Windows Azure

A case study on the cloud operating system by Microsoft Corporation

S.Sharmili Priyadarsini

Windows Azure

A case study on the cloud operating system by Microsoft Corporation

By

S.Sharmili Priyadarsini

Third year, CSE

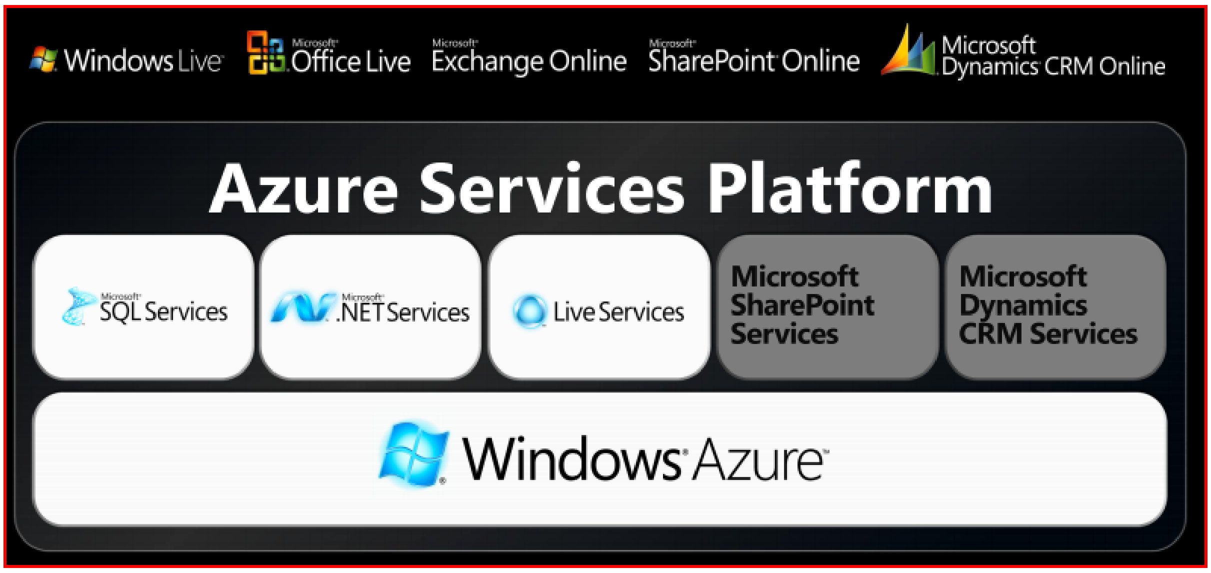
PSNA College of Engineering and Technology,

Dindigul.

Index

Introduction:

- Windows azure is a cloud operating system, developed by Microsoft Corporation.
- It provides various web services and enables the users to develop their own applications on the cloud platform.
- Azure service platforms are Microsoft SQL services, Microsoft.NET services, Microsoft live services, Microsoft sharepoint services, Microsoft CRM services.


Windows Live
Microsoft Office Live
Microsoft Exchange Online
Microsoft SharePoint Online
Microsoft Dynamics CRM Online
Azure Services Platform
Microsoft SQL Services
Microsoft .NET Services
Live Services
Microsoft SharePoint Services
Microsoft Dynamics CRM Services
Windows Azure

- Azure platform provides computation, storage and management of resources.

What is Cloud Computing?

A virtualized computing platform that provides

- Infinite resources for running your applications–
- It leverages economies of scale to save you money by only requiring you to pay for what you use.

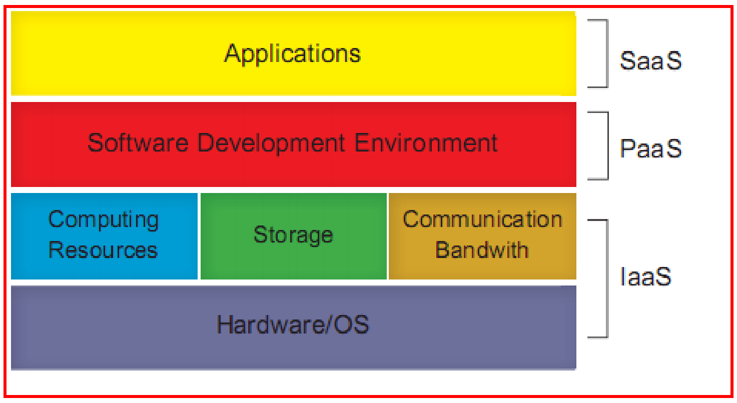
Features of Cloud computing are

- Scalability
- Virtuality
- Availability
- Efficiency
- Flexibility
- Cost saving

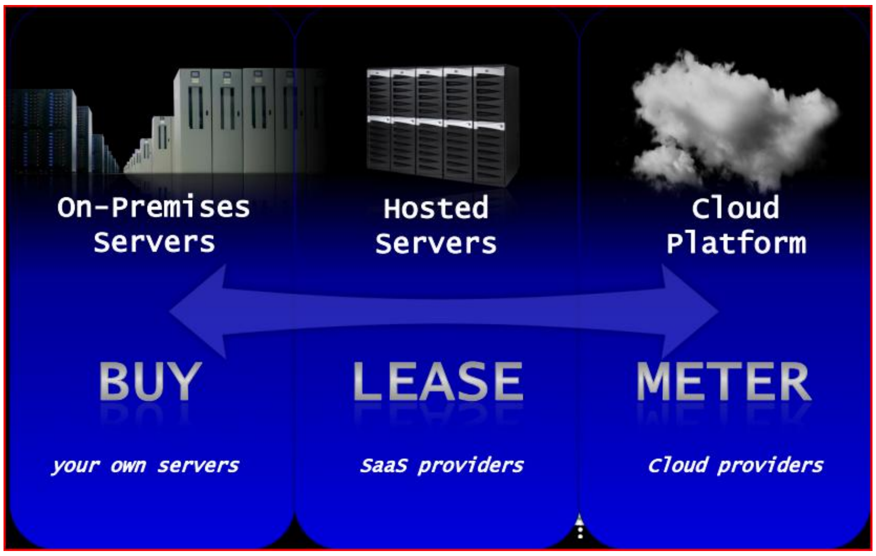
Cloud computing exist as

- Software as service (SaaS)
- Infrastructure as service(Iaas)

- Platform as service(PaaS)



Platform continuum:



New Aspects provided by cloud:

- Infinite computing resources, available on-demand
- No up-front commitment by cloud users
- Only pay for what you use, short term billing

Windows Azure:

