообщения в логе может занять несколько секунд в связи с временем «сна» Worker-роли.

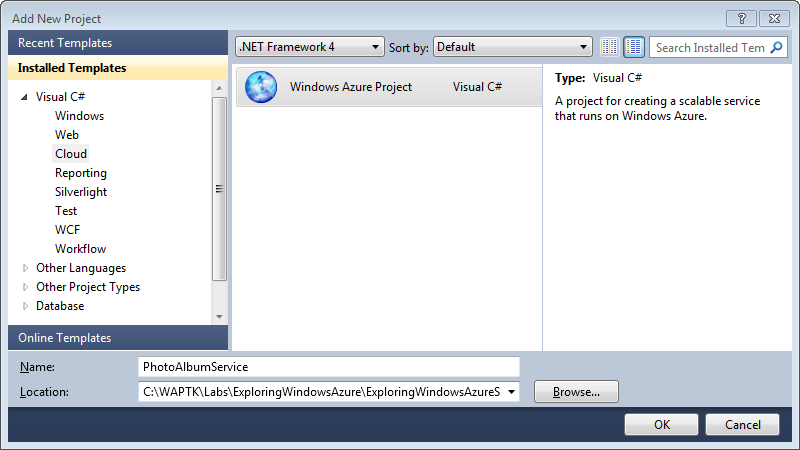
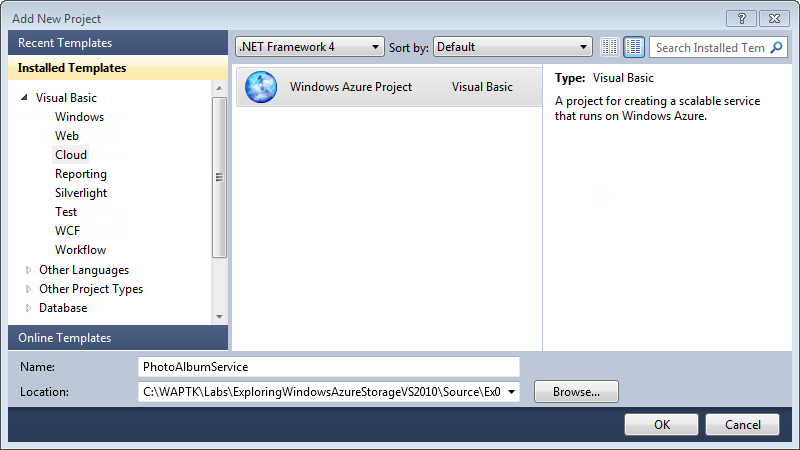
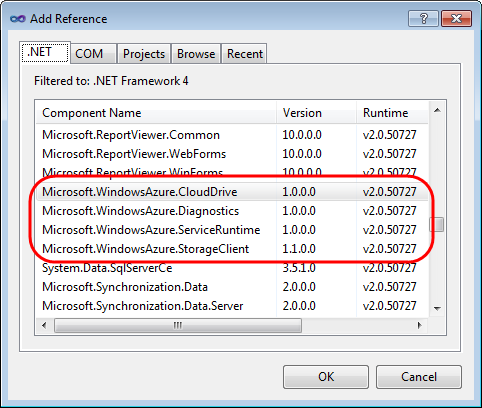
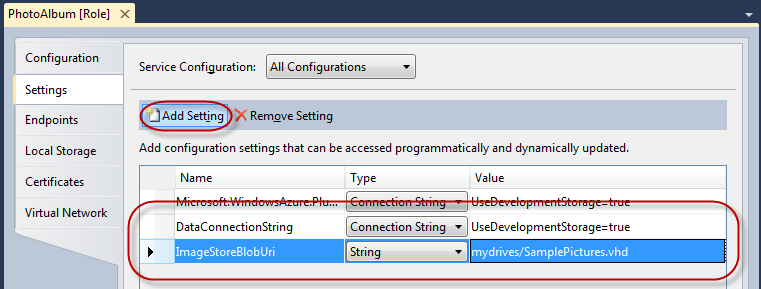
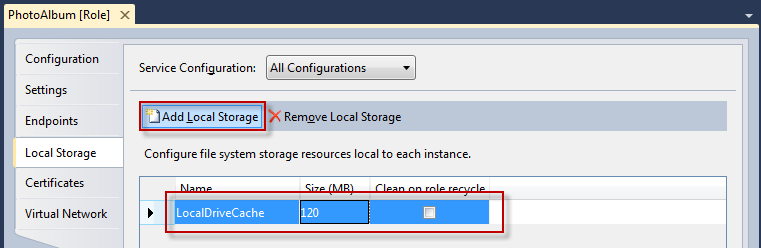
Упражнение 4: Работа с дисками

* 1. Диск Windows Azure представляет собой файл NTFS VHD, хранящийся в страничном блобе. Вы можете примонтировать этот VHD в экземпляр вычислительного сервиса для предоставления долговечного хранилища, доступного приложениям через файловую систему Windows. Содержимое диска Azure будет сохранено даже при очистке экземпляра, к которому примонтирован диск.
  2. В этом упражнении вы возьмете за основу существующее приложение, использующее обычные API файловой системы Windows для доступа к информации, хранящейся на локальном диске и запущенное как сервис Windows Azure.
  3. В первой части упражнения вы запустите приложение на сервере разработки ASP.NET для знакомства с его функциями. Далее вы создадите проект «облачных» сервисов как Web-роль, и запустите его в эмуляторе вычислений с использованием эмуляции дисков Azure. Вы также создадите на локальной машине VHD, загрузите его в хранилище блобов, развернете приложение в Windows Azure и примонтируете диск в экземпляре Windows Azure.

