

**Hands-On Lab**

**Introduction to Windows Azure**

Lab version:2.0.0

Last updated:12/17/2010

Contents

[Overview 3](#_Toc294687171)

[Objectives 5](#_Toc294687172)

[Prerequisites 5](#_Toc294687173)

[Setup 5](#_Toc294687174)

[Using the Code Snippets 6](#_Toc294687175)

[Exercises 6](#_Toc294687176)

[Exercise 1: Building Your First Windows Azure Application 7](#_Toc294687177)

[Task 1 – Creating the Visual Studio Project 7](#_Toc294687178)

[Task 2 – Creating a Data Model for Entities in Table Storage 11](#_Toc294687179)

# Overview

A service hosted in Windows Azure consists of one or more web roles and worker roles.

* A *web role* is an ASP.NET Web application accessible via an HTTP or HTTPS endpoint and is commonly the front-end for an application.
* *Worker roles* are background-processing applications and are typically found in the back-end.

Windows Azure services may be comprised of one or both types of roles and can run multiple instances of each type. Role instances can be added or removed based on demand and allow applications to quickly and economically scale-up or down when the need arises.

*Windows Azure storage services* provide storage in the cloud, which includes:

* *Blob services* for storing text and binary data
* *Table services* for structured storage that can be queried
* *Queue services* for reliable and persistent messaging between services

In this hands-on lab, you will explore the basic elements of a Windows Azure service by creating a simple GuestBook application that demonstrates many features of the Windows Azure platform, including web and worker roles, blob storage, table storage, and queues.

In the GuestBook application, a web role provides the front-end that allows users to view the contents of the guest book and submit new entries. Each entry contains a name, a message, and an associated picture. The application also contains a worker role that can generate thumbnails for the images that users submit.

When users post a new item, the web role uploads the picture to blob storage and creates an entry in table storage that contains the information entered by the user and a link to the blob with the picture. The web role renders this information to the browser so users can view the contents of the guest book.

After storing the image and creating the entry, the web role posts a work item to a queue to have the image processed. The worker role fetches the work item from the queue, retrieves the image from blob storage, and resizes it to create a thumbnail.

Using queues to post work items is a common pattern in cloud applications and enables the separation of compute-bound tasks from the front-end. The advantage of this approach is that front and back ends can be scaled independently.

## Objectives

In this hands-on lab, you will learn how to:

* Create applications in Windows Azure using web roles and worker roles
* Use Windows Azure storage services including blobs, queues and tables
* Deploy an application to Windows Azure

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| For a more advanced scenario based on the GuestBook application that shows how to deal with poison messages in Azure queues and how to design an Azure role to make use of plug-in components that can be dynamically loaded, see the **Asynchronous Workload Handling** lab in this training kit. |

## Prerequisites

The following is required to complete this hands-on lab:

* IIS 7 (with ASP.NET, WCF HTTP Activation)
* [Microsoft Visual Studio 2010](http://msdn.microsoft.com/vstudio/products/)
* Microsoft .NET Framework 4.0
* [Windows Azure Tools for Microsoft Visual Studio 1.3](http://go.microsoft.com/fwlink/?LinkID=128752)
* [SQL Server 2005 Express Edition (or later)](http://www.microsoft.com/express/sql/download/)

## Setup

For convenience, much of the code used in this hands-on lab is available as Visual Studio code snippets. To check the prerequisites of the lab and install the code snippets:

1. Open a Windows Explorer window and browse to the lab’s **Source\Setup** folder.
2. Double-click the **Dependencies.dep** file in this folder to launch the Dependency Checker tool and install any missing prerequisites and the Visual Studio code snippets.
3. If the User Account Control dialog is shown, confirm the action to proceed.

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| This process may require elevation. The *.dep* extension is associated with the Dependency Checker tool during its installation. For additional information about the setup procedure and how to install the Dependency Checker tool, refer to the **Setup.docx** document in the **Assets** folder of the training kit. |

## Using the Code Snippets

Throughout the lab document, you will be instructed to insert code blocks. For your convenience, most of that code is provided as Visual Studio Code Snippets, which you can use from within Visual Studio 2010 to avoid having to add it manually.

If you are not familiar with the Visual Studio Code Snippets, and want to learn how to use them, you can refer to the **Setup.docx** document in the **Assets** folder of the training kit, which contains a section describing how to use them.

## Exercises

This hands-on lab includes the following exercises:

* Building Your First Windows Azure Application
* Background Processing with Worker Roles and Queues
* Deploying a Windows Azure Application

Estimated time to complete this lab: **60 minutes**.

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| When you first start Visual Studio, you must select one of the predefined settings collections. Every predefined collection is designed to match a particular development style and determines window layouts, editor behavior, IntelliSense code snippets, and dialog box options. The procedures in this lab describe the actions necessary to accomplish a given task in Visual Studio when using the **General Development Settings** collection. If you choose a different settings collection for your development environment, there may be differences in these procedures that you need to take into account. |

# Exercise 1: Building Your First Windows Azure Application

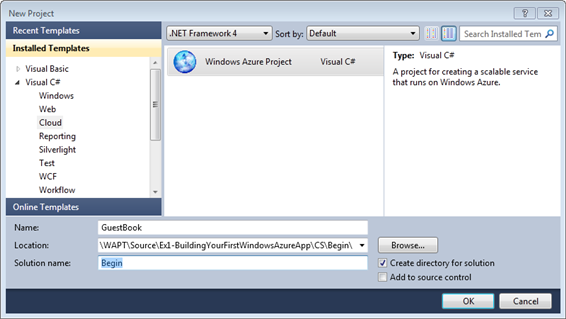
In this exercise, you create a guest book application and execute it in the local development fabric. For this purpose, you will use the Windows Azure Tools for Microsoft Visual Studio to create the project using the Cloud Service project template. These tools extend Visual Studio to enable the creation, building and running of Windows Azure services. You will continue to work with this project throughout the remainder of the lab.

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| To reduce typing, you can right-click where you want to insert source code, select Insert Snippet, select My Code Snippets and then select the entry matching the current exercise step. |

## Task 1 – Creating the Visual Studio Project

In this task, you create a new Cloud Service project in Visual Studio.