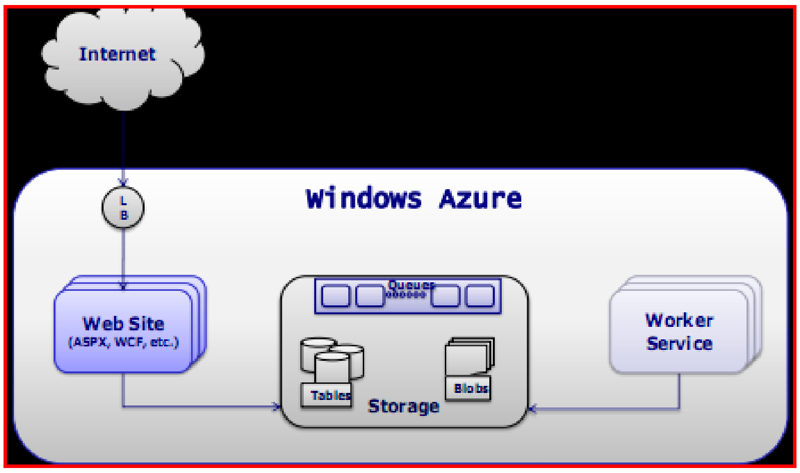
 Scalable, virtualized hosting environment

� Flexible storage with blobs, tables, and queues

� Model-driven service lifecycle management

� Rich local and offline developer experience

Windows Azure Architecture:



Running applications on machines in an Internet-accessible data center can bring plenty of advantages. Yet wherever they run, applications are built on some kind of platform. For on-premises applications, this platform usually includes an operating system, some way to store data, and perhaps more. Applications running in the cloud need a similar foundation. The goal of Microsoft's Windows Azure is to provide this. Part of the larger Azure Services Platform, Windows Azure is a platform for running Windows applications and storing data in the cloud.

Windows Azure runs on machines in Microsoft data centers. Rather than providing software that Microsoft customers can install and run themselves on their own computers, Windows

Azure is a service: Customers use it to run applications and store data on Internet-accessible machines owned by Microsoft. Those applications might provide services to businesses, to consumers, or both. Here are some examples of the kinds of applications that might be built on Windows Azure:

- An independent software vendor (ISV) could create an application that targets business users, an approach that's often referred to as Software as a Service (SaaS). ISVs can use Windows Azure as a foundation for a variety of business-oriented SaaS applications.
- An ISV might create a SaaS application that targets consumers. Windows Azure is designed to support very scalable software, and so a firm that plans to target a large consumer market might well choose it as a foundation for a new application.
- Enterprises might use Windows Azure to build and run applications that are used by their own employees. While this situation probably won't require the enormous scale of a consumer-facing application, the reliability and manageability that Windows Azure offers could still make it an attractive choice.

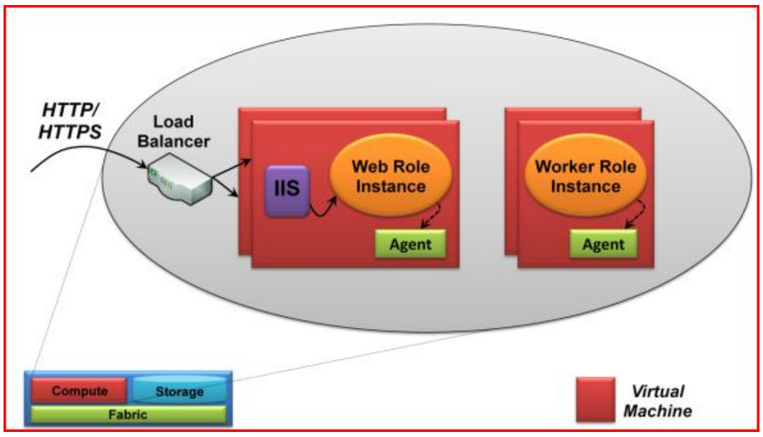


Fig: In the CTP version, Windows Azure applications can consist of Web role instances and/or Worker role instances, each of which runs in its own style of virtual machine.

1. THE COMPUTE SERVICE:

The Windows Azure Compute service can run many different kinds of applications. A primary goal of this platform, however, is to support applications that have a very large number of simultaneous users. (In fact, Microsoft has said that it will build its own SaaS applications on Windows Azure, which sets the bar high.) Reaching this goal by scaling up—running on bigger and bigger machines—isn't possible. Instead, Windows Azure is designed to support applications that scale out, running multiple copies of the same code across many commodity servers.

To allow this, a Windows Azure application can have multiple instances, each executing in its own virtual machine (VM). These VMs run 64-bit Windows Server 2008, and they're provided by a hypervisor (based on Hyper-V) that's been modified for use in Microsoft's cloud. To run an application, a developer accesses the Windows Azure portal through her Web browser, signing in with a Windows Live ID. She then chooses whether to create a hosting account for running applications, a storage account for storing data, or both.

Once the developer has a hosting account, she can upload her application, specifying how many instances the application needs. Windows Azure then creates the necessary VMs and runs the application. It's important to note that a developer can't supply her own VM image for Windows Azure to run. Instead, the platform itself provides and maintains its own copy of Windows. Developers focus solely on creating applications that run on Windows Azure.

In the initial incarnation of Windows Azure, known as the Community Technology Preview (CTP), two different instance types are available for developers to use: Web role instances and Worker role instances.

As its name suggests, a Web role instance can accept incoming HTTP or HTTPS requests. To allow this, it runs in a VM that includes Internet Information Services (IIS) 7. Developers can create Web role instances using ASP.NET, WCF, or another .NET technology that works with IIS. Developers can also create applications in native code—using the .NET Framework isn't required. (This means that developers can upload and run other technologies as well, such as PHP.) And as Figure above shows, Windows Azure provides built-in hardware load balancing to spread requests across Web role instances that are part of the same application.

By running multiple instances of an application, Windows Azure helps that application scale. To accomplish this, however, Web role instances must be stateless. Any client-specific state should be written to Windows Azure storage or passed back to the client after each request. Also, because the Windows Azure load balancer doesn't allow creating an affinity with a particular Web role instance, there's no way to guarantee that multiple requests from the same user will be sent to the same instance.

Worker role instances aren't quite the same as their Web role cousins. For example, they can't accept requests from the outside world. Their VMs don't run IIS, and a Worker application can't accept any incoming network connections. Instead, a Worker role instance initiates its own requests for input. It can read messages from a queue, for instance, as described later, and it can open connections with the outside world. Given this more self-directed nature, Worker role instances can be viewed as akin to a batch job or a Windows service.

A developer can use only Web role instances, only Worker role instances, or a combination of the two to create a Windows Azure application. If the application's load increases, he can use the Windows Azure portal to request more Web role instances, more Worker role instances, or more of both for his application. If the load decreases, he can reduce the number of running instances. To shut down the application completely, the developer can shut down all of the application's Web role and Worker role instances.

The VMs that run both Web role and Worker role instances also run a Windows Azure agent, as Figure above shows. This agent exposes a relatively simple API that lets an instance interact with the Windows Azure fabric. For example, an instance can use the agent to write to a Windows Azure-maintained log, send alerts to its owner via the Windows Azure fabric, and do a few more things.