Задание1 – Приложение PhotoAlbum

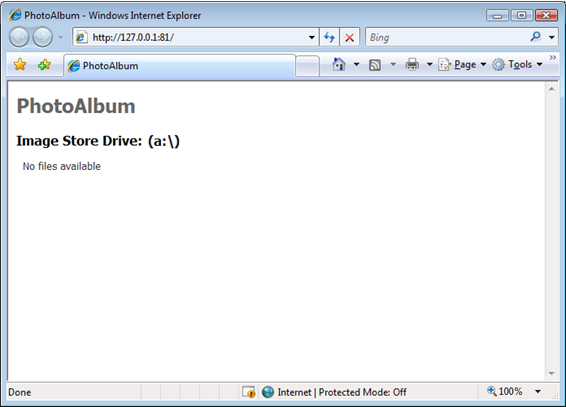
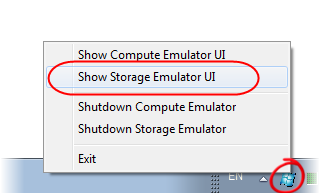
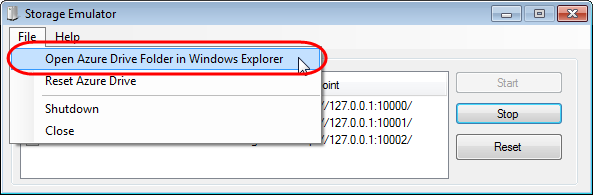
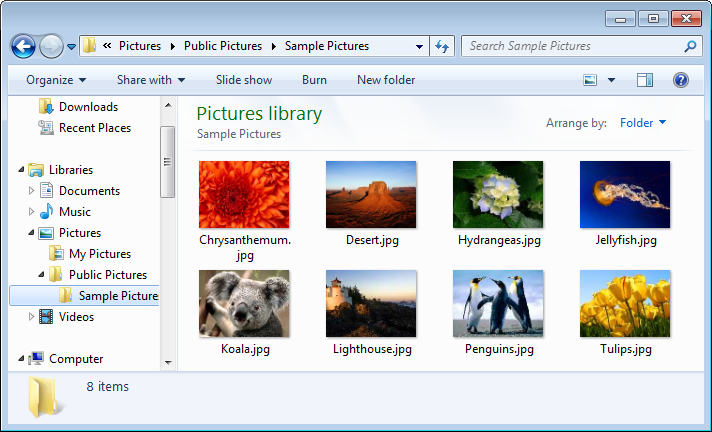
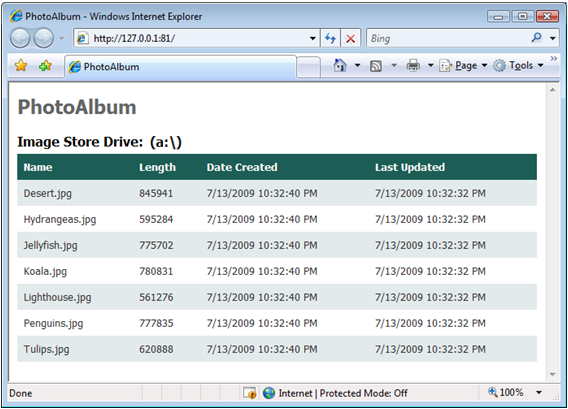
* 1. PhotoAlbum является приложением-примером, использующим стандартные API файловой системы для получения содержимого хранилища изображений. В этом задании вы получите представление о функциональности приложения и настроите расположение хранилища изображений для хранения данных в папке на вашей машине для локального тестирования приложения.
  2. Запустите Microsoft Visual Studio 2010 в режиме администратора. Для этого в **Start | All Programs | Microsoft Visual Studio 2010** щелкните правой кнопкой мыши на ярлыке **Microsoft Visual Studio 2010** и выберите **Run as administrator**.
     1. **Примечание:** Использовать режим администратора при запуске приложения на сервере разработки ASP.NET не нужно. Однако он необходим при запуске приложения в эмуляторе вычислений.
  3. Если появится диалоговое окно **User Account Control**, нажмите **Continue**.
  4. В меню **File** нажмите **Open** и **Project/Solution**. В диалоговом окне **Open Project** перейдите в to **\Source\Ex04-WorkingWithDrives\begin** в папке лабораторной **Source**, выберите соответствующий вашему языку программирования (Visual C#/Visual Basic) **PhotoAlbum.sln** и нажмите **Open.**
  5. Просмотрите файлы приложения. Щелкните два раза на файле **Default.aspx**. Обратите внимание, что на странице для отображения списка хранящихся в хранилище изображений используется контрол **GridView**, источником данных для которого является контрол **LinqDataSource**, «привязанный» к объекту контекста **PhotoAlbumDataSource**.
  6. Откройте файл **PhotoAlbumDataSource.cs** ( **PhotoAlbumDataSource.vb)** для просмотра класса контекста, используемого **LinqDataSource**. Обратите внимание на свойство **File**, используемое контролом источника данных – оно возвращает коллекцию [FileInfo](http://msdn.microsoft.com/en-us/library/system.io.fileinfo.aspx) и использует стандартные API файловой системы для определения файлов PNG и JPEG в хранилище изображений.
  7. Откройте файл **Global.asax** и обратите внимание на свойство **ImageStorePath** , содержащее расположение папки хранилища изображений и обработчик событий **Application\_Start** , инициализирующий значение свойства **ImageStorePath** как путь, полученный из настроек приложения.
  8. Перед запуском приложения вам необходимо настроить расположение хранилища изображений. Щелкните два раза на файле **Web.config**. В секции **appSettings** найдите настройку *ImageStorePath* и обновите ее значение на путь к папке **Sample Pictures** на вашей машине.
     1. 
     2. Изображение
     3. Настройка расположения хранилища изображений
     4. **Примечание:** Библиотека стандартных рисунков доступна в большинстве установок Windows по пути “*%PUBLIC%\Pictures\Sample Pictures”*. Библиотека содержит небольшой набор изображений. Если в вашей среде нет этой папки, вы можете заменить ее на любую папку, содержащую коллекцию изображений в форматах JPEG и PNG, например, папку *Source\Assets\Images* в папке лабораторной.
     5. Обратите внимание, что *%PUBLIC%* является переменной окружения, ведущей в в папку *Public* в папке пользовательских профилей. Вы должны написать полный путь к этой папке при настройке *ImageStorePath* (например, *C:\Users\Public\Pictures\Sample Pictures*).
  9. Нажмите **F5** для сборки и запуска приложения. Обратите внимание, что стандартная страница показывает список файлов из хранилища изображений, и путь к хранилищу изображений ведет на папку на вашей машине, настроенный в файле конфигурации приложения.
     1. 
     2. Изображение
     3. Запуск приложения локально
  10. Закройте окно браузера. Вы узнали, что приложение использует для доступа к файлам в хранилище стандартные API файловой системы. В следующем задании вы обновите приложение до использования его как «облачного» сервиса.

Задание 2 – Перенос приложения в Windows Azure с использованием дисков

* 1. При переносе приложений в Windows Azure обычным способом является перенос хранилища изображений в хранилище блобов. Приложение ожидает, что изображения будут храниться в файловой системе. Вы всегда можете изменить код доступа к изображениям в хранилище на использование API сервиса блобов. Для простых приложений это не будет сложной задачей, но представляет собой проблему при более сложных приложениях. Использование дисков Windows Azure позволяет перенести приложение в «облако» без изменений кода (кроме кода монтирования диска страничного блоба).
  2. В этом задании вы обновите приложение до использования его как «облачного» сервиса Windows Azure и использования диска Windows Azure в качестве хранилища изображений.
  3. Добавьте проект «облачного» сервиса в решение: в **Solution Explorer**, щелкните правой кнопкой мыши на корне решения, выберите **Add** и **New Project.**
  4. В диалоговом окне **Add** **New Project** в **Installed Templates** разверните ветку предпочитаемого языка программирования (Visual C# или Visual Basic) и выберите **Cloud**. Выберите шаблон **Windows Azure Project**, укажите значение имени проекта **Name** как **PhotoAlbumService**, остальное оставьте по умолчанию и нажмите **OK**.
     1. 
     2. Изображение
     3. Создание проекта Windows Azure (Visual C#)
     4. 
     5. Изображение
     6. Создание проекта Windows Azure (Visual Basic)
  5. В диалоговом окне **New Windows Azure Project** нажмите **OK**, не добавляя новых ролей – в качестве web-роли вы будете использовать существующее приложение.
  6. Добавьте ссылки на сборки Windows Azure: в **Solution Explorer** щелкните правой кнопкой мыши на проекте **PhotoAlbum** и выберите **Add Reference**. В диалоговом окне **Add Reference** переключитесь на вкладку **.NET**, выберите сборки **Microsoft.WindowsAzure.CloudDrive**, **Microsoft.WindowsAzure.Diagnostics**, **Microsoft.WindowsAzure.ServiceRuntime**, и **Microsoft.WindowsAzure.StorageClient** и нажмите **OK**.
     1. 
     2. Изображение
     3. Добавление ссылок на сборки Windows Azure
  7. В **Solution Explorer** щелкните правой кнопкой мыши на **Roles** в проекте **PhotoAlbumService**, выберите **Add** и **Web Role Project in solution**.
  8. В диалоговом окне **Associate with Role Project** выберите проект **PhotoAlbum** и нажмите **OK**.
  9. Теперь необходимо настроить Web-роль: щелкните два раза на роли **PhotoAlbum** в **Roles** в проекте **PhotoAlbumService**.
  10. В окне свойств **PhotoAlbum [Role]** переключитесь на вкладку **Settings** и нажмите **Add Setting**. Определите значение имени новой настройки **Name** как *DataConnectionString*, **Type** как *Connection String* и нажмите кнопку с троеточием справа от столбца **Value**. В диалоговом окне **Storage Connection String** укажите **Use storage emulator** и нажмите **OK**.
  11. Добавьте вторую настройку для URL «облачного» диска в хранилище блобов. Укажите значение **Name** как *ImageStoreBlobUri*, **Type** как *String*, **Value** как “*mydrives/SamplePictures.vhd*”.
      1. **Примечание:**  *ImageStoreBlobUri* определяет URI сервиса блобов и является чувствительной к регистру символов. Убедитесь, что вы ввели всё так, как показано.
      2. 
      3. Изображение
      4. Определение настроек конфигурации Web-роли
  12. Переключитесь на вкладку **Local Storage** и нажмите **Add Local Storage**. Определите значение имени новой настройки **Name** как *LocalDriveCache*, **Size** как *120*, **Clean on Role Recycle** оставьте без изменений (неотмеченной).
      1. 
      2. Изображение
      3. Настройка локального хранилища для Web-роли для кэширования содержимого диска Azure
      4. **Примечание:** При неотмеченной опции **Clean on Role Recycled** содержимое кэша будет сохранено при очистке экземпляра роли.
      5. В связи с особым методом выделения ресурсов локального хранилища системой часть запрошенного места будет недоступна для использования кэша. Начиная с Windows Azure Guest OS 1.8, весь кэш дисков Windows Azure размещается предварительно, и если локальных ресурсов недостаточно для предварительного размещения кэша, вызов Mount приведет к ошибке. Во избежание этой проблемы при настройке размера локальных ресурсов для кэша добавьте к необходимому размеру еще 20Мб. Например, используйте для кэша размер 1000Мб значение 1020Мб.
  13. Нажмите **CTRL+S** для сохранения изменений.
  14. Настройте «слушателя» для вывода диагностической информации в лог Windows Azure: щелкните два раза на файле **Web.config** и вставьте секцию **system.diagnostics**, как показано ниже.
      1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-01-DiagnosticMonitorTraceListener*)
      2. XML
      3. <?xml version="1.0"?>
      4. <configuration>
      5. **<system.diagnostics>**
      6. **<trace>**
      7. **<listeners>**
      8. **<add type="Microsoft.WindowsAzure.Diagnostics.DiagnosticMonitorTraceListener, Microsoft.WindowsAzure.Diagnostics, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"**
      9. **name="AzureDiagnostics">**
      10. **<filter type="" />**
      11. **</add>**
      12. **</listeners>**
      13. **</trace>**
      14. **</system.diagnostics>**
      15. ...
      16. </configuration>
  15. Добавьте в проект Web-роли класс для инициализации и выключения: щелкните правой кнопкой мыши на проекте **PhotoAlbum**, выберите **Add** и **Existing Item**. В диалоговом окне **Add Existing Item** перейдите в папку **Ex04-WorkingWithDrives\Assets**, выберите **WebRole.cs** ( **WebRole.vb)** и нажмите **Add**.
      1. **Примечание:** Файл **WebRole(.cs|.vb)** содержит стандартный класс, наследуемый от **RoleEntryPoint**, похожий на то, что создается при выборе шаблона проекта Web-роли Windows Azure в Visual Studio.
  16. Вставьте в метод **Application\_Start** в файле **Global.asax.cs** / **Global.asax.vb** код, выделенный **жирным**.
      1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-02-ApplicationStartMethod-CS*)
      2. C#
      3. protected void Application\_Start(object sender, EventArgs e)
      4. {
      5. if (imageStorePath == null)
      6. {
      7. ImageStorePath = WebConfigurationManager.AppSettings["ImageStorePath"];
      8. }
      9. **// Получение доступа к настройке хранилища в аккаунте через publisher**
      10. **CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =>**
      11. **{**
      12. **string connectionString = RoleEnvironment.GetConfigurationSettingValue(configName);**
      13. **configSetter(connectionString);**
      14. **});**
      15. **try**
      16. **{**
      17. **// Инициализация локального кэша для диска Azure**
      18. **LocalResource cache = RoleEnvironment.GetLocalResource("LocalDriveCache");**
      19. **CloudDrive.InitializeCache(cache.RootPath + "cache", cache.MaximumSizeInMegabytes);**
      20. **// Получение аккаунта хранилища**
      21. **CloudStorageAccount account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
      22. **// Получение URI страничного блоба, содержащего «облачный» диск, из настроек конфигурации**
      23. **string imageStoreBlobUri = RoleEnvironment.GetConfigurationSettingValue("ImageStoreBlobUri");**
      24. **// Размонтирование примонтированных ранее дисков.**
      25. **foreach (var drive in CloudDrive.GetMountedDrives())**
      26. **{**
      27. **var mountedDrive = account.CreateCloudDrive(drive.Value.PathAndQuery);**
      28. **mountedDrive.Unmount();**
      29. **}**
      30. **// Создание диска Windows Azure и соответствующего ему страничного блоба**
      31. **CloudDrive imageStoreDrive = account.CreateCloudDrive(imageStoreBlobUri);**
      32. **try**
      33. **{**
      34. **imageStoreDrive.Create(16);**
      35. **}**
      36. **catch (CloudDriveException)**
      37. **{**
      38. **// Диск уже существует**
      39. **}**
      40. **// Примонтирование диска и инициализация приложения путём к хранилищу изображений на диске Azure**
      41. **Global.ImageStorePath = imageStoreDrive.Mount(cache.MaximumSizeInMegabytes / 2, DriveMountOptions.None);**
      42. **}**
      43. **catch (CloudDriveException driveException)**
      44. **{**
      45. **Trace.WriteLine("Error: " + driveException.Message);**
      46. **}**
      47. }
      48. (Сниппет – *ExploringWindowsAzureStorage-Ex04-02-ApplicationStartMethod-VB*)
      49. Visual Basic
      50. Protected Sub Application\_Start(sender As Object, e As EventArgs)
      51. If ImageStorePath Is Nothing Then
      52. ImageStorePath = WebConfigurationManager.AppSettings("ImageStorePath")
      53. End If
      54. **' Получение доступа к настройке хранилища в аккаунте через publisher**
      55. **CloudStorageAccount.SetConfigurationSettingPublisher(Function(configName, configSetter) ConfigurationSettingPublisher(configName, configSetter))**
      56. **Try**
      57. **' Инициализация локального кэша для диска Azure**
      58. **Dim cache As LocalResource = RoleEnvironment.GetLocalResource("LocalDriveCache")**
      59. **CloudDrive.InitializeCache(cache.RootPath & "cache", cache.MaximumSizeInMegabytes)**
      60. **' Получение аккаунта хранилища**
      61. **Dim account As CloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
      62. **' Получение URI страничного блоба, содержащего «облачный» диск, из настроек конфигурации**
      63. **Dim imageStoreBlobUri As String = RoleEnvironment.GetConfigurationSettingValue("ImageStoreBlobUri")**
      64. **' Размонтирование примонтированных ранее дисковРазмонтирование примонтированных ранее дисков**
      65. **For Each drive In CloudDrive.GetMountedDrives()**
      66. **Dim mountedDrive = account.CreateCloudDrive(drive.Value.PathAndQuery)**
      67. **mountedDrive.Unmount()**
      68. **Next**
      69. **' Создание диска Windows Azure и соответствующего ему страничного блоба**
      70. **Dim imageStoreDrive As CloudDrive = account.CreateCloudDrive(imageStoreBlobUri)**
      71. **Try**
      72. **imageStoreDrive.Create(16)**
      73. **Catch e1 As CloudDriveException**
      74. **' Диск уже существует**
      75. **End Try**
      76. **' Примонтирование диска и инициализация приложения путём к хранилищу изображений на диске Azure**
      77. **Global\_asax.ImageStorePath = imageStoreDrive.Mount(cache.MaximumSizeInMegabytes / 2, DriveMountOptions.None)**
      78. **Catch driveException As CloudDriveException**
      79. **Trace.WriteLine("Error: " & driveException.Message)**
      80. **End Try**
      81. End Sub