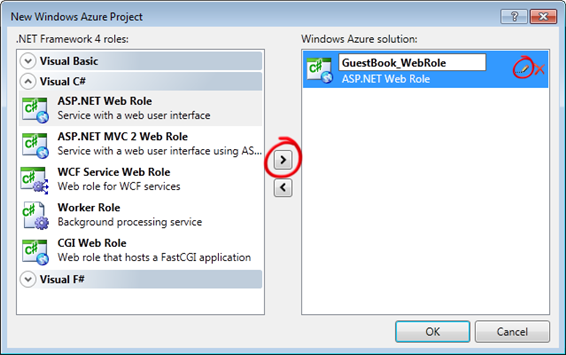
1. Open Visual Studioas administrator from **Start | All Programs | Microsoft Visual Studio 2010** by right clicking the **Microsoft Visual Studio 2010** shortcut and choosing **Run as administrator**.
2. If the **User Account Control** dialog appears, click **Continue**.
3. From the **File** menu, choose **New** and then **Project**.
4. In the **New Project** dialog, expand the language of your preference (Visual C# or Visual Basic) in the **Installed Templates** list and select **Cloud**. Choose the **Windows Azure Project** template, set the **Name** of the project to **GuestBook**, set the location to **Ex1-BuildingYourFirstWindowsAzureApp\CS** in the **Source** folder of the lab, change the solution name to **Begin**, and ensure that **Create directory for solution** is checked. Click **OK** to create the project.



**Figure 1**  
*Creating a new Windows Azure Cloud Service project (C#)*

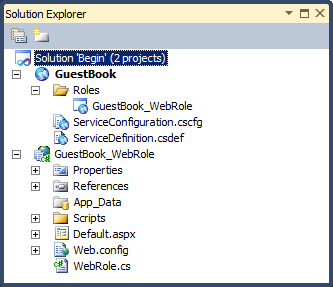
|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| Windows Azure supports the .NET Framework 4.0. If you use Visual Studio 2010 to create the project, you can select this version for the target framework and take advantage of its new features. |

1. In the **New Windows Azure Project** dialog, inside the **Roles** panel, expand the tab for the language of your choice (Visual C# or Visual Basic), select **ASP.NET Web Role** from the list of available roles and click the arrow (>) to add an instance of this role to the solution. Before closing the dialog, select the new role in the right panel, click the pencil icon and rename the role as **GuestBook\_WebRole**. Click **OK** to create the cloud service solution.



**Figure 3**  
*Assigning roles to the cloud service project (C#)*

1. In **Solution Explorer**, review the structure of the created solution.



**Figure 5**  
*Solution Explorer showing the GuestBook application (C#)*

**C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:**

The generated solution contains two separate projects.

* The first project, named **GuestBook**, holds the configuration for the web and worker roles that compose the cloud application.
  + It includes the service definition file, **ServiceDefinition.csdef**, which contains metadata needed by the Windows Azure fabric to understand the requirements of your application, such as which roles are used, their trust level, the endpoints exposed by each role, the local storage requirements and the certificates used by the roles. The service definition also establishes configuration settings specific to the application.
  + The service configuration file, **ServiceConfiguration.cscfg**, specifies the number of instances to run for each role and sets the value of configuration settings defined in the service definition file.
  + This separation between service definition and configuration allows you to update the settings of a running application by uploading a new service configuration file.
  + The **Roles** node in the cloud service project enables you to configure what roles the service includes (web, worker or both) as well as which projects to associate with these roles. Adding and configuring roles through the Roles node will update the **ServiceDefinition.csdef** and **ServiceConfiguration.cscfg** files.
* The second project, named **GuestBook\_WebRole**, is a standard ASP.NET Web Application project template modified for the Windows Azure environment. It contains an additional class that provides the entry point for the web role and contains methods to manage the initialization, starting, and stopping of the role.

## Task 2 – Creating a Data Model for Entities in Table Storage

The application stores guest book entries in Windows Azure Table storage.

The Table service offers semi-structured storage in the form of tables that contain collections of entities. Entities have a primary key and a set of properties, where a property is a name, typed-value pair.

In addition to the properties required by your model, every entity in Table Storage has two key properties: the **PartitionKey** and the **RowKey**. These properties together form the table's primary key and uniquely identify each entity in the table. Entities also have a **Timestamp** system property, which allows the service to keep track of when an entity was last modified. This field is intended for system use and should not be accessed by the application.

The Table Storage client API provides a **TableServiceEntity** class that defines the necessary properties. Although you can use the **TableServiceEntity** class as the base class for your entities, this is not required.

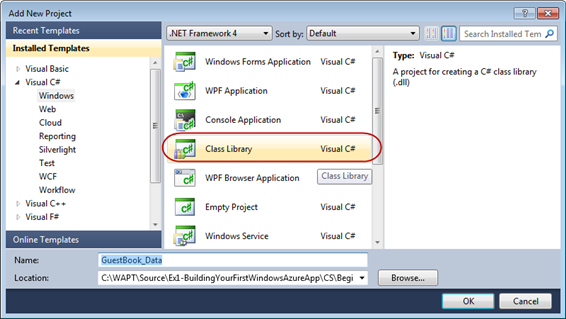
The Table service API is compliant with the REST API provided by [WCF Data Services](http://msdn.microsoft.com/en-us/library/cc668792.aspx) (formerly ADO.NET Data Services Framework) allowing you to use the [WCF Data Services Client Library](http://msdn.microsoft.com/en-us/library/cc668772.aspx) (formerly .NET Client Library) to work with data in Table Storage using .NET objects.

The Table service does not enforce any schema for tables making it possible for two entities in the same table to have different sets of properties. Nevertheless, the GuestBook application uses a fixed schema to store its data.

In order to use the WCF Data Services Client Library to access data in table storage, you need to create a context class that derives from **TableServiceContext**, which itself derives from **DataServiceContext** in WCF Data Services. The Table Storage API allows applications to create the tables that they use from these context classes. For this to happen, the context class must expose each required table as a property of type **IQueryable<*SchemaClass*>**, where ***SchemaClass*** is the class that models the entities stored in the table.

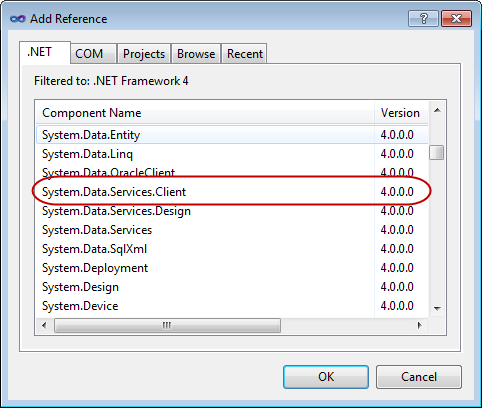
In this task, you model the schema of the entities stored by the GuestBook application and create a context class to use WCF Data Services to access the information in table storage. To complete the task, you create an object that can be data bound to data controls in ASP.NET and implements the basic data access operations: read, update, and delete.