To create Windows Azure applications, a developer uses the same languages and tools as for any Windows application. She might write a Web role using ASP.NET and Visual Basic, for example, or with WCF and C#. Similarly, she might create a Worker role in one of these .NET languages or directly in C++ without the .NET Framework. And while Windows Azure provides add-ins for Visual Studio, using this development environment isn't required. A developer who has installed PHP, for example, might choose to use another tool to write applications.

Both Web role instances and Worker role instances are free to access their VM's local file system. This storage isn't persistent, however: When the instance is shut down, the VM and its local storage go away.

Yet applications commonly need persistent storage that holds on to information even when they're not running. Meeting this need is the goal of the Windows Azure Storage service, described next.

2. THE STORAGE SERVICE:

Applications work with data in many different ways. Accordingly, the Windows Azure Storage service provides several options. Figure below shows what's in the CTP version of this technology.

The simplest way to store data in Windows Azure storage is to use blobs. A blob contains binary data, and as Figure below suggests, there's a simple hierarchy: A storage account can have one or more containers, each of which holds one or more blobs. Blobs can be big—up to 50 gigabytes each—and they can also have associated metadata, such as information about where a JPEG photograph was taken or who the singer is for an MP3 file.

Blobs are just right for some situations, but they're too unstructured for others. To let applications work with data in a more fine-grained way, Windows Azure storage provides tables. Don't be misled by the name: These aren't relational tables. In fact, even though they're called "tables", the data they hold is actually stored in a simple hierarchy of entities that contain properties. And rather than using SQL, an application accesses a table's data using the conventions defined by ADO.NET Data Services. The reason for this apparently idiosyncratic approach is that it allows scale-out storage—scaling by spreading data spread across many machines—much more effectively than would a standard relational database. In fact, a single Windows Azure table can contain billions of entities holding terabytes of data.

Blobs and tables are both focused on storing and accessing data. The third option in Windows Azure storage, queues, has a quite different purpose. A primary function of queues is to provide a way for Web role instances to communicate with Worker role instances. For example, a user might submit a request to perform some compute-intensive task via a Web page implemented by a Windows Azure Web role. The Web role instance that receives this request can write a message into a queue describing the work to be done. A Worker role instance that's waiting on this queue can then read the message and carry out the task it specifies. Any results can be returned via another queue or handled in some other way.

Regardless of how data is stored—in blobs, tables, or queues—all information held in Windows Azure storage is replicated three times. This replication allows fault tolerance, since losing a copy isn't fatal. The system provides strong consistency, however, so an application that immediately reads data it has just written is guaranteed to get back what it just wrote.

Windows Azure storage can be accessed by a Windows Azure application, by an application running on-premises within some organization, or by an application running at a hoster. In all of these cases, all three Windows Azure storage styles use the conventions of REST to identify and expose data, as Figure below suggests. In other words, blobs, tables, and queues are all named using URIs and accessed via standard HTTP operations. A .NET client might use the ADO.NET Data Services libraries to do this, but it's not required—an application can also make raw HTTP calls.

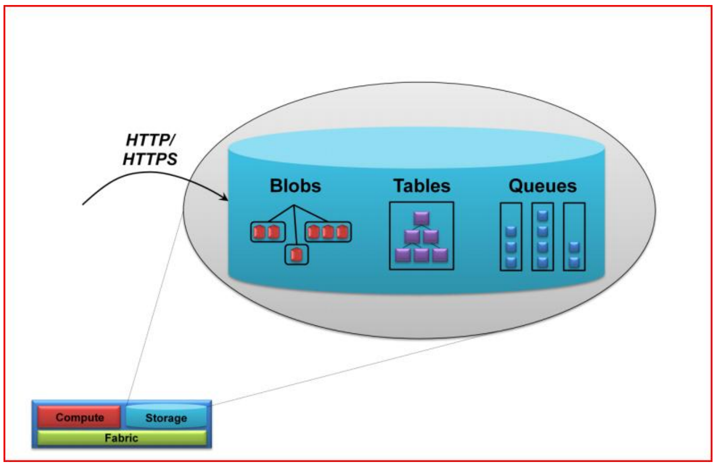


Fig: storage in windows azure

THE FABRIC:

All Windows Azure applications and all of the data in Windows Azure Storage live in some Microsoft data center. Within that data center, the set of machines dedicated to Windows Azure is organized into a fabric. Figure below shows how this looks.

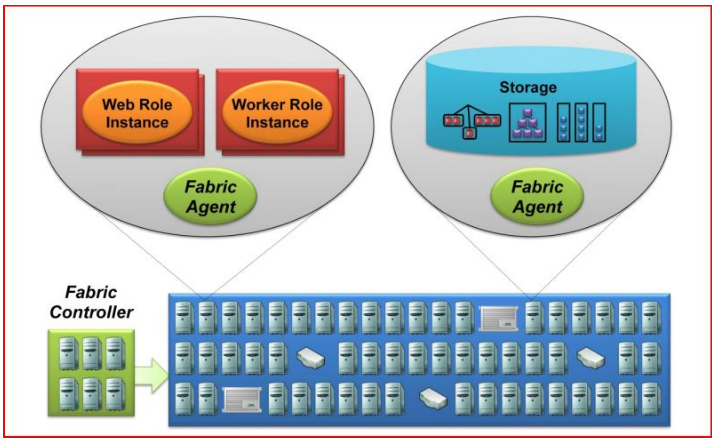


Fig: The fabric controller interacts with Windows Azure applications via the fabric agent.

As the figure shows, the Windows Azure Fabric consists of a (large) group of machines, all of which are managed by software called the fabric controller. The fabric controller is replicated across a group of five to seven machines, and it owns all of the resources in the fabric: computers, switches, load balancers, and more. Because it can communicate with a fabric agent on every computer, it’s also aware of every Windows Azure application in this fabric. (Interestingly, the fabric controller sees Windows Azure Storage as just another application, and so the details of data management and replication aren’t visible to the controller.)