**Примечание:** В коде выше получается путь к локальному хранилищу, определенный ранее при настройке Web-роли, после чего этот путь используется для инициализации кэша диска и определения максимального количества емкости локального хранилища, которое он может использовать. Далее создаётся объект **CloudDrive** с указанием URL страничного блоба, определенного ранее в настройках конфигурации роли. После этого форматированный страничный блоб монтируется к определенной букве диска для использования приложением Windows Azure.

Обратите внимание, что кэш, назначенный для использования дисков, составляет половину зарезервированного для кэша места. Далее вы создадите второй диск и назначите для него оставшееся место.

* 1. Если вы используете VB, вам необходимо добавить в конец файла **Global.asax.vb** код, выделенный **жирным.**
     1. (Сниппет – ExploringWindowsAzureStorage-Ex04-03-ConfigurationSettingPublisher-VB)
     2. Visual Basic
     3. Public Class Global\_asax
     4. Inherits System.Web.HttpApplication
     5. . . .
     6. **Private Function ConfigurationSettingPublisher(ByVal configName As Object, ByVal configSetter As Func(Of String, Boolean)) As Object**
     7. **Dim connectionString As String = RoleEnvironment.GetConfigurationSettingValue(configName)**
     8. **configSetter(connectionString)**
     9. **Return Nothing**
     10. **End Function**
     11. End Class
  2. Убедитесь в присутствии в файлах **Global.asax.cs** / **Global.asax.vb** необходимых пространств имён.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-04-GlobalNamespace-CS*)
     2. C#
     3. **using System.Diagnostics;**
     4. **using Microsoft.WindowsAzure;**
     5. **using Microsoft.WindowsAzure.ServiceRuntime;**
     6. **using Microsoft.WindowsAzure.StorageClient;**
     7. (Сниппет – *ExploringWindowsAzureStorage-Ex04-04-GlobalNamespace-VB*)
     8. Visual Basic
     9. **Imports System.Diagnostics**
     10. **Imports Microsoft.WindowsAzure**
     11. **Imports Microsoft.WindowsAzure.ServiceRuntime**
     12. **Imports Microsoft.WindowsAzure.StorageClient**
  3. Вставьте выделенный код в начало метода **Application\_End** – он необходим для размонтирования диска Windows Azure при выключении Web-роли.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-05-ApplicationEndMethod-CS*)
     2. C#
     3. protected void Application\_End(object sender, EventArgs e)
     4. {
     5. **// Получение ссылки на «облачный» диск и его размонтирование**
     6. **CloudStorageAccount account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**
     7. **string imageStoreBlobUri = RoleEnvironment.GetConfigurationSettingValue("ImageStoreBlobUri");**
     8. **CloudDrive imageStoreDrive = account.CreateCloudDrive(imageStoreBlobUri);**
     9. **imageStoreDrive.Unmount();**
     10. }
     11. (Сниппет – *ExploringWindowsAzureStorage-Ex04-05-ApplicationEndMethod-VB*)
     12. Visual Basic
     13. Sub Application\_End(ByVal sender As Object, ByVal e As EventArgs)
     14. **' Получение ссылки на «облачный» диск и его размонтирование**
     15. **Dim account As CloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**
     16. **Dim imageStoreBlobUri As String = RoleEnvironment.GetConfigurationSettingValue("ImageStoreBlobUri")**
     17. **Dim imageStoreDrive As CloudDrive = account.CreateCloudDrive(imageStoreBlobUri)**
     18. **imageStoreDrive.Unmount()**
     19. End Sub
     20. **Примечание:** В коде получается ссылка на примонтированный ранее «облачный» диск и далее происходит его размонтирование.
  4. Приложение готово к запуску в качестве сервиса Windows Azure. Нажмите **F5** для сборки и запуска приложения в эмуляторе вычислений. При запуске приложения оно покажет содержимое диска Windows Azure, который пока пуст. Обратите внимание, что диск хранилища изображений примонтирован к букве диска. Оставьте окно браузера открытым.
     1. 
     2. Изображение
     3. Запуск приложения в эмуляторе вычислений с пустым хранилищем
  5. Определите расположение папки, используемой эмулятором хранилища для эмуляции «облачного» диска. Для перехода в пользовательский интерфейс эмулятора вычислений щелкните правой кнопкой мыши на иконке Windows Azure и нажмите **Show Storage Emulator UI**.
     1. 
     2. Изображение
     3. Пользовательский интерфейс эмулятора
  6. В пользовательском интерфейс в меню **File** нажмите **Open Azure Drive Folder in Windows Explorer**.
     1. 
     2. Изображение
     3. Открытие папки диска Azure
     4. **Примечание:** При локальном запуске эмулятор хранилища не использует сервис блобов для эмуляции «облачного» диска, вместо этого диск «привязывается» к локальной папке. Вы можете открыть эту временную папку из пользовательского интерфейса эмулятора хранилища.
  7. В папке диска Azure перейдите к ***devstoreaccount1\mydrives\SamplePictures.vhd***. Обратите внимание, что этот путь соответствует URI блоба – отсюда и ***devstoreaccount1***—который вы определили для хранилища.
  8. В меню **Start** выберите **Pictures** и щелкните два раза на папке **Sample Pictures**.
     1. 
     2. Изображение
     3. Библиотека стандартных изображений Windows
     4. **Примечание:** Библиотека стандартных рисунков доступна в большинстве установок Windows по пути “*%PUBLIC%\Pictures\Sample Pictures”*. Библиотека содержит небольшой набор изображений. Если в вашей среде нет этой папки, вы можете использовать любую другую, содержащую коллекцию изображений в форматах JPEG и PNG, например, папку *Source\Assets\Images* в папке лабораторной.
  9. Скопируйте несколько файлов из библиотеки изображений в папку эмулированного «облачного» диска.
  10. Переключитесь в окно браузера, показывающее содержимое хранилища изображений, и обновите страницу. Обратите внимание на появившиеся файлы.
      1. 
      2. Изображение
      3. Обновленное содержимое диска Windows Azure
  11. Закройте окно браузера.