1. Create a new project for the schema classes. To create the project, in the **File** menu, point to **Add** and then select **New Project**.
2. In the **Add New Project** dialog, expand the language of your choice under the **Installed Templates** tree view, select the **Windows** category, and then choose the **Class Library** project template. Set the name to **GuestBook\_Data**, leave the proposed location inside the solution folder unchanged, and then click **OK**.



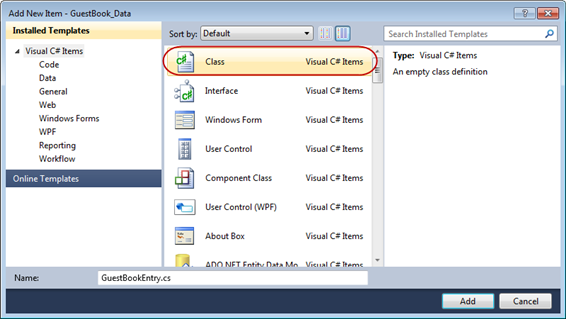
**Figure 7**  
*Creating a class library for GuestBook entities (C#)*

1. Delete the default class file generated by the class library template. To do this, right-click **Class1.cs** (for Visual C# Projects) or **Class1.vb** (for Visual Basic Projects) and choose **Delete**. Click **OK** in the confirmation dialog.
2. Add a reference to the .NET Client Library for WCF Data Services in the **GuestBook\_Data** project. In **Solution Explorer,** right-click the **GuestBook\_Data** project node, select **Add Reference**, click the **.NET** tab, select the **System.Data.Services.Client** component and click **OK**.



**Figure 9**  
*Adding a reference to the System.Data.Service.Client component*

1. Repeat the previous step to add a reference to the Windows Azure storage client API assembly, this time choosing the **Microsoft.WindowsAzure.StorageClient** component instead.
2. Before you can store an entity in a table, you must first define its schema. To do this, right-click **GuestBook\_Data** in **Solution Explorer**, point to **Add** and select **Class**. In the **Add New Item** dialog, set the name to **GuestBookEntry.cs** and click **Add**.



**Figure 10**  
*Adding the GuestBookEntry class (C#)*

1. At the top of the file, insert the following namespace declaration to import the types contained in the **Microsoft.WindowsAzure.StorageClient** namespace.

using Microsoft.WindowsAzure.StorageClient;

1. If not already opened, open the **GuestBookEntry.cs** file and then update the declaration of the **GuestBookEntry** class to make it public and derive from the **TableServiceEntity** class.
2. public class GuestBookEntry

: Microsoft.WindowsAzure.StorageClient.TableServiceEntity

{

}

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| **TableServiceEntity** is a class found in the Storage Client API. This class defines the **PartititionKey**, **RowKey** and **TimeStamp** system properties required by every entity stored in a Windows Azure table.  Together, the **PartitionKey** and **RowKey** define the **DataServiceKey** that uniquely identifies every entity within a table. |

1. Add a default constructor to the **GuestBookEntry** class that initializes its **PartitionKey** and **RowKey** properties.

public GuestBookEntry()

{

PartitionKey = DateTime.UtcNow.ToString("MMddyyyy");

// Row key allows sorting, so we make sure the rows come back in time order.

RowKey = string.Format("{0:10}\_{1}",

DateTime.MaxValue.Ticks – DateTime.Now.Ticks,

Guid.NewGuid());

}

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| To partition the data, the GuestBook application uses the date of the entry as the **PartitionKey**, which means that there will be a separate partition for each day of guest book entries. In general, you choose the value of the partition key to ensure load balancing of the data across storage nodes.  The **RowKey** is a reverse DateTime field with a GUID appended for uniqueness. Tables within partitions are sorted in RowKey order, so this will sort the tables into the correct order to be shown on the home page, with the newest entry shown at the top. |

1. To complete the definition of the the **GuestBookEntry** class, add properties for **Message**, **GuestName**, **PhotoUrl**, and **ThumbnailUrl** to hold information about the entry.

public string Message { get; set; }

public string GuestName { get; set; }

public string PhotoUrl { get; set; }

public string ThumbnailUrl { get; set; }

1. Save the **GuestBookEntry.cs** file.
2. Next, you need to create the context class required to access the *GuestBook* table using WCF Data Services. To do this, in **Solution Explorer**, right-click the **GuestBook\_Data** project, point to **Add** and select **Class**. In the **Add New Item** dialog, set the **Name** to **GuestBookDataContext.cs** and click **Add**.
3. In the new class file, update the declaration of the new class to make it public and inherit the **TableServiceContext** class.
4. public class GuestBookDataContext

: Microsoft.WindowsAzure.StorageClient.TableServiceContext

{

}

1. Now, add a default constructor to initialize the base class with storage account information.

public GuestBookDataContext(string baseAddress, Microsoft.WindowsAzure.StorageCredentials credentials)

: base(baseAddress, credentials)

{ }

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| You can find the **TableServiceContext** class in the storage client API. This class derives from **DataServiceContext** in WCF Data Services and manages the credentials required to access your storage account as well as providing support for a retry policy for its operations. |

1. Add a property to the **GuestBookDataContext** class to expose the **GuestBookEntry** table. To do this, insert the following (highlighted) code into the class.

public class GuestBookDataContext

: Microsoft.WindowsAzure.StorageClient.TableServiceContext

{

...

public IQueryable<GuestBookEntry> GuestBookEntry

{

get

{

return this.CreateQuery<GuestBookEntry>("GuestBookEntry");

}

}

}

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| You can use the **CreateTablesFromModel** method in the **CloudTableClient** class to create the tables needed by the application. When you supply a **DataServiceContext** (or **TableServiceContext**) derived class to this method, it locates any properties that return an **IQueryable<*T*>**, where the generic parameter ***T*** identifies the class that models the table schema, and creates a table in storage named after the property. |

1. Finally, you need to implement an object that can be bound to data controls in ASP.NET. In **Solution Explorer**, right-click **GuestBook\_Data**, point to **Add**, and select **Class**. In the **Add New Item** dialog, set the name to **GuestBookDataSource.cs** and click **Add**.
2. In the new class file, add the following namespace declarations to import the types contained in the **Microsoft.WindowsAzure** and **Microsoft.WindowsAzure.StorageClient** namespaces.

using Microsoft.WindowsAzure;

using Microsoft.WindowsAzure.StorageClient;

1. In the **GuestBookDataSource** class, make the class **public** and define member fields for the data context and the storage account information, as shown below.

public class GuestBookDataSource

{

private static CloudStorageAccount storageAccount;

private GuestBookDataContext context;

}

1. Now, add a static constructor to the data source class as shown in the following (highlighted) code. This code creates the tables from the **GuestBookDataContext** class.