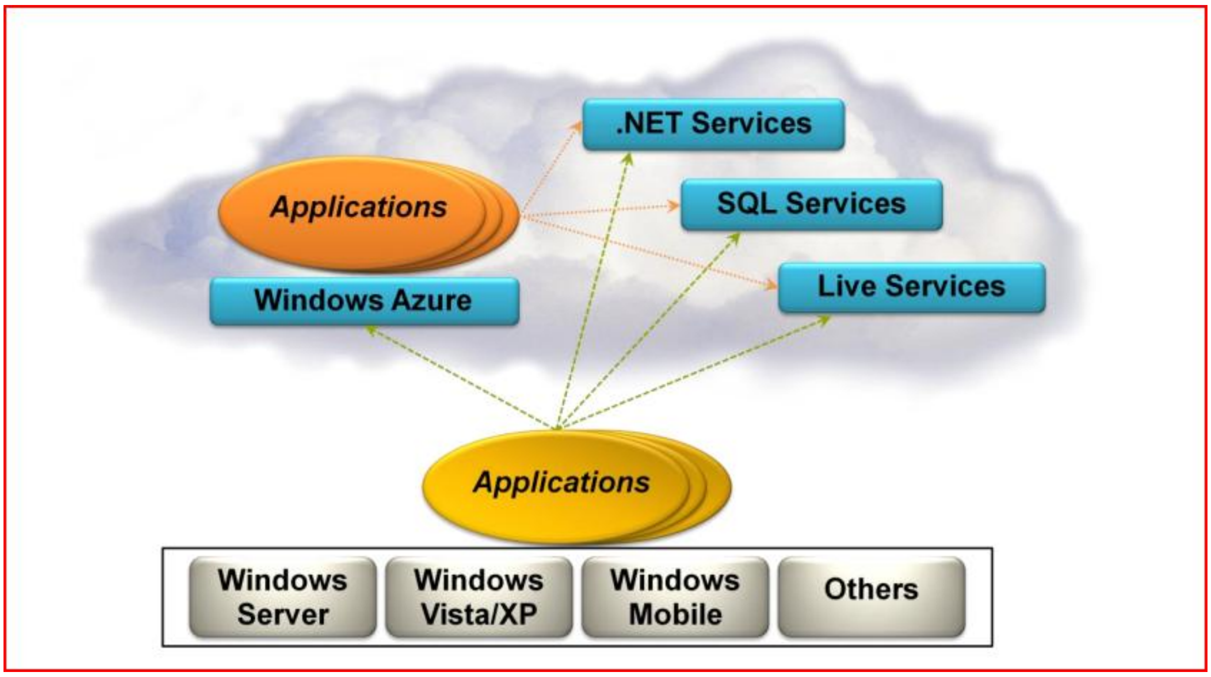
This broad knowledge lets the fabric controller do many useful things. It monitors all running applications, for example, giving it an up-to-the-minute picture of what’s happening in the fabric. It manages operating systems, taking care of things like patching the version of Windows Server 2008 that runs in Windows Azure VMs. It also decides where new applications should run, choosing physical servers to optimize hardware utilization.

To do this, the fabric controller depends on a configuration file that is uploaded with each Windows Azure application. This file provides an XML-based description of what the application needs: how many Web role instances, how many Worker role instances, and more. When the fabric controller receives this new application, it uses this configuration file to determine how many Web role and Worker role VMs to create.

Once it's created these VMs, the fabric controller then monitors each of them. If an application requires five Web role instances and one of them dies, for example, the fabric controller will automatically restart a new one. Similarly, if the machine a VM is running on dies, the fabric controller will start a new instance of the Web or Worker role in a new VM on another machine, resetting the load balancer as necessary to point to this new machine.

While this might change over time, the fabric controller in the Windows Azure CTP maintains a one-to-one relationship between a VM and a physical processor core. Because of this, performance is predictable—each application instance has its own dedicated processor core. It also means that there's no arbitrary limit on how long an application instance can execute. A Web role instance, for example, can take as long as it needs to handle a request from a user, while a Worker role instance can compute the value of pi to a million digits if necessary. Developers are free to do what they think is best.

Windows service platforms:



 Windows Azure: Provides a Windows-based environment for running applications and storing data on servers in Microsoft data centers.

 Microsoft .NET Services: Offers distributed infrastructure services to cloud-based and local applications.

 Microsoft SQL Services: Provides data services in the cloud based on SQL Server.

 Live Services: Through the Live Framework provides access to data from Microsoft's live applications and others. The Live Framework also allows synchronizing this data across desktops and devices, finding and downloading applications, and more.

Building and deploying your first windows azure application:

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

Prerequisites

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

Microsoft .NET Framework 3.5 SP1

Microsoft Visual Studio 2008 SP1 (or above)

Windows Azure Tools for Microsoft Visual Studio (November 2009)

SQL Server 2005 Express Edition (or above)

IIS 7 (with ASP.NET, WCF HTTP Activation)

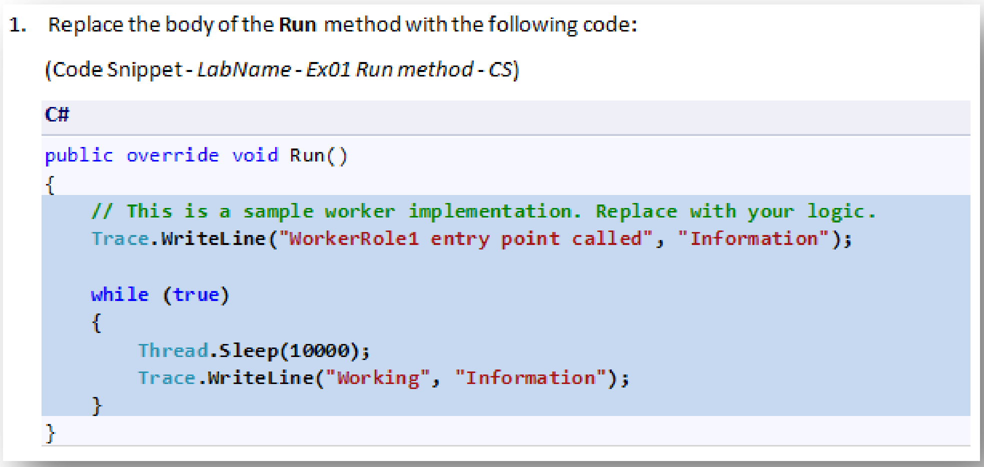
Setup

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. Run the **SetupLab.cmd** script located in the lab's **Source\Setup** folder to check dependencies and install any missing prerequisites.
2. Once you have verified every prerequisite, follow the instructions to install the code snippets.

Using the Code Snippets

With code snippets, you have all the code you need at your fingertips. The lab document will tell you exactly when you can use them. For example,

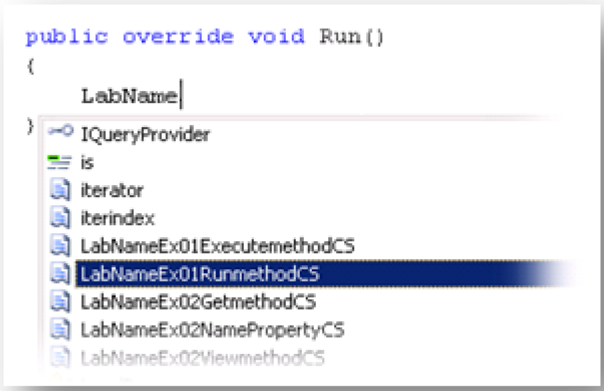

1. Replace the body of the **Run** method with the following code:

(Code Snippet - *LabName - Ex01 Run method - CS*)

C#

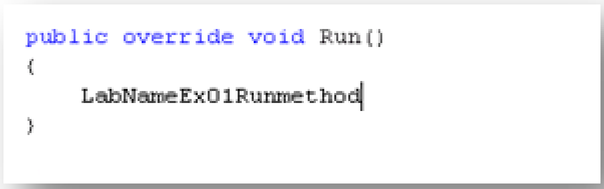
```
public override void Run()
{
    // This is a sample worker implementation. Replace with your logic.
    Trace.WriteLine("WorkerRole1 entry point called", "Information");

    while (true)
    {
        Thread.Sleep(10000);
        Trace.WriteLine("Working", "Information");
    }
}
```

To add this code snippet in Visual Studio, you simply place the cursor where you would like the code to be inserted, start typing the snippet name (without spaces or hyphens), in this case *LabNameEx01RunmethodCS*, watch as Intellisense picks up the snippet name, and then hit the TAB key twice once the snippet you want is selected. The code will be inserted at the cursor location.



```
public override void Run()
{
    LabName
}
```

Figure 1
Hit TAB to select the highlighted snippet.



```
public override void Run()
{
    LabNameEx01Runmethod
}
```

Figure 2
Hit TAB again and the snippet will expand

To insert a code snippet using the mouse rather than the keyboard, right-click where you want the code snippet to be inserted, select **Insert Snippet** followed by **My Code Snippets** and then pick the relevant snippet from the list.