Задание 3 – Создание нового диска в «облаке»

* 1. В этом задании вы создадите в приложении новый «облачный» диск, примонтируете его и скопируете в него содержимое диска-источника.
  2. Щелкните правой кнопкой мыши на файле **Default.aspx** и нажмите **View Markup**.
  3. Вставьте код разметки между тегами **h1** и **h2**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-06-MountedDrives Panel*)
     2. HTML

...

<body>

<form id="form1" runat="server">

<div>

<h1>PhotoAlbum</h1>

**<asp:Panel ID="SelectDrive" runat="server" Visible="false">**

**<asp:LinkButton ID="NewDrive" runat="server" Text="New Drive" onclick="NewDrive\_Click" CssClass="newdrive" />**

**Mounted Drives:**

**<asp:DropDownList ID="MountedDrives" runat="server" AutoPostBack="true"**

**DataTextField="Name" DataValueField="Value"**

**OnSelectedIndexChanged="MountedDrives\_SelectedIndexChanged" />**

**</asp:Panel>**

<h2>Image Store Drive: (<%=this.CurrentPath%>)</h2>

<asp:GridView DataSourceID="LinqDataSource1" AutoGenerateColumns="False"

...

* + 1. **Примечание:** Разметка теперь содержит выпадающий список примонтированных к web—роли дисков и кнопку создания нового диска Windows Azure.
  1. Щелкните правой кнопкой мыши на файле **Default.aspx** и нажмите **View Code**.
  2. Добавьте пространства имён в начало файла.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-07-AzureNamespaces-CS*)
     2. C#

**using Microsoft.WindowsAzure;**

**using Microsoft.WindowsAzure.Diagnostics;**

**using Microsoft.WindowsAzure.ServiceRuntime;**

**using Microsoft.WindowsAzure.StorageClient;**

* + 1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-07-AzureNamespaces-VB*)
    2. Visual Basic

**Imports Microsoft.WindowsAzure**

**Imports Microsoft.WindowsAzure.Diagnostics**

**Imports Microsoft.WindowsAzure.ServiceRuntime**

**Imports Microsoft.WindowsAzure.StorageClient**

* 1. Вставьте в конец метода **Page\_PreRender** выделенный код.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-08-Page\_PreRender-CS*)
     2. C#

protected void Page\_PreRender(object sender, EventArgs e)

{

GridView1.Columns[GridView1.Columns.Count - 1].Visible = (this.CurrentPath != Global.ImageStorePath);

**if (RoleEnvironment.IsAvailable)**

**{**

**MountedDrives.DataSource = from item in CloudDrive.GetMountedDrives()**

**select new**

**{**

**Name = item.Key + " => " + item.Value,**

**Value = item.Key**

**};**

**MountedDrives.DataBind();**

**MountedDrives.SelectedValue = this.CurrentPath;**

**SelectDrive.Visible = true;**

**NewDrive.Text = MountedDrives.Items.Count < 2 ? "New Drive" : "Delete Drive";**

**}**

}

* + 1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-08-Page\_PreRender-VB*)
    2. Visual Basic

Protected Sub Page\_PreRender(ByVal sender As Object, ByVal e As EventArgs) Handles Me.PreRender

GridView1.Columns(GridView1.Columns.Count - 1).Visible = (Me.CurrentPath <> Global\_asax.ImageStorePath)

**If RoleEnvironment.IsAvailable Then**

**MountedDrives.DataSource = From item In CloudDrive.GetMountedDrives() \_**

**Select New With {.Name = item.Key & " => " & item.Value.ToString(), .Value = item.Key}**

**MountedDrives.DataBind()**

**MountedDrives.SelectedValue = Me.CurrentPath**

**SelectDrive.Visible = True**

**NewDrive.Text = IIf(MountedDrives.Items.Count < 2, "New Drive", "Delete Drive")**

**End If**

End Sub

* + 1. **Примечание:** В коде выпадающий список заполняется примонтированными к web-роли дисками. Список показывает «привязку» URI страничного блоба и соответствующую букву диска диска.
  1. Добавьте код обработчика событий для кнопки **New Drive:** вставьте код в класс \_**Default**.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-09-NewDrive\_Click-CS*)
     2. C#

**protected void NewDrive\_Click(object sender, EventArgs e)**

**{**

**if (RoleEnvironment.IsAvailable)**

**{**

**// Получение аккаунта хранилища**

**CloudStorageAccount account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**

**// Настройка URI страничного блоба для нового диска с помощью изменения расширения URI источника**

**string imageStoreBlobUri = RoleEnvironment.GetConfigurationSettingValue("ImageStoreBlobUri");**

**string cloneStoreBlobUri = Path.ChangeExtension(imageStoreBlobUri, "bak");**

**// Создание диска и соответствующего страничного блоба**

**CloudDrive clonedDrive = account.CreateCloudDrive(cloneStoreBlobUri);**

**if (MountedDrives.Items.Count < 2)**

**{**

**try**

**{**

**clonedDrive.Create(16);**

**}**

**catch (CloudDriveException)**

**{**

**// Диск уже существует**

**}**

**// Монтирование диска и получение пути**

**LocalResource cache = RoleEnvironment.GetLocalResource("LocalDriveCache");**

**string clonedStorePath = clonedDrive.Mount(cache.MaximumSizeInMegabytes / 2, DriveMountOptions.None);**

**// Копирование содержимого диска-источника на новый диск**

**foreach (string sourceFileName in Directory.GetFiles(Global.ImageStorePath, "\*.\*").Where(name => name.EndsWith(".jpg") || name.EndsWith(".png")))**

**{**

**string destinationFileName = Path.Combine(clonedStorePath, Path.GetFileName(sourceFileName));**

**File.Copy(sourceFileName, destinationFileName, true);**

**}**

**SelectImageStore(clonedStorePath);**

**}**

**else**

**{**

**clonedDrive.Unmount();**

**clonedDrive.Delete();**

**SelectImageStore(Global.ImageStorePath);**

**}**

**}**

**}**

* + 1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-09-NewDrive\_Click-VB*)
    2. Visual Basic