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource Static Constructor – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public class GuestBookDataSource  {  ...  **static GuestBookDataSource()**  **{**  **storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**  **CloudTableClient.CreateTablesFromModel(**  **typeof(GuestBookDataContext),**  **storageAccount.TableEndpoint.AbsoluteUri,**  **storageAccount.Credentials);**  **}**  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource Shared Constructor – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class GuestBookDataSource  ...  **Shared Sub New()**  **storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**  **CloudTableClient.CreateTablesFromModel(GetType(GuestBookDataContext), storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials)**  **End Sub**  End Class | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The static (Shared in Visual Basic) constructor initializes the storage account by reading its settings from the configuration and then uses the **CreateTablesFromModel** method in the **CloudTableClient** class to create the tables used by the application from the model defined by the **GuestBookDataContext** class. By using the static constructor, you ensure that this initialization task is executed only once. |

1. Add a default constructor to the **GuestBookDataSource** class to initialize the data context class used to access table storage.

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource Constructor – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public class GuestBookDataSource  {  ...  **public GuestBookDataSource()**  **{**  **this.context = new GuestBookDataContext(storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials);**  **this.context.RetryPolicy = RetryPolicies.Retry(3, TimeSpan.FromSeconds(1));**  **}**  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource Constructor – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class GuestBookDataSource  ...  **Public Sub New()**  **Me.context = New GuestBookDataContext(storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials)**  **Me.context.RetryPolicy = RetryPolicies.Retry(3, TimeSpan.FromSeconds(1))**  **End Sub**  End Class | |

1. Next, insert the following method to return the contents of the *GuestBookEntry* table.

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource Select – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public class GuestBookDataSource  {  ...  **public IEnumerable<GuestBookEntry> GetGuestBookEntries()**  **{**  **var results = from g in this.context.GuestBookEntry**  **where g.PartitionKey == DateTime.UtcNow.ToString("MMddyyyy")**  **select g;**  **return results;**  **}**  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource Select – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class GuestBookDataSource  ...  **Public Function GetGuestBookEntries() As IEnumerable(Of GuestBookEntry)**  **Dim results = From g In Me.context.GuestBookEntry \_**  **Where g.PartitionKey = DateTime.UtcNow.ToString("MMddyyyy") \_**  **Select g**  **Return results**  **End Function**  End Class | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The **GetGuestBookEntries** method retrieves today's guest book entries by constructing a LINQ statement that filters the retrieved information using the current date as the partition key value. The web role uses this method to bind to a data grid and display the guest book. |

1. Now, add the following method to insert new entries into the *GuestBookEntry* table.

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource AddGuestBookEntry – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public class GuestBookDataSource  {  ...  **public void AddGuestBookEntry(GuestBookEntry newItem)**  **{**  **this.context.AddObject("GuestBookEntry", newItem);**  **this.context.SaveChanges();**  **}**  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource AddGuestBookEntry – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class GuestBookDataSource  ...  **Public Sub AddGuestBookEntry(ByVal newItem As GuestBookEntry)**  **Me.context.AddObject("GuestBookEntry", newItem)**  **Me.context.SaveChanges()**  **End Sub**  End Class | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| This method adds a new **GuestBookEntry** object to the data context and then calls **SaveChanges** to write the entity to storage. |

1. Finally, add a method to the data source class to update the thumbnail URL property for an entry.

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource UpdateImageThumbnail – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public class GuestBookDataSource  {  ...  **public void UpdateImageThumbnail(string partitionKey, string rowKey, string thumbUrl)**  **{**  **var results = from g in this.context.GuestBookEntry**  **where g.PartitionKey == partitionKey && g.RowKey == rowKey**  **select g;**  **var entry = results.FirstOrDefault<GuestBookEntry>();**  **entry.ThumbnailUrl = thumbUrl;**  **this.context.UpdateObject(entry);**  **this.context.SaveChanges();**  **}**  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 GuestBookDataSource UpdateImageThumbnail – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class GuestBookDataSource  ...  **Public Sub UpdateImageThumbnail(ByVal partitionKey As String, ByVal rowKey As String, ByVal thumbUrl As String)**  **Dim results = From g In Me.context.GuestBookEntry \_**  **Where g.PartitionKey = partitionKey AndAlso g.RowKey = rowKey \_**  **Select g**  **Dim entry = results.FirstOrDefault()**  **entry.ThumbnailUrl = thumbUrl**  **Me.context.UpdateObject(entry)**  **Me.context.SaveChanges()**  **End Sub**  End Class | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The **UpdateImageThumbnail** method locates an entry using its partition key and row key; it updates the thumbnail URL, notifies the data context of the update, and then saves the changes. |

1. Save the **GuestBookDataSource.cs** file (for Visual C# projects) or **GuestBookDataSource.vb** file (for Visual Basic projects).

**Task 3 – Creating a Web Role to Display the Guest Book and Process User Input**

In this task, you update the web role project that you generated in Task 1, when you created the Windows Azure Cloud Service solution. This involves updating the UI to render the list of guest book entries. For this purpose, you will find a page that has the necessary elements in the **Assets** folder of this exercise, which you will add to the project. Next, you implement the code necessary to store submitted entries in table storage and images in blob storage. To complete this task, you configure the storage account used by the Web role.

1. Add a reference in the web role to the **GuestBook\_Data** project. In **Solution Explorer**, right-click the **GuestBook\_WebRole** project node and select **Add Reference**, switch to the **Projects** tab, select the **GuestBook\_Data** project, and then click then **OK**.
2. The web role template generates a default page. You will replace it with another page that contains the UI of the guest book application. To delete the page, in **Solution Explorer**, right-click **Default.aspx** in the **GuestBook\_WebRole** project and select **Delete**.
3. Add the main page and its associated assets to the web role. To do this, right-click **GuestBook\_WebRole** in **Solution Explorer**, point to **Add** and select **Existing Item**. In the **Add Existing Item** dialog, browse to the **Assets** folder in the **Source\Ex1-BuildingYourFirstWindowsAzureApp** for the language of your project (Visual C# or Visual Basic), hold the **CTRL** key down while you select every file in this folder and click **Add**.

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The **Assets** folder contains five files that you need to add to the project, a Default.aspx file with its code-behind and designer files, a CSS file, and an image file. |

1. Open the code-behind file for the main page in the **GuestBook\_WebRole** project. To do this, right-click the **Default.aspx** file in **Solution Explorer** and select **View Code**.
2. In the code-behind file, insert the following namespace declarations.