To learn more about Visual Studio IntelliSense Code Snippets, including how to create your own, please see Creating and Using IntelliSense Code Snippets.

Exercises

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 2008 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.

Note: 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.

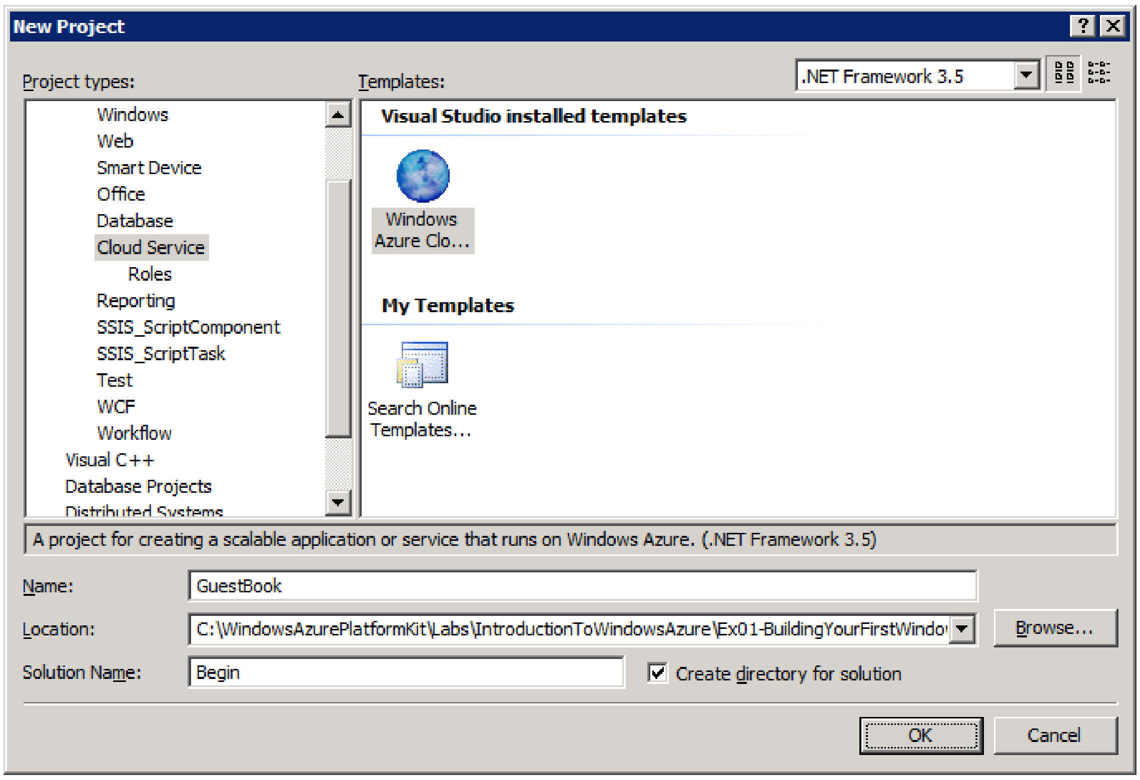
Task 1 – Creating the Visual Studio Project

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

1. Open Microsoft Visual Studio 2008 elevated as **Administrator**, from **Start** | **All Programs** | **Microsoft Visual Studio 2008**, right-click **Microsoft Visual Studio 2008** and choose **Run as Administrator**. If the **User Account Control** dialog appears, click **Continue**.
2. From the **File** menu, choose **New** and then **Project**.

3. In the **New Project** dialog, expand the language of your preference (Visual C# or Visual Basic) in the **Project types** list and select **Cloud Service**.

4. In the **Templates** list, select **Windows Azure Cloud Service**. Enter the name "**GuestBook**" and the solution name "**Begin**", then set the location to the folder for the language of your preference (Visual C# or Visual Basic) inside **Ex01-BuildingYourFirstWindowsAzureApp** in the **Source** folder of the lab. Ensure **Create directory for solution** is checked and click **OK** to create the project.



A project for creating a scalable application or service that runs on Windows Azure. (.NET Framework 3.5)

Name: GuestBook

Location: C:\WindowsAzurePlatformKit\Labs\IntroductionToWindowsAzure\Ex01-BuildingYourFirstWindo