**Protected Sub NewDrive\_Click(ByVal sender As Object, ByVal e As EventArgs)**

**If RoleEnvironment.IsAvailable Then**

**' Получение аккаунта хранилища**

**Dim account As CloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**

**' Настройка URI страничного блоба для нового диска с помощью изменения расширения URI источника**

**Dim imageStoreBlobUri As String = RoleEnvironment.GetConfigurationSettingValue("ImageStoreBlobUri")**

**Dim cloneStoreBlobUri As String = Path.ChangeExtension(imageStoreBlobUri, "bak")**

**' Создание диска и соответствующего страничного блоба**

**Dim clonedDrive As CloudDrive = account.CreateCloudDrive(cloneStoreBlobUri)**

**If (MountedDrives.Items.Count < 2) Then**

**Try**

**clonedDrive.Create(16)**

**Catch e1 As CloudDriveException**

**' Диск уже существует**

**End Try**

**' Монтирование диска и получение пути**

**Dim cache As LocalResource = RoleEnvironment.GetLocalResource("LocalDriveCache")**

**Dim clonedStorePath As String = clonedDrive.Mount(cache.MaximumSizeInMegabytes / 2, DriveMountOptions.None)**

**' Копирование содержимого диска-источника на новый диск**

**For Each sourceFileName As String In Directory.GetFiles(Global\_asax.ImageStorePath, "\*.\*").Where(Function(name) name.EndsWith(".jpg") Or name.EndsWith(".png"))**

**Dim destinationFileName As String = Path.Combine(clonedStorePath, Path.GetFileName(sourceFileName))**

**File.Copy(sourceFileName, destinationFileName, True)**

**Next**

**SelectImageStore(clonedStorePath)**

**Else**

**clonedDrive.Unmount()**

**clonedDrive.Delete()**

**SelectImageStore(Global\_asax.ImageStorePath)**

**End If**

**End If**

**End Sub**

* + 1. **Примечание:** В коде происходит проверка, не примонтирован ли уже второй диск, и если нет, то создается новый NTFS диск и соответствующий страничный блоб, после чего содержимое диска-источника копируется на второй диск. Если диск уже существует, сперва он размонтируется и после этого будет удалён.
  1. Добавьте обработчик события **SelectedIndexChanged** для выпадающего списка: вставьте в класс **\_Default** код.
     1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-10-MountedDrives\_SelectedIndexChanged-CS*)
     2. C#

**protected void MountedDrives\_SelectedIndexChanged(object sender, EventArgs e)**

**{**

**SelectImageStore(MountedDrives.SelectedValue);**

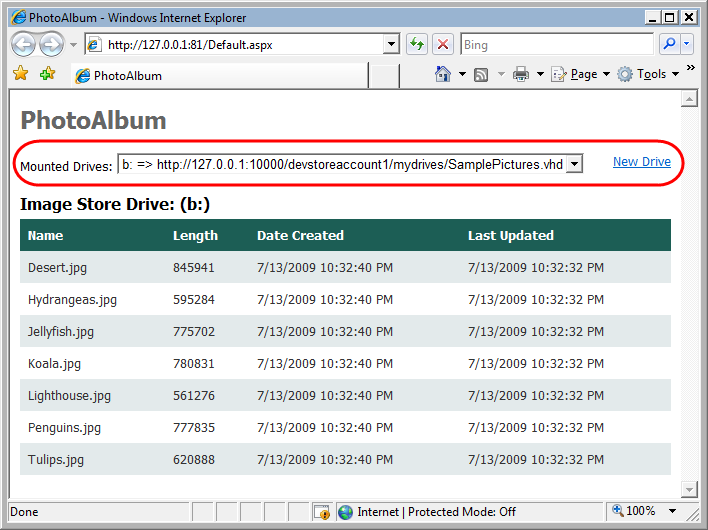
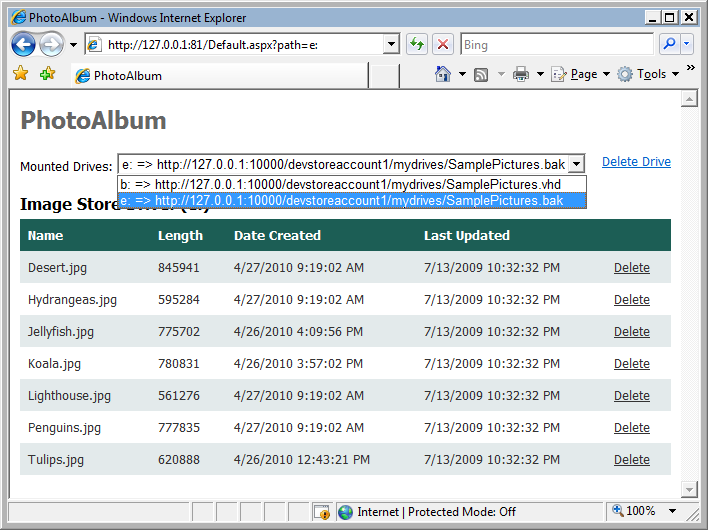
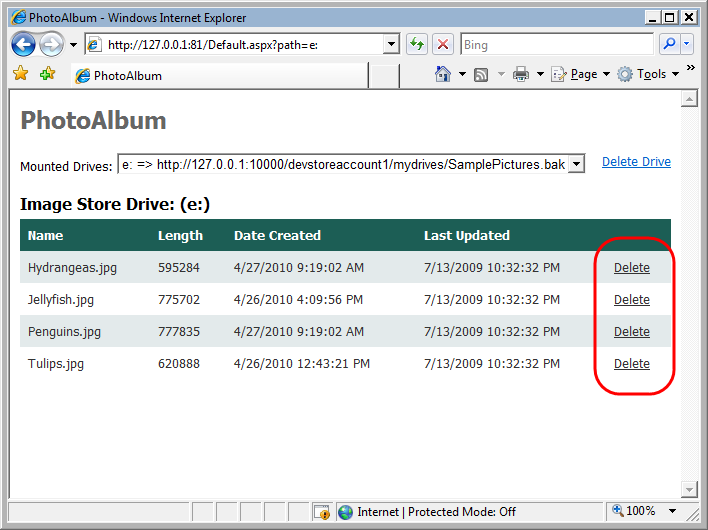
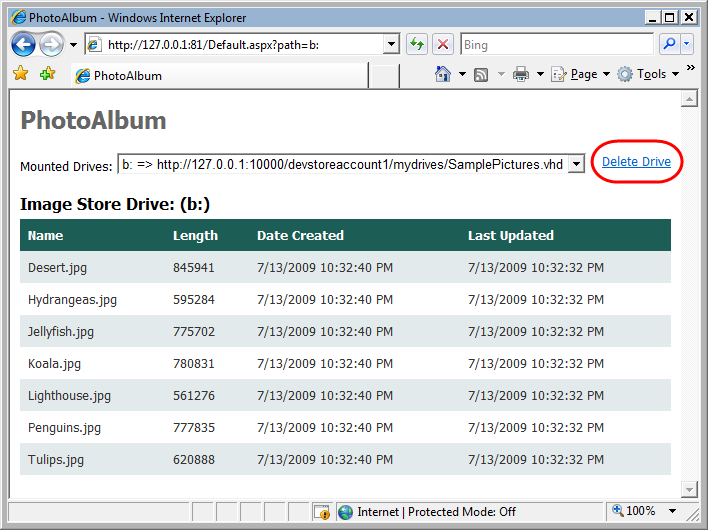
**}**

* + 1. (Сниппет – *ExploringWindowsAzureStorage-Ex04-10-MountedDrives\_SelectedIndexChanged-VB*)
    2. Visual Basic