(Code Snippet*– Introduction to Windows Azure - Ex1 Web Role Namespace Declarations – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| **using System.IO;**  **using System.Net;**  **using Microsoft.WindowsAzure;**  **using Microsoft.WindowsAzure.ServiceRuntime;**  **using Microsoft.WindowsAzure.StorageClient;**  **using GuestBook\_Data;** | |

(Code Snippet*– Introduction to Windows Azure - Ex1 Web Role Namespace Declarations – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| **Imports System.IO**  **Imports System.Net**  **Imports Microsoft.WindowsAzure**  **Imports Microsoft.WindowsAzure.ServiceRuntime**  **Imports Microsoft.WindowsAzure.StorageClient**  **Imports GuestBook\_Data** | |

1. Declare the following member fields in the **Default** class.

(Code Snippet – *Introduction to Windows Azure - Ex1 Web Role Member Fields – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public partial class Default : System.Web.UI.Page  {  **private static bool storageInitialized = false;**  **private static object gate = new Object();**  **private static CloudBlobClient blobStorage;**  ...  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 Web Role Member Fields – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Partial Public Class \_Default  Inherits System.Web.UI.Page  **Private Shared storageInitialized As Boolean = False**  **Private Shared gate As New Object()**  **Private Shared blobStorage As CloudBlobClient**  ...  End Class | |

1. Locate the **SignButton\_Click** event handler in the code-behind file and insert the following code.

(Code Snippet*– Introduction to Windows Azure - Ex1 SignButton\_Click – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public partial class Default : System.Web.UI.Page  {  ...  protected void SignButton\_Click(object sender, EventArgs e)  {  **if (FileUpload1.HasFile)**  **{**  **InitializeStorage();**  **// upload the image to blob storage**  **string uniqueBlobName = string.Format("guestbookpics/image\_{0}{1}", Guid.NewGuid().ToString(), Path.GetExtension(FileUpload1.FileName));**  **CloudBlockBlob blob = blobStorage.GetBlockBlobReference(uniqueBlobName);**  **blob.Properties.ContentType = FileUpload1.PostedFile.ContentType;**  **blob.UploadFromStream(FileUpload1.FileContent);**  **System.Diagnostics.Trace.TraceInformation("Uploaded image '{0}' to blob storage as '{1}'", FileUpload1.FileName, uniqueBlobName);**  **// create a new entry in table storage**  **GuestBookEntry entry = new GuestBookEntry() { GuestName = NameTextBox.Text, Message = MessageTextBox.Text, PhotoUrl = blob.Uri.ToString(), ThumbnailUrl = blob.Uri.ToString() };**  **GuestBookDataSource ds = new GuestBookDataSource();**  **ds.AddGuestBookEntry(entry);**  **System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName);**  **}**  **NameTextBox.Text = "";**  **MessageTextBox.Text = "";**  **DataList1.DataBind();**  }  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 SignButton\_Click – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class \_Default  Inherits System.Web.UI.Page  ...  Protected Sub SignButton\_Click(ByVal sender As Object, ByVal e As EventArgs) Handles SignButton.Click  **If FileUpload1.HasFile Then**  **InitializeStorage()**  **' upload the image to blob storage**  **Dim uniqueBlobName As String = String.Format("guestbookpics/image\_{0}{1}", Guid.NewGuid().ToString(), Path.GetExtension(FileUpload1.FileName))**  **Dim blob As CloudBlockBlob = blobStorage.GetBlockBlobReference(uniqueBlobName)**  **blob.Properties.ContentType = FileUpload1.PostedFile.ContentType**  **blob.UploadFromStream(FileUpload1.FileContent)**  **System.Diagnostics.Trace.TraceInformation("Uploaded image '{0}' to blob storage as '{1}'", FileUpload1.FileName, uniqueBlobName)**  **' create a new entry in table storage**  **Dim entry As New GuestBookEntry() With {.GuestName = NameTextBox.Text, .Message = MessageTextBox.Text, .PhotoUrl = blob.Uri.ToString(), .ThumbnailUrl = blob.Uri.ToString()}**  **Dim ds As New GuestBookDataSource()**  **ds.AddGuestBookEntry(entry)**  **System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName)**  **End If**  **NameTextBox.Text = ""**  **MessageTextBox.Text = ""**  **DataList1.DataBind()**  End Sub  End Class | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| To process a new guest book entry after the user submits the page, the handler first calls the **InitializeStorage** method to ensure that the blob container used to store images exists and allows public access. You will implement this method shortly.  It then obtains a reference to the blob container, generates a unique name and creates a new blob, and then uploads the image submitted by the user into this blob. Notice that the method initializes the **ContentType** property of the blob from the content type of the file submitted by the user. When the guest book page reads the blob back from storage, the response returns this content type, allowing a page to display the image contained in the blob simply by referring to its URL. After that, it creates a new **GuestBookEntry** entity, which is the entity you defined in the previous task, initializes it with the information submitted by the user, and then uses the **GuestBookDataSource** class to save the entry to table storage using the .NET Client Library for WCF Data Services. Finally, it data binds the guest book entries list to refresh its contents. |

1. Update the body of the **Timer1\_Tick** method with the code shown below.

(Code Snippet*– Introduction to Windows Azure - Ex1 Timer1\_Tick – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public partial class Default : System.Web.UI.Page  {  ...  protected void Timer1\_Tick(object sender, EventArgs e)  {  **DataList1.DataBind();**  }  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 Timer1\_Tick – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class \_Default  Inherits System.Web.UI.Page  ...  Protected Sub Timer1\_Tick(ByVal sender As Object, ByVal e As EventArgs) Handles Timer1.Tick  **DataList1.DataBind()**  End Sub  End Class | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The timer periodically forces the page to refresh the contents of the guest book entries list. |

1. Locate the **Page\_Load** event handler and update its body with the following code to enable the page refresh timer.

(Code Snippet – *Introduction to Windows Azure - Ex1 Page\_Load – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public partial class Default : System.Web.UI.Page  {  ...  protected void Page\_Load(object sender, EventArgs e)  {  **if (!Page.IsPostBack)**  **{**  **Timer1.Enabled = true;**  **}**  }  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 Page\_Load – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class \_Default  Inherits System.Web.UI.Page  ...  Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load  **If Not (Page.IsPostBack) Then**  **Timer1.Enabled = True**  **End If**  End Sub  End Class | |

1. Implement the **InitializeStorage** method by replacing its body with the following (highlighted) code.

(Code Snippet – *Introduction to Windows Azure - Ex1 InitializeStorage – CS*)