Solution Name: Begin

☑ Create directory for solution

Figure 3

Creating a new Windows Azure Cloud Service project

5. In the **New Cloud Service 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.

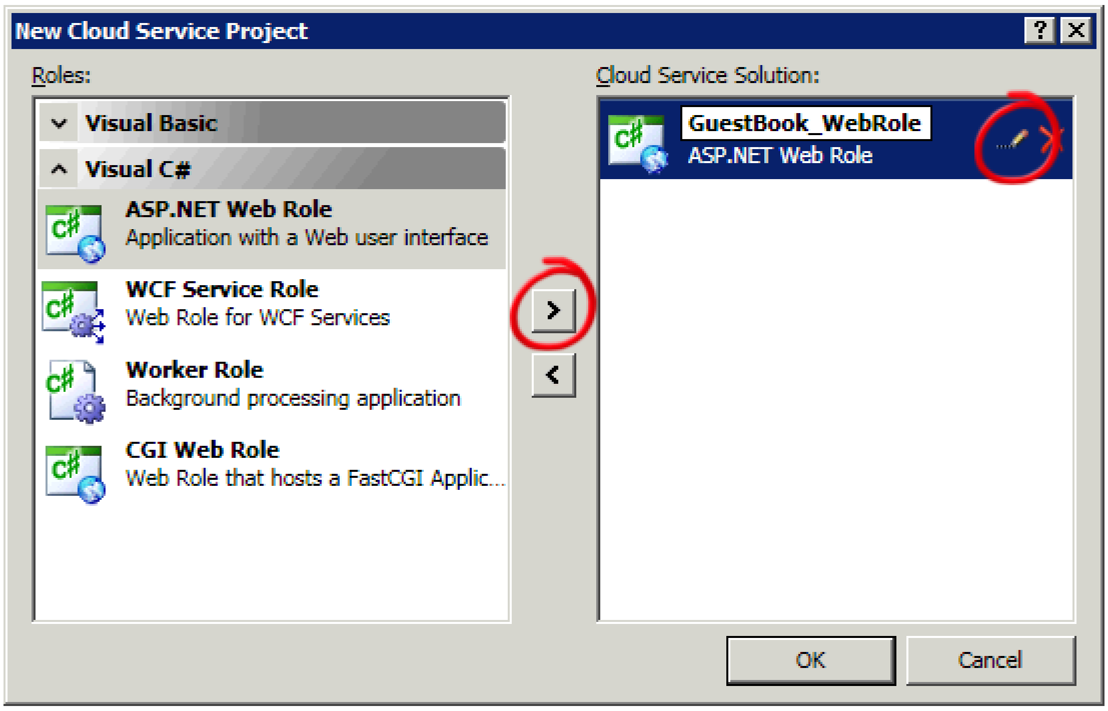

New Cloud Service Project
Roles:
Visual Basic
Visual C#
ASP.NET Web Role
Application with a Web user interface
WCF Service Role
Web Role for WCF Services
Worker Role
Background processing application
CGI Web Role
Web Role that hosts a FastCGI Applic...
Cloud Service Solution:
GuestBook_WebRole
ASP.NET Web Role
OK
Cancel

Figure 4

Assigning roles to the cloud service project (Visual C#)

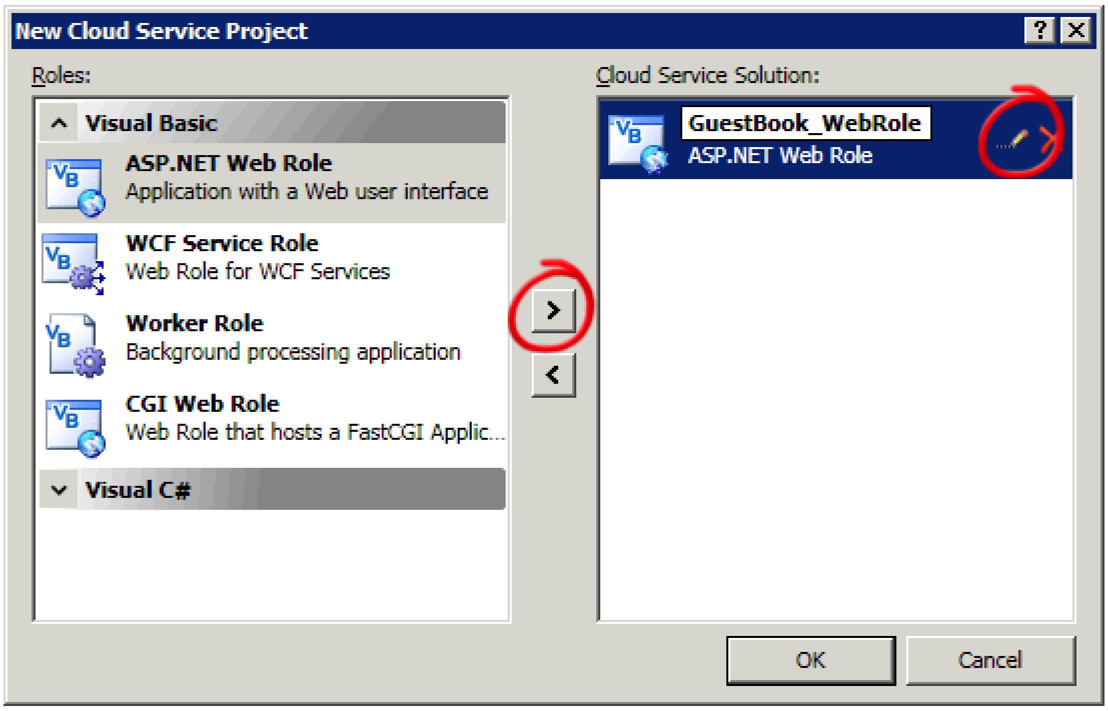

New Cloud Service Project
Roles:
Visual Basic
ASP.NET Web Role
Application with a Web user interface
WCF Service Role
Web Role for WCF Services
Worker Role
Background processing application
CGI Web Role
Web Role that hosts a FastCGI Applic...
Visual C#
Cloud Service Solution:
GuestBook_WebRole
ASP.NET Web Role
OK
Cancel

Figure 5

Assigning roles to the cloud service project (Visual Basic)

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

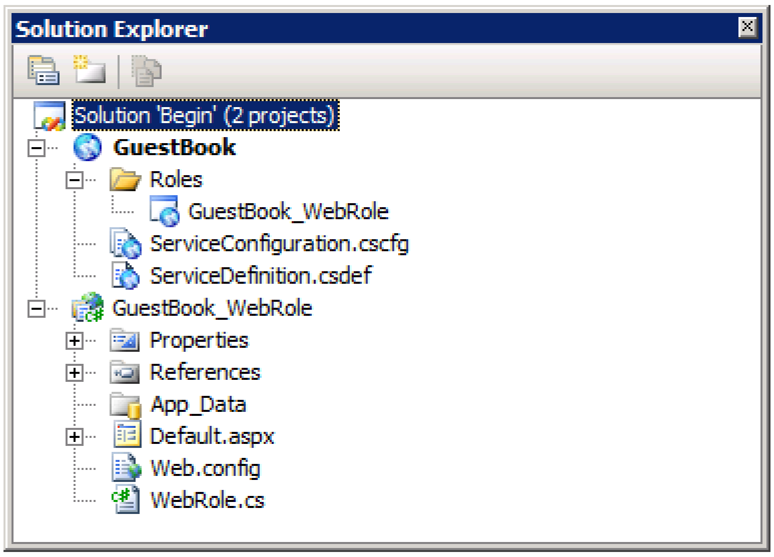


Figure 6

Solution Explorer showing the GuestBook application (C#)

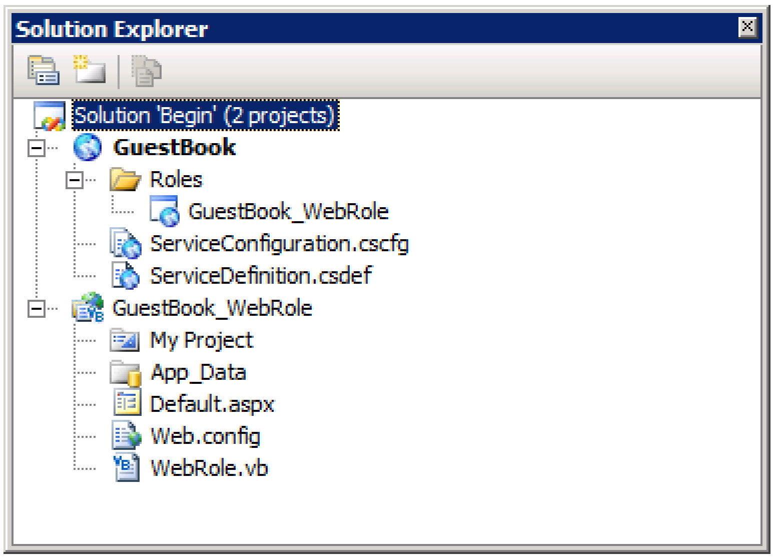


Figure 7

Solution Explorer showing the GuestBook application (Visual Basic)

Note: 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 the Table service 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. Every entity in the Table service also has a **Timestamp** system property, which allows the service to keep track of when an entity was last modified. This Timestamp 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, which you can use as the base class for your entities.