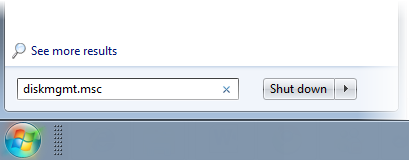
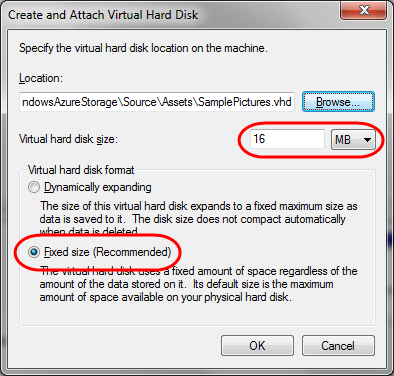
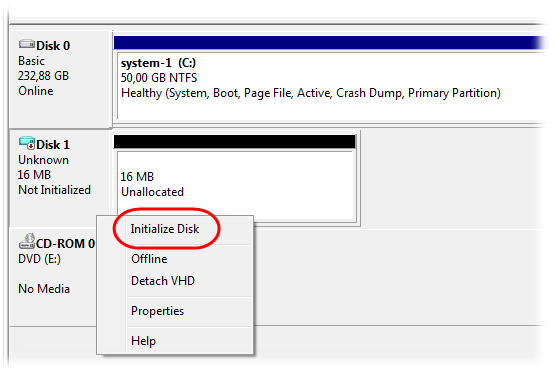
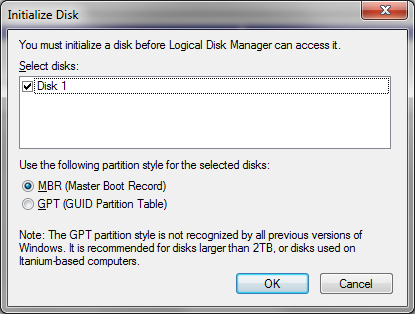
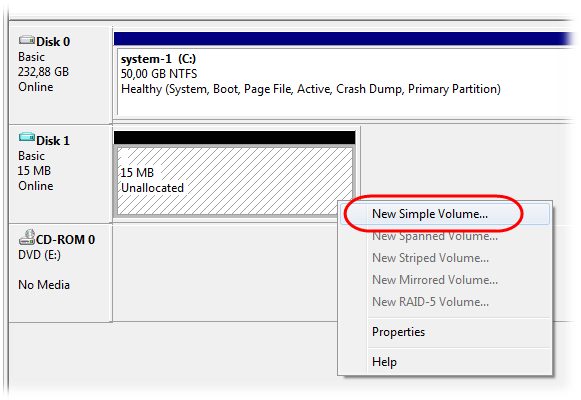
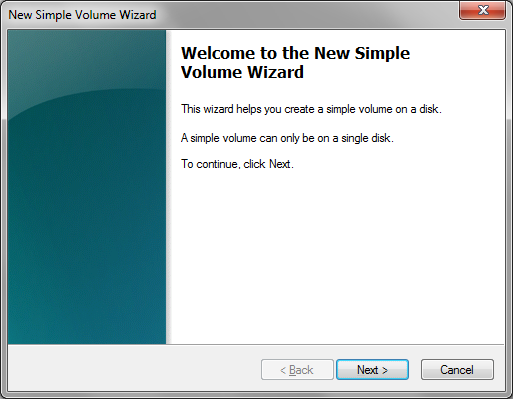
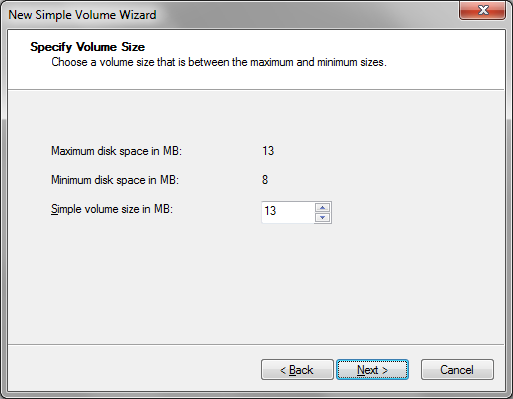
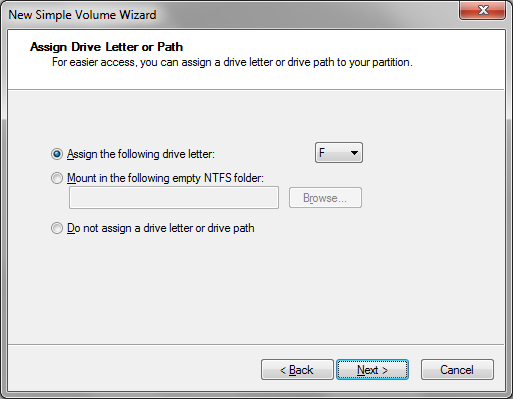
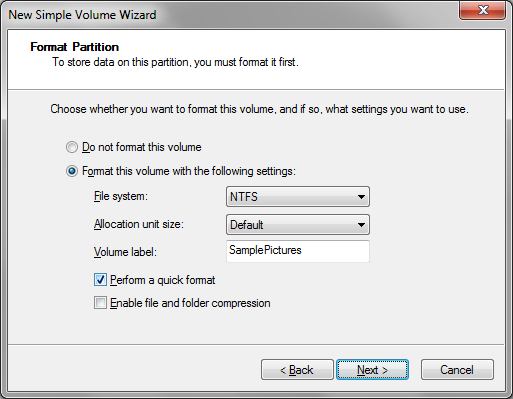
**Protected Sub MountedDrives\_SelectedIndexChanged(ByVal sender As Object, ByVal e As EventArgs)**

**SelectImageStore(MountedDrives.SelectedValue)**

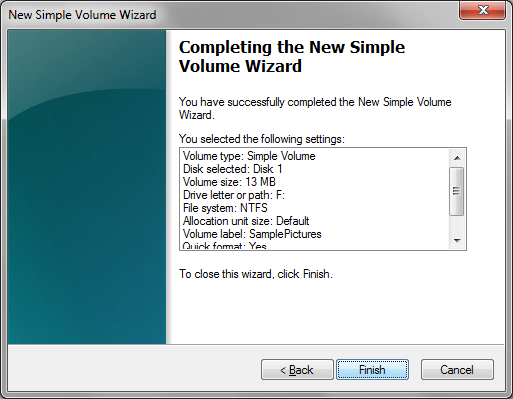
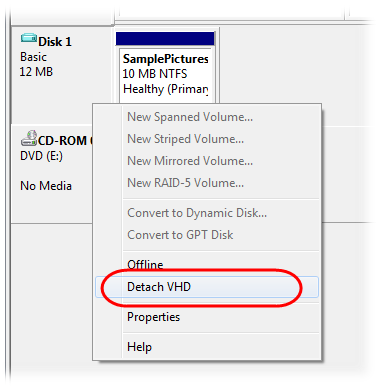
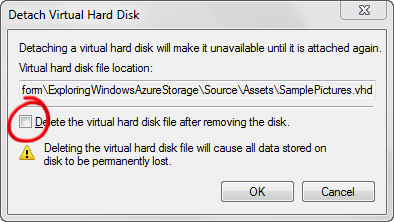
**End Sub**

* + 1. **Примечание:** В коде происходит инициализация обновления содержимого страницы и показа содержимого выделенного диска.
  1. Нажмите **F5** для запуска приложения в эмуляторе вычислений. После запуска приложения обратите внимание, что выпадающий список **Mounted Drives** содержит буквы, назначенные дискам, и соответствующие URL страничных блобов, служащих в качестве хранилища.
     1. 
     2. Изображение
     3. Примонтированные в данный момент диски в приложении
  2. Нажмите на ссылку **New Drive** для создания нового диска и копирования содержимого диска-источника на новый диск. После создания диска страница обновится и покажет содержимое нового диска. Обратите внимание, что теперь в выпадающем списке примонтированных дисков показывается новый диск.
     1. 
     2. Изображение
     3. Созданный диск в приложении
  3. Удалите несколько файлов, нажав на ссылку **Delete**.
     1. 
     2. Изображение
     3. Удаление файлов на втором диске
  4. Выберите диск-источник в выпадающем списке **Mounted Drives** и проверьте, что его содержимое не изменилось.
  5. Нажмите **Delete Drive** для размонтирования диска и удаления соответствующего страничного блоба.
     1. 
     2. Изображение
     3. Размонтирование и удаления диска
     4. **Примечание:** Удалится только второй диск, диск-источник удалён не будет.
  6. Закройте окно браузера. Теперь вы развернете и протестируете приложение в Windows Azure.

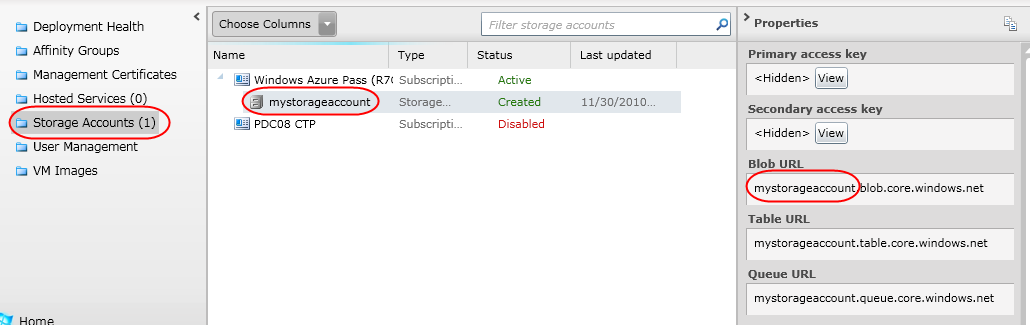
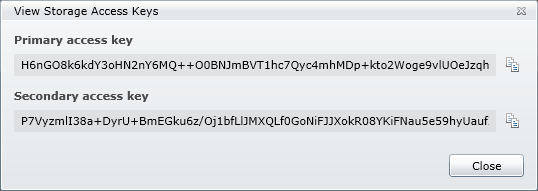
Задание 4 - Создание NTFS VHD локально