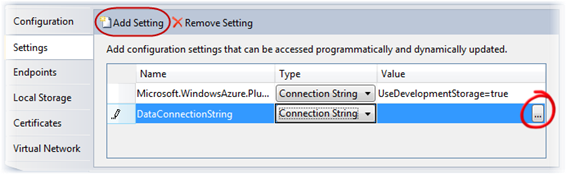
|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public partial class Default : System.Web.UI.Page  {  ...  private void InitializeStorage()  {  **if (storageInitialized)**  **{**  **return;**  **}**  **lock (gate)**  **{**  **if (storageInitialized)**  **{**  **return;**  **}**  **try**  **{**  **// read account configuration settings**  **var storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");**  **// create blob container for images**  **blobStorage = storageAccount.CreateCloudBlobClient();**  **CloudBlobContainer container = blobStorage.GetContainerReference("guestbookpics");**  **container.CreateIfNotExist();**  **// configure container for public access**  **var permissions = container.GetPermissions();**  **permissions.PublicAccess = BlobContainerPublicAccessType.Container;**  **container.SetPermissions(permissions);**  **}**  **catch (WebException)**  **{**  **throw new WebException("Storage services initialization failure. "**  **+ "Check your storage account configuration settings. If running locally, "**  **+ "ensure that the Development Storage service is running.");**  **}**  **storageInitialized = true;**  **}**  }  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 InitializeStorage – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class \_Default  Inherits System.Web.UI.Page  ...  Private Sub InitializeStorage()  **If storageInitialized Then**  **Return**  **End If**  **SyncLock gate**  **If storageInitialized Then**  **Return**  **End If**  **Try**  **' read account configuration settings**  **Dim storageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")**  **' create blob container for images**  **blobStorage = storageAccount.CreateCloudBlobClient()**  **Dim container As CloudBlobContainer = blobStorage.GetContainerReference("guestbookpics")**  **container.CreateIfNotExist()**  **' configure container for public access**  **Dim permissions = container.GetPermissions()**  **permissions.PublicAccess = BlobContainerPublicAccessType.Container**  **container.SetPermissions(permissions)**  **Catch e1 As WebException**  **Throw New WebException("Storage services initialization failure. " \_**  **& "Check your storage account configuration settings. If running locally, " \_**  **& "ensure that the Development Storage service is running.")**  **End Try**  **storageInitialized = True**  **End SyncLock**  End Sub  End Class | |

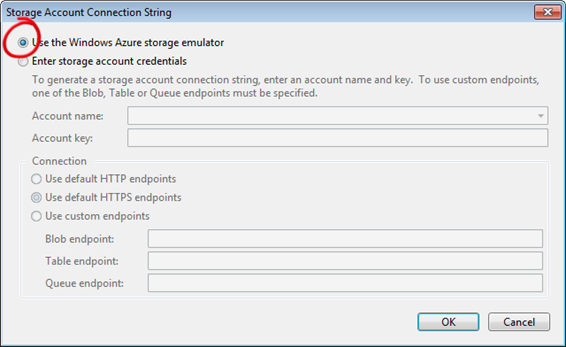
|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The **InitializeStorage** method first ensures that it executes only once. It reads the storage account settings from the Web role configuration, creates a blob container for the images uploaded with each guest book entry and configures it for public access. |

1. Because the web role uses Windows Azure storage services, you need to provide your storage account settings. To create a new setting, in **Solution Explorer**, expand the **Roles** node in the **GuestBook** project, double-click **GuestBook\_WebRole** to open the properties for this role and select the **Settings** tab. Click **Add Setting**, type “*DataConnectionString*” in the **Name** column, change the **Type** to *ConnectionString*, and then click the button labeled with an ellipsis.



**Figure 12**  
*Configuring the storage account settings*

1. In the **Storage Account Connection String** dialog, choose the option labeled **Use the Windows Azure storage emulator** and then click **OK**.



**Figure 13**  
*Creating a connection string for the storage emulator*

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| A storage account is a unique endpoint for the Windows Azure Blob, Queue, and Table services. You must create a storage account in the Management Portal to use these services. In this exercise, you use Windows Azure storage emulator, which is included in the Windows Azure SDK development environment to simulate the Blob, Queue, and Table services available in the cloud. If you are building a hosted service that employs storage services or writing any external application that calls storage services, you can test locally against the Windows Azure storage emulator. To use the storage emulator, you set the value of the **UseDevelopmentStorage** keyword in the connection string for the storage account to *true*. When you deploy your application to Windows Azure, you need to update the connection string to specify storage account settings including your account name and shared key. For example, <Setting name="DataConnectionString" value="DefaultEndpointsProtocol=https;AccountName=YourAccountName;AccountKey=YourAccountKey" /> where *YourAccountName* is the name of your Azure Storage account and *YourAccountKey* is your access key. |

1. Press **CTRL + S** to save changes to the role configuration.
2. Finally, you need to set up the environment for the configuration publisher. In the **GuestBook\_WebRole** project, open the **Global.asax.cs** file (for Visual C# projects) or the **Global.asax.vb** file (for Visual Basic projects).
3. At the top of the file, insert the following namespace declaration to import the types contained in the **Microsoft.WindowsAzure** and **Microsoft.WindowsAzure.ServiceRuntime** namespaces.

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| using Microsoft.WindowsAzure;  using Microsoft.WindowsAzure.ServiceRuntime; | |

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Imports Microsoft.WindowsAzure  Imports Microsoft.WindowsAzure.ServiceRuntime | |

1. Insert the following code into the **Application\_Start** method replacing the default comment.

(Code Snippet – *Introduction to Windows Azure - Ex1 SetConfigurationSettingPublisher – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| void Application\_Start(object sender, EventArgs e)  {  **Microsoft.WindowsAzure.CloudStorageAccount.SetConfigurationSettingPublisher((configName, configSetter) =>**  **{**  **configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));**  **});**  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 SetConfigurationSettingPublisher – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Sub Application\_Start(ByVal sender As Object, ByVal e As EventArgs)  **CloudStorageAccount.SetConfigurationSettingPublisher(Function(configName, configSetter) configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))**  End Sub | |

**Task 4 – Queuing Work Items for Background Processing**

In preparation for the next exercise, you now update the front-end web role to dispatch work items to an Azure queue for background processing. These work items will remain in the queue until you add a worker role that picks items from the queue and generates thumbnails for each uploaded image.

1. Open the code-behind file for the main page of the web role project. To do this, right-click the **Default.aspx** file in **Solution Explorer** and select **View Code**.
2. Declare a queue client member by inserting the following (highlighted) declaration into the **Default** class.