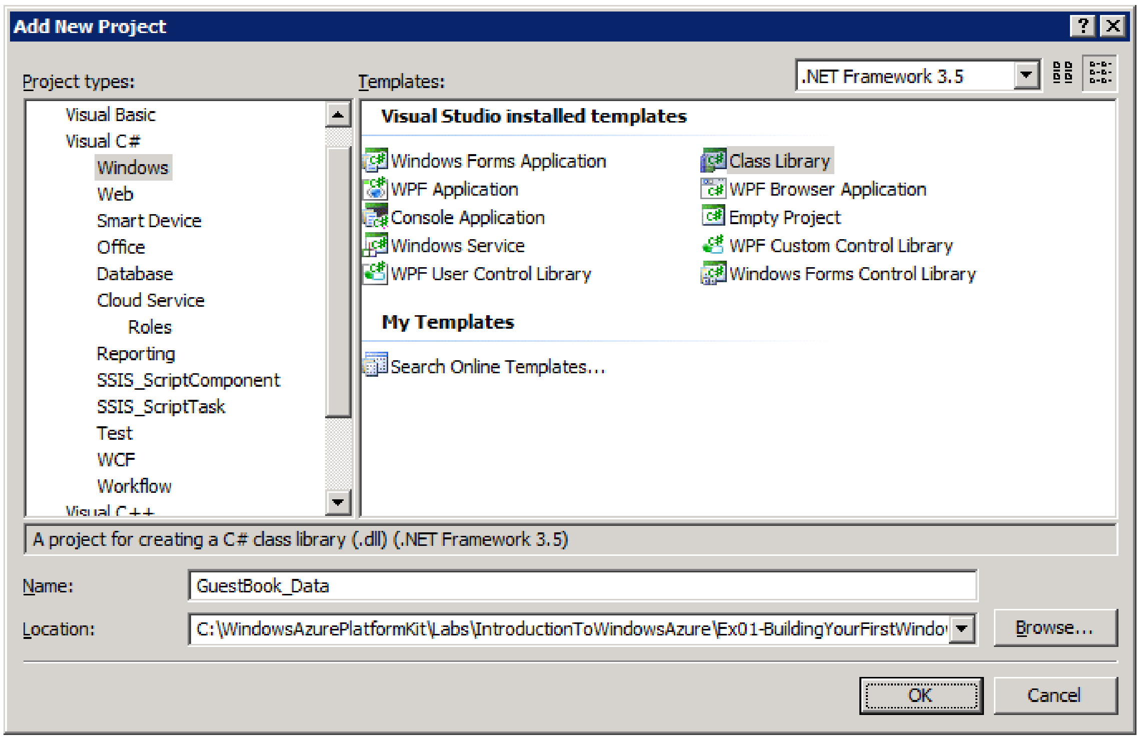
The Table service API is compliant with the REST API provided by ADO.NET Data allowing you to use the.NET Client Library for ADO.NET Data Services to work with data in the Table service using .NET objects.

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

To use the.NET Client Library for ADO.NET Data Services to access the data in table storage, you need to create a context class that derives from **TableServiceContext**, which itself derives from **DataServiceContext** in ADO.NET 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 ADO.NET Data Services to access the information in table storage. Finally, 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. In **Solution Explore**, right-click the **Begin** solution, point to **Add** and select **New Project.**
2. In the **New Project** dialog, expand the language of your preference (Visual C# or Visual Basic) in the **Project types** list, choose the **Windows** category, and select **Class Library** in the **Templates** list**.** Enter the name **"GuestBook_Data"** and click **OK**.



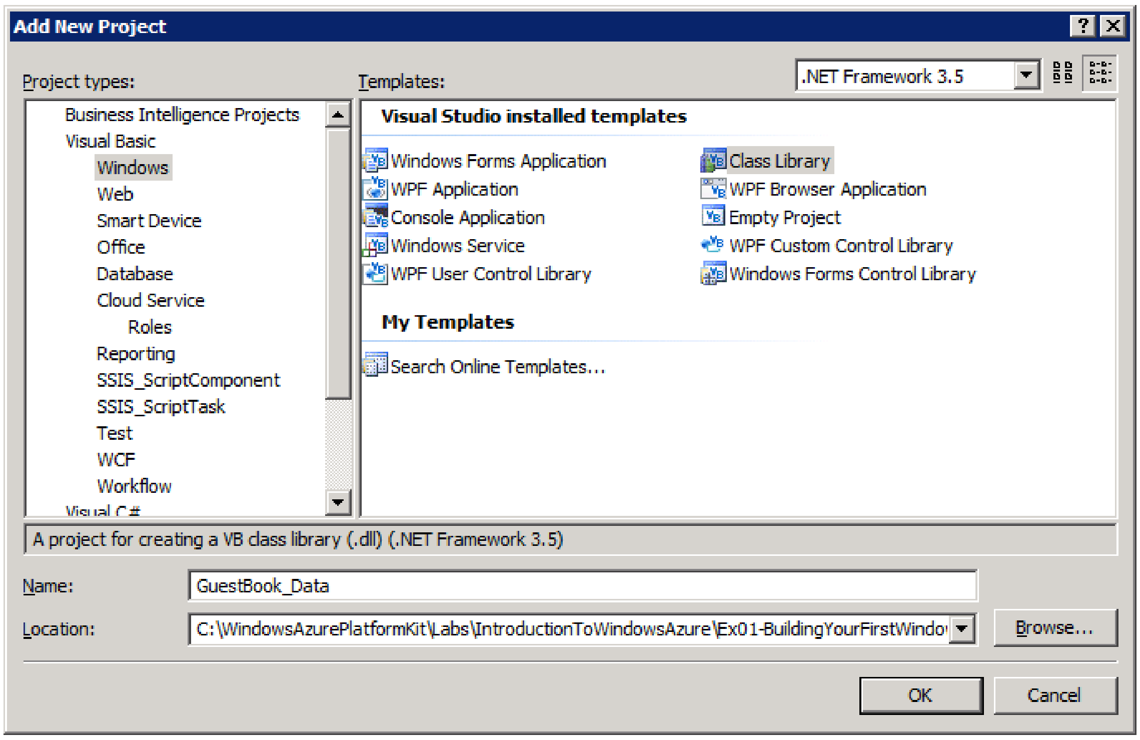
A project for creating a C# class library (.dll) (.NET Framework 3.5)