* 1. В предыдущих заданиях вы использовали диски Windows Azure в среде эмулятора вычислений. Когда вы разворачиваете приложение в Windows Azure, вам необходимо механизм для загрузки информации в блоб, используемый приложением. В этом случае можно создать Virtual Hard Drive (VHD) локально на вашей машине, скопировать на него необходимую информацию и загрузить в страничный блоб Windows Azure.
  2. В этом задании вы создадите файл NTFS VHD, содержащий изображения, который вы сможете загрузить в хранилище Windows Azure и использовать как хранилище для диска Windows Azure.
  3. .
  4. **Примечание:** Это задание является необязательным и опирается на доступные в Windows 7 и Windows Server 2008 R2 функции. Если у вас не установлены ОС этих версий, вы можете пропустить это задание и воспользоваться уже созданным VHD из папки *Assets* лабораторной.
  5. **Важно:** Для выполнения оставшихся заданий в этом упражнении вам необходима действующая подписка Windows Azure.
  6. Подробнее [Windows Azure Portal](http://www.microsoft.com/windowsazure/).
  7. Откройте консоль **Disk Management**: в меню **Start** наберите в строке поиска **diskmgmt.msc** и нажмите **Enter**.
     1. 
     2. Изображение
     3. Запуск консоли Disk Management
  8. В консоли Disk Management откройте меню **Action** и выберите **Create VHD**.
  9. В диалоговом окне **Create and Attach Virtual Hard Disk** нажмите **Browse**,перейдите в **\Source\Ex04-WorkingWithDrives**, введите имя как **SamplePictures.vhd** и нажмите **Save**. Укажите размер диска **Virtual hard disk size** как *16 MB*, **Virtual hard disk format** как **Fixed size**, после чего нажмите **OK** для создания и подсоединения виртуального жесткого диска.
     1. 
     2. Изображение
     3. Создание виртуального жесткого диска (VHD)
  10. Перед использованием нового диска вы должны инициализировать его: щелкните правой кнопкой мыши на иконке диска для созданного диска в нижней панели **Disk Management** и нажмите **Initialize Disk**.
      1. 
      2. Изображение
      3. Инициализация виртуального жесткого диска (VHD)
  11. В диалоговом окне **Initialize Disk** убедитесь, что выбран диск, соответствующий подсоединенному VHD, укажите **MBR (Master Boot Record)** и нажмите **OK**.
      1. 
      2. Изображение
      3. Настройка инициализации диска
  12. Щелкните правой кнопкой мыши на неразмеченной Unallocated области подсоединенного виртуального жесткого диска и нажмите **New Simple Volume**.
      1. 
      2. Изображение
      3. Создание тома в виртуальном жестком диске (VHD)
  13. В **New Simple Volume Wizard** нажмите **Next**.
      1. 
      2. Изображение
      3. Страница приветствия Simple Volume Wizard
  14. Оставьте значение **Simple volume size** таким же—оно должно совпадать с **Maximum disk space**—и нажмите **Next**.
      1. 
      2. Изображение
      3. Определение размера тома диска
  15. Назначьте букву диска и нажмите **Next**.
      1. 
      2. Изображение
      3. Назначение буквы диска тома
  16. Выберите тип форматирования новой партиции. Укажите **File system** как *NTFS*, оставьте стандартное значение **Allocation unit size** и определите **Volume label** как *SamplePictures*. Убедитесь, что вы включили опцию **Perform a quick format** и оставили выключенной **Enable file and folder compression**. Нажмите **Next**.
      1. 
      2. Изображение
      3. Форматирование партиции нового диска

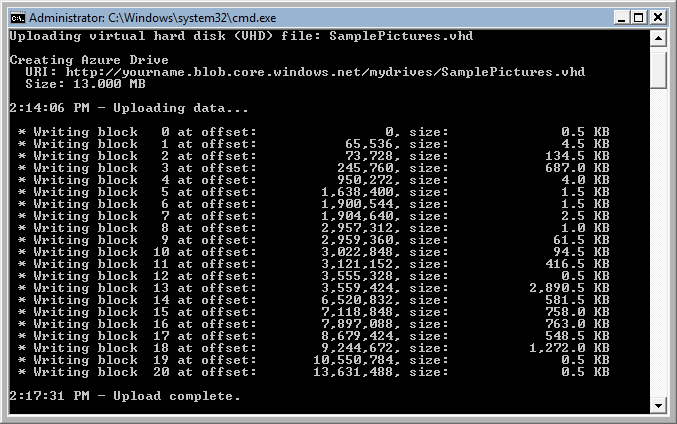
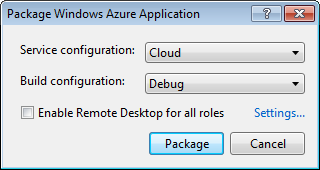
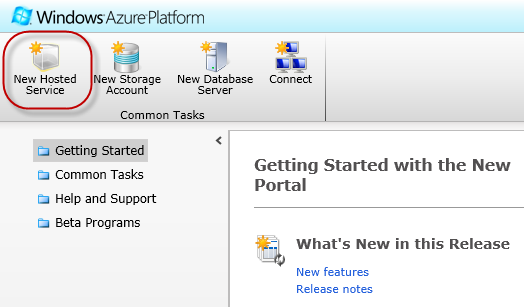
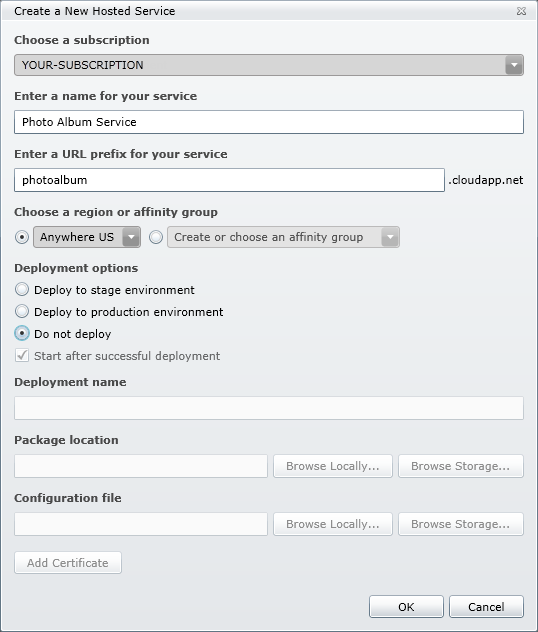
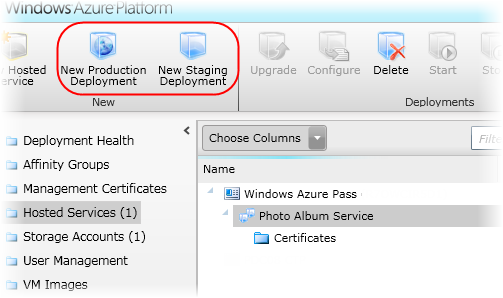
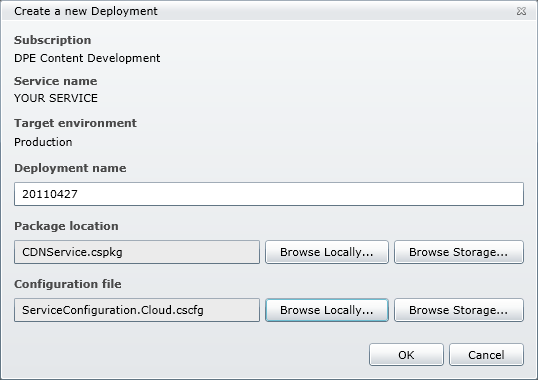
**Примечание:** На данный момент диски Windows Azure требуют виртуального жесткого диска (VHD), форматированного в NTFS.

* 1. Проверьте информацию на странице Summary и нажмите **Finish** для создания нового тома.
     1. 
     2. Изображение
     3. Завершение New Simple Volume Wizard
  2. Дождитесь окончания форматирования, которое должно занять несколько секунд. При включенном **AutoPlay** вам будет задан вопрос, необходимо ли просмотреть подсоединенный диск. В этом случае нажмите **Open folder to view files**. Если вопрос задан не будет, щелкните правой кнопкой мыши на томе в консоли Disk Management и нажмите **Open**. Оставьте окно открытым.
  3. В меню **Start** выберите **Pictures** и перейдите в папку **Sample Pictures**.
     1. **Примечание:** Если в вашей среде нет этой папки, вы можете использовать любую другую, содержащую коллекцию изображений в форматах JPEG и PNG, например, папку *Source\Assets\Images* в папке лабораторной.
  4. Скопируйте все файлы из папки **Sample Pictures** в первое окно, скопировав таким образом все изображения на диск VHD. Если диск заполнится, не беспокойтесь - вам необходимо лишь несколько файлов (для тестирования работы приложения).
  5. Переключитесь в консоль Disk Management, щелкните правой кнопкой мыши на подсоединенном диске— щелкните на диске, а не на области партиции – и нажмите **Detach VHD**.
     1. 
     2. Изображение
     3. Подготовка к отсоединению виртуального жесткого диска (VHD)
  6. В диалоговом окне **Detach Virtual Hard Disk** убедитесь, что опция **Delete the virtual hard disk file after removing the disk** отключена, после чего нажмите **OK**.
     1. 
     2. Изображение
     3. Отсоединение виртуального жесткого диска
  7. Теперь вам необходимо загрузить виртуальный жесткий диск (VHD) в хранилище Windows Azure.