(Code Snippet – *Introduction to Windows Azure - Ex1 CloudQueueClient member – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public partial class \_Default : System.Web.UI.Page  {  private static bool storageInitialized = false;  private static object gate = new Object();  private static CloudBlobClient blobStorage;  **private static CloudQueueClient queueStorage;**  ...  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 CloudQueueClient member – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Public Class \_Default  Inherits System.Web.UI.Page  Private Shared storageInitialized As Boolean = False  Private Shared gate As New Object()  Private Shared blobStorage As CloudBlobClient  **Private Shared queueStorage As CloudQueueClient**  ...  End Class | |

1. Now, update the storage initialization code to create the queue, if it does not exist, and then initialize the queue reference created in the previous step. To do this, locate the **InitializeStorage** method and insert the following (highlighted) code into this method immediately after the code that configures the blob container for public access.

(Code Snippet – *Introduction to Windows Azure - Ex1 Create Queue – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| public partial class Default : System.Web.UI.Page  {  ...  private void InitializeStorage()  {  ...  try  {  ...  // configure container for public access  var permissions = container.GetPermissions();  permissions.PublicAccess = BlobContainerPublicAccessType.Container;  container.SetPermissions(permissions);  **// create queue to communicate with worker role**  **queueStorage = storageAccount.CreateCloudQueueClient();**  **CloudQueue queue = queueStorage.GetQueueReference("guestthumbs");**  **queue.CreateIfNotExist();**  }  catch (WebException)  {  ...  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 Create Queue – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Partial Public Class \_Default  Inherits System.Web.UI.Page  ...  Private Sub InitializeStorage()  ...  Try  ...  ' configure container for public access  Dim permissions = container.GetPermissions()  permissions.PublicAccess = BlobContainerPublicAccessType.Container  container.SetPermissions(permissions)  **' create queue to communicate with worker role**  **queueStorage = storageAccount.CreateCloudQueueClient()**  **Dim queue As CloudQueue = queueStorage.GetQueueReference("guestthumbs")**  **queue.CreateIfNotExist()**  Catch e1 As WebException  ...  End Class | |

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The updated code creates a queue that the web role uses to submit new jobs to the worker role. |

1. Finally, add code to post a work item to the queue. To do this, locate the **SignButton\_Click** event handler and insert the following (highlighted) code immediately after the lines that create a new entry in table storage.

(Code Snippet – *Introduction to Windows Azure - Ex1 Queueing work items – CS*)

|  |  |
| --- | --- |
| **C#** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| protected void SignButton\_Click(object sender, EventArgs e)  {  if (FileUpload1.HasFile)  {  ...  // create a new entry in table storage  GuestBookEntry entry = new GuestBookEntry() { GuestName = NameTextBox.Text, Message = MessageTextBox.Text, PhotoUrl = blob.Uri.ToString(), ThumbnailUrl = blob.Uri.ToString() };  GuestBookDataSource ds = new GuestBookDataSource();  ds.AddGuestBookEntry(entry);  System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName);  **// queue a message to process the image**  **var queue = queueStorage.GetQueueReference("guestthumbs");**  **var message = new CloudQueueMessage(String.Format("{0},{1},{2}", blob.Uri.ToString(), entry.PartitionKey, entry.RowKey));**  **queue.AddMessage(message);**  **System.Diagnostics.Trace.TraceInformation("Queued message to process blob '{0}'", uniqueBlobName);**  }  NameTextBox.Text = "";  MessageTextBox.Text = "";  DataList1.DataBind();  } | |

(Code Snippet – *Introduction to Windows Azure - Ex1 Queueing work items – VB*)

|  |  |
| --- | --- |
| **Visual Basic** | **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\copycode.gifCopy Code** |
| Protected Sub SignButton\_Click(ByVal sender As Object, ByVal e As EventArgs) Handles SignButton.Click  If FileUpload1.HasFile Then  ...  ' create a new entry in table storage  Dim entry As New GuestBookEntry() With {.GuestName = NameTextBox.Text, .Message = MessageTextBox.Text, .PhotoUrl = blob.Uri.ToString(), .ThumbnailUrl = blob.Uri.ToString()}  Dim ds As New GuestBookDataSource()  ds.AddGuestBookEntry(entry)  System.Diagnostics.Trace.TraceInformation("Added entry {0}-{1} in table storage for guest '{2}'", entry.PartitionKey, entry.RowKey, entry.GuestName)  **' queue a message to process the image**  **Dim queue = queueStorage.GetQueueReference("guestthumbs")**  **Dim message = New CloudQueueMessage(String.Format("{0},{1},{2}", blob.Uri.ToString(), entry.PartitionKey, entry.RowKey))**  **queue.AddMessage(message)**  **System.Diagnostics.Trace.TraceInformation("Queued message to process blob '{0}'", uniqueBlobName)**  End If  NameTextBox.Text = ""  MessageTextBox.Text = ""  DataList1.DataBind()  End Sub | |

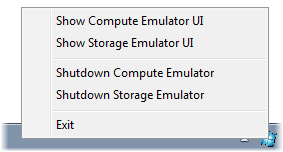
|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| The updated code obtains a reference to the “*guestthumbs*” queue. It constructs a new message that consists of a comma-separated string with the name of the blob that contains the image, the partition key, and the row key of the entity that was added. The worker role can easily parse messages with this format. The method then submits the message to the queue. |

**Verification**

The Windows Azure compute emulator, formerly Development Fabric or devfabric, is a simulated environment for developing and testing Windows Azure applications in your machine. In this task, you launch the GuestBook application in the emulator and create one or more guest book entries.

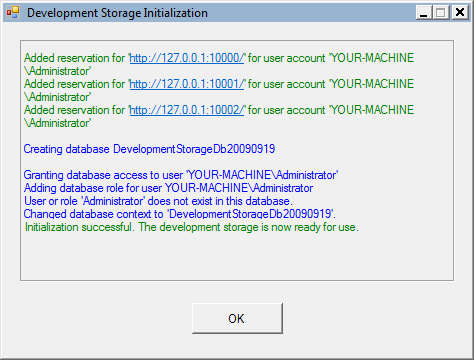
Among the features available in the Windows Azure Tools for Microsoft Visual Studio is a Windows Azure Storage browser that allows you to connect to a storage account and browse the blobs and tables it contains. If you are using this version of Visual Studio, you will use it during this task to examine the storage resources created by the application.

1. Press **F5** to execute the service. The service builds and then launches the local Windows Azure compute emulator. To show the Compute Emulator UI, right-click its icon located in the system tray and select **Show Compute Emulator UI**.