Name: GuestBook_Data

Location: C:\WindowsAzurePlatformKit\Labs\IntroductionToWindowsAzure\Ex01-BuildingYourFirstWindo

Figure 8

Creating a class library for GuestBook entities (C#)



A project for creating a VB class library (.dll) (.NET Framework 3.5)

Name: GuestBook_Data

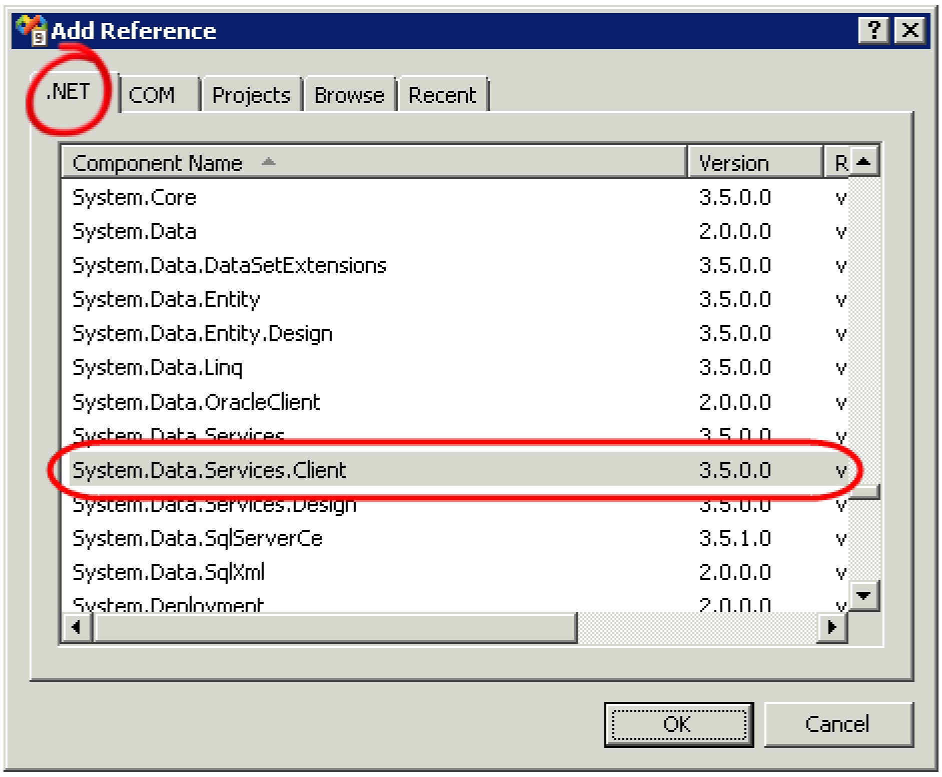
Location: C:\WindowsAzurePlatformKit\Labs\IntroductionToWindowsAzure\Ex01-BuildingYourFirstWindo

Figure 9

Creating a class library for the GuestBook entities (Visual Basic)

3. 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**.

4. Add a reference to the.NET Client Library for ADO.NET 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.Service.Client** component and click **OK**.

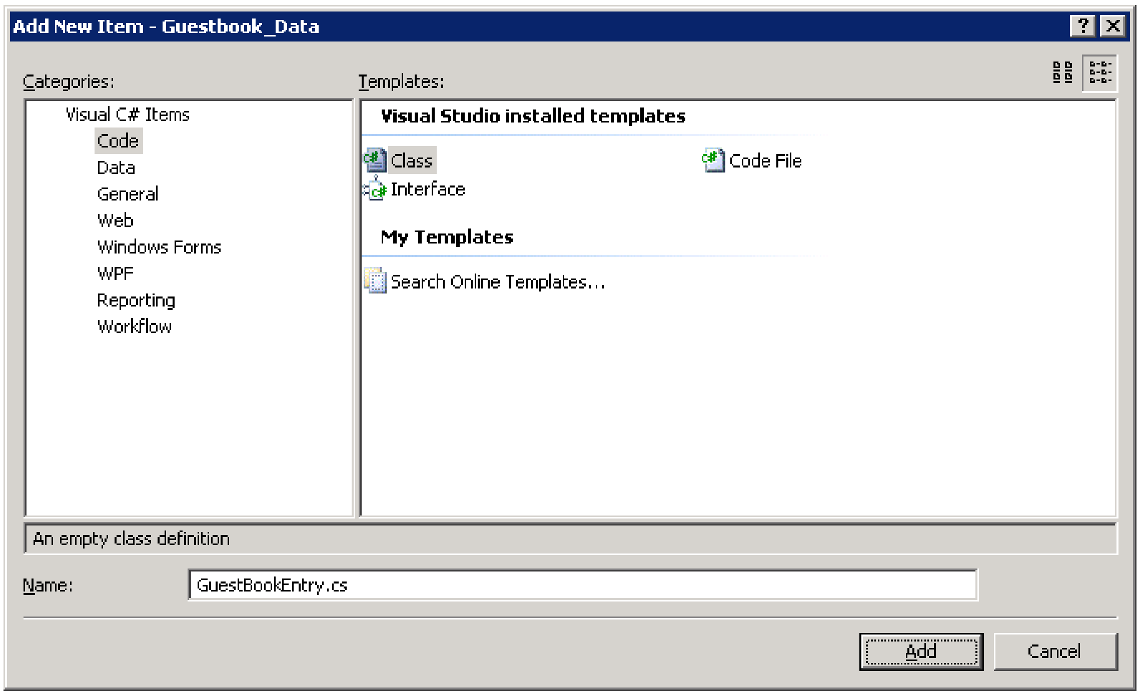


Component Name	Version	R
System.Core	3.5.0.0	v
System.Data	2.0.0.0	v
System.Data.DataSetExtensions	3.5.0.0	v
System.Data.Entity	3.5.0.0	v
System.Data.Entity.Design	3.5.0.0	v
System.Data.Linq	3.5.0.0	v
System.Data.OracleClient	2.0.0.0	v
System.Data.Services	3.5.0.0	v
System.Data.Services.Client	3.5.0.0	v
System.Data.Services.Design	3.5.0.0	v
System.Data.SqlServerCe	3.5.1.0	v
System.Data.SqlXml	2.0.0.0	v
System.Deployment	2.0.0.0	v

Figure 10

Adding a reference to the System.Data.Service.Client component

5. Repeat the previous step to add a reference to the Windows Azure storage client API assembly, this time choosing the **Microsoft.WindowsAzure.StorageClient** assembly instead.
6. 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** and select **Add New Item**. In the **Add New Item** dialog, choose the **Code** category and select **Class** in the **Templates** list. Enter the name **GuestBookEntry.cs** (for Visual C# projects) or **GuestBookEntry.vb** (for Visual Basic projects) and click **Add**.


Add New Item - Guestbook_Data

Categories:

- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Reporting
  - Workflow

Templates:

Visual Studio installed templates

Class Code File
Interface

My Templates