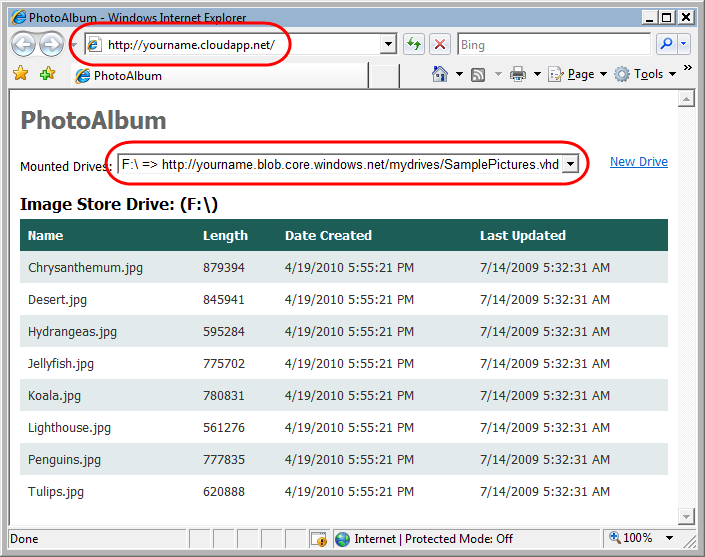
Задание 5 – Развертывание приложения и загрузка диска в Windows Azure

* 1. В этом задании вы загрузите созданный виртуальный жесткий диск в страничный блоб Windows Azure. Материалы к лабораторной включают утилиту, необходимую для этого.
  2. Перед загрузкой VHD вам необходимо определить имя и ключ доступа к аккаунту – для этого зайдите на портал управления [Windows Azure Management Portal](http://windows.azure.com/) и выберите подписку, в которой будет развернуто ваше приложение. Выберите сервис хранилища из списка сервисов и запишите значения имени **name** (первый сегмент URL точки входа) и ключ доступа **Primary Access Key,** нажав на кнопку **View** (для копирования ключа в буфер используйте кнопку **Copy to Clipboard**).
     1. 
     2. Изображение
     3. Просмотр информации аккаунта хранилища Windows Azure
     4. 
     5. Изображение
     6. Получение ключей доступа к аккаунту хранилища
  3. Запустите командную строку и перейдите в папку **ExploringWindowsAzureStorageVS2010\Source\Assets\VHDUpload**.
  4. В командной строке наберите команду, заменив «болванки» **<vhdFilePath>** на путь к созданному ранее VHD, **<accountName>** и **<accountKey>** на имя и ключ доступа к аккаунту хранилища соответственно.
     1. Командная строка

**VHDUPLOAD <vhdFilePath> mydrives/SamplePictures.vhd <accountName> <accountKey>**

* + 1. **Примечание:** URI сервиса блоба, который определяет расположение файла .vhd, чувствителен к регистру – будьте внимательны при вводе.
    2. **Примечание:** Если ваша платформа не поддерживает создание и монтирование файлов VHD и вы не смогли выполнить предыдущее задание, используйте файл **SamplePictures.vhd**, расположенный в одной папке с утилитой.
  1. Нажмите **Enter** для загрузки виртуального жесткого диска (VHD) в хранилище блобов. Дождитесь окончания операции – она может занять несколько минут.
     1. 
     2. Изображение
     3. Загрузка VHD в страничный блоб Windows Azure
  2. Загрузив VHD в хранилище Azure, вы теперь готовы к развертыванию приложения.
  3. Переключитесь в Visual Studio.
  4. Добавьте информацию аккаунта хранилища в конфигурацию Web-роли: в проекте **PhotoAlbumService** разверните узел **Roles** и щелкните два раза на роли **PhotoAlbum**.
  5. В окне **PhotoAlbum [Role]** переключитесь на вкладку **Settings** , найдите настройку *Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString* setting и нажмите на кнопку с троеточием справа. В диалоговом окне **Storage Connection String** выберите **Enter Storage Credentials** и введите имя аккаунта **Account name** и ключ доступа **Account key**. Убедитесь, что в **Connection** выбрана опция **Use default HTTPS endpoints,** после чего нажмите**OK**.
  6. Повторите последовательность действий для настройки *DataConnectionString*. Введите те же данные, в **Connection** же включите **Use default HTTP endpoints**.
     1. 
     2. Изображение
     3. Настройка аккаунта хранилища для «облачного» диска на использование точек входа HTTP
     4. **Примечание:** На данный момент диски Windows Azure поддерживают только точки входа HTTP.
  7. Для создания пакета сервиса щелкните правой кнопкой мыши на проекте **PhotoAlbumService** и нажмите **Package**.
     1. **Примечание:** Подробнее про развертывание приложений обратитесь к лабораторной **Развертывание Windows Azure –** в упражнении 3 этой лабораторной показан процесс развертывания приложений с помощью Visual Studio.
     2. 
     3. Изображение
     4. Создание пакета сервиса в Visual Studio
  8. В диалоговом окне **Package Windows Azure Application** нажмите **Package** для создания пакета. После создания откроется папка с созданным пакетом. Несмотря на возможность использования Windows Azure Tools для развертывания сервиса прямо из Visual Studio, в этой лабораторной вы развернете сервис с помощью портала управления Windows Azure.
  9. Для развертывания сервиса зайди на портал управления [Management Portal](http://windows.azure.com/).
  10. На портале выберите проект, в который будете разворачивать приложение. Если вы не создавали ранее сервис, вы должны создать его на этом шаге. В случае наличия сервиса перейдите к шагу 20.
  11. На левой панели нажмите **Hosted Services** и нажмите кнопку **New Hosted Service**, расположенную в меню.
      1. 
      2. Изображение
      3. Создание нового сервиса
  12. В диалоговом окне **Create a new Hosted Service** выберите вашу подписку из выпадающего списка **Choose a subscription**.
  13. Введите имя сервиса в текстовое поле **Enter a name for your service** и укажите URL, введя соответствующее значение **Enter a URL prefix for your service.**
  14. Выберите один из доступных регионов размещения в **Choose a region**.
  15. Укажите **Do not Deploy**.
      1. **Примечание:** Несмотря на то, что вы можете создать и развернуть ваше приложение в Windows Azure одной операцией, заполнив секцию **Deployment Options**, в этой лабораторной вы выполните развертывание в следующем задании.
      2. 
      3. Изображение
      4. Настройка сервиса
  16. Нажмите **OK** для создания сервиса и подождите, пока не завершится процесс инициализации.
  17. Выберите сервис, в котором будет развернуто приложение.
  18. Нажмите **New Production Deployment** или **New Staging Deployment** в зависимости от ячейки развертывания, в которой должно работать приложение – в реальном развертывании или тестовом.
      1. 
      2. Изображение
      3. Развертывание приложения в сервис
  19. В диалоговом окне **Create a new Deployment**, чтобы определить **Package location**, нажмите **Browse Locally**. Введите метку для имени развертывания **Deployment name**, например **v1.0,** и нажмите **OK**.
      1. 
      2. Изображение

Создание нового развертывания

* + 1. **Примечание:** Виртуальные машины Windows Azure выполняют ваше приложение в гостевой операционной системе. Для поддержки функциональности дисков Windows Azure версия операционной системы должна быть совместима с Windows Azure SDK версии 1.1 и выше. Подробная информация о доступных версиях гостевых ОС [Windows Azure Guest OS Releases and SDK Compatibility Matrix](http://msdn.microsoft.com/en-us/library/ee924680(v=MSDN.10).aspx).
    2. Рекомендуется использовать наиболее современные доступные ОС для получения преимуществ новой функциональности и «заплаток» безопасности. Если вы не определили версию ОС в вашем файле конфигурации, обновление вашей ОС до последних версий будет производиться автоматически. В этой лабораторной вы используете режим автоматического обновления. Таким образом, вы можете быть уверены, что приложение работает в гостевой ОС, поддерживающей функциональность дисков Windows Azure.
  1. Дождитесь окончания процесса загрузки. По готовности статус сервиса должен показывать **Ready**.
  2. На вкладке **Properties** нажмите на **DNS name** для открытия приложения в браузере. Приложение должно работать также как работало при развертывании локально. Однако обратите внимание, что URL страничного блоба для примонтированного диска теперь ведёт на страничный блоб, в который вы загрузили файл VHD.
     1. 
     2. Изображение
     3. Запуск приложения в Windows Azure
  3. Нажмите на ссылку **New Drive** и проверьте, создался ли новый диск и скопировались ли в него файлы. Проверьте функциональность удаления, удалив несколько файлов со второго диска.
  4. Нажмите **Delete Drive** для размонтирования диска и удаления соответствующего страничного блоба.

Заключение

* 1. В этой лабораторной работе вы научились работать с хранилищем Windows Azure, используя таблицы, блобы, очереди и диски.
  2. Таблицы хранят данные в виде коллекций сущностей. Сущности похожи на записи. Каждая сущность имеет первичный ключ и набор свойств. Свойства являются парами ключ-значение и похожи на столбцы.
  3. Блобы могут содержать любые бинарные данные. В этой лабораторной вы использовали сервис блобов для хранения и показа изображений и соответствующих метаданных в веб-приложении.
  4. При использовании очередей вы увидели, как можно управлять простыми сообщениями (содержащими строки, XML или бинарные данные). Это отличный способ осуществления обмена сообщениями между Web и Worker-ролями.
  5. Вы изучили работу дисков Windows Azure, которые позволяют читать и записывать данные в хранилище блобов, используя стандартные функции файловой системы.