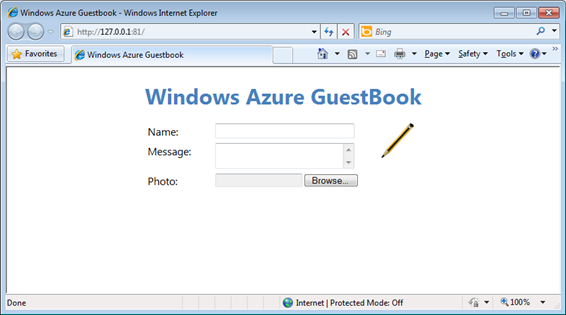
**Figure 14**  
*Showing the Compute Emulator UI*

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| When you use the storage emulator for the first time, it needs to execute a one-time initialization procedure to create the necessary database and tables. If this is the case, wait for the procedure to complete and examine the **Development Storage Initialization** dialog to ensure that it completes successfully. |



**Figure 15**  
*Storage emulator initialization process*

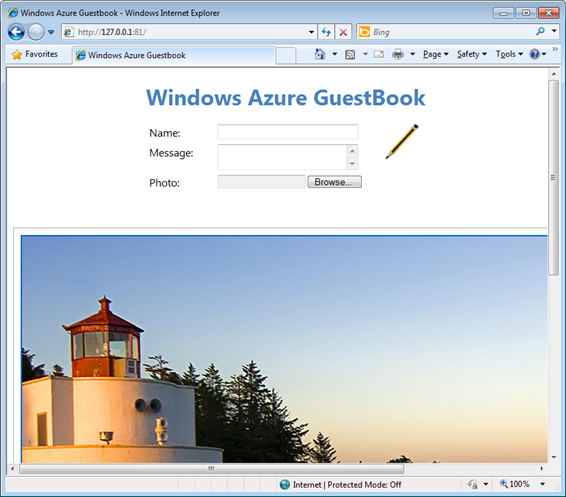
1. Switch to Internet Explorer to view the GuestBook application.
2. Add a new entry to the guest book. To do this, type your name and a message, choose an image to upload, and then click the pencil icon to submit the entry.



**Figure 16**  
*Windows Azure GuestBook home page*

|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| It is a good idea to choose a large hi-resolution image because, once the application is complete, the guestbook service will resize uploaded images. |

Once you submit an entry, the web role creates a new entity in the guest book table and uploads the photo to blob storage. The page contains a timer that triggers a page refresh every 5 seconds, so the new entry should appear on the page after a brief interval. Initially, the new entry contains a link to the blob that contains the uploaded image so it will appear with the same size as the original image.



**Figure 17**  
*GuestBook application showing an uploaded image in its original size*

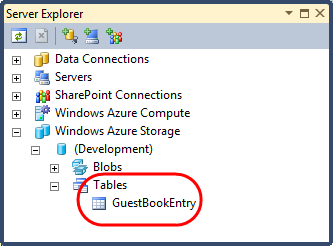
If you are using Visual Studio 2010, you can use the Windows Azure Storage Explorer to view storage resources directly from Visual Studio. This functionality is not available in Visual Studio 2008.

1. To open the Storage Explorer in Visual Studio 2010, open the **View** menu, select **Server Explorer**, and then expand the **Windows Azure Storage** node.

The **Windows Azure Storage** node lists the storage accounts that you have currently registered and, by default, includes an entry for the storage emulator account labeled as **(Development)**.

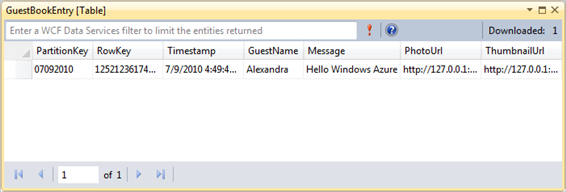
|  |
| --- |
| **C:\WindowsAzurePlatformTrainingKit\Labs\IntroductionToWindowsAzureVS2010\Lab.html\local\note.gifNote:** |
| Windows Azure Storage Explorer is not available in Visual Studio 2008. |

1. Expand the **(Development)** node and then the **Tables** node inside it. Notice that it contains a table named *GuestBookEntry* created by the application that should contain details for each entry.



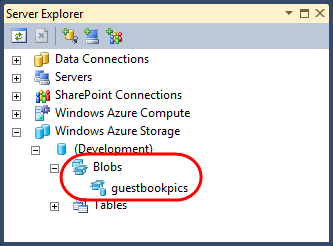
**Figure 18**  
*Viewing tables in the Windows Azure storage emulator*

1. Double-click the *GuestBookEntry* node in the **Windows Azure Storage** explorer to show the contents of this table. The *GuestBookEntry* table contains information for the entry that you created earlier in this task, including its *GuestName*, *Message*, *PhotoUrl*, and *ThumbnailUrl* properties, as well as the *PartitionKey*, *RowKey*, and *Timestamp* properties common to all table storage entities. Notice that the *PhotoUrl* and *ThumbnailUrl* properties are currently the same. In the next exercise, you will modify the application to generate image thumbnails and to update the corresponding URL.



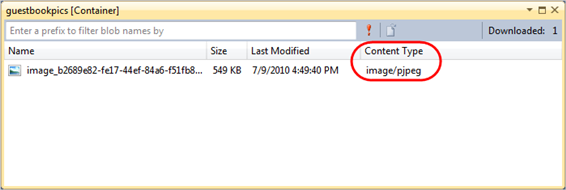
**Figure 19**  
*Viewing tables using the Windows Azure Tools for Visual Studio*

1. Now, expand the **Blobs** node in the **Windows Azure Storage** explorer. Inside this node, you will find an entry for a container named *guestbookpics* that contains blobs with raw data for the images uploaded by the application.



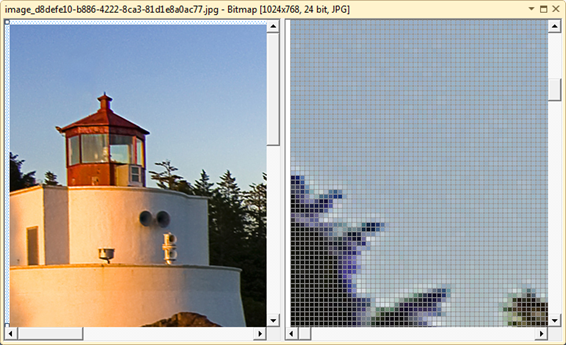
**Figure 20**  
*Viewing blobs using the Windows Azure Tools for Visual Studio*

1. Double-click the node for the *guestbookpics* container to list the blobs it contains. It should include an entry for the image that you uploaded earlier.



**Figure 21**  
*Viewing the contents of a blob container in Visual Studio*

1. Each blob in blob storage has an associated content type that Visual Studio uses to select a suitable viewer for the blob. To display the contents of the blob, double-click the corresponding entry in the container listing to display the image.



**Figure 22**  
*Viewing blob contents in Visual Studio*

1. Press **SHIFT + F5** to stop the debugger and shut down the deployment in the development fabric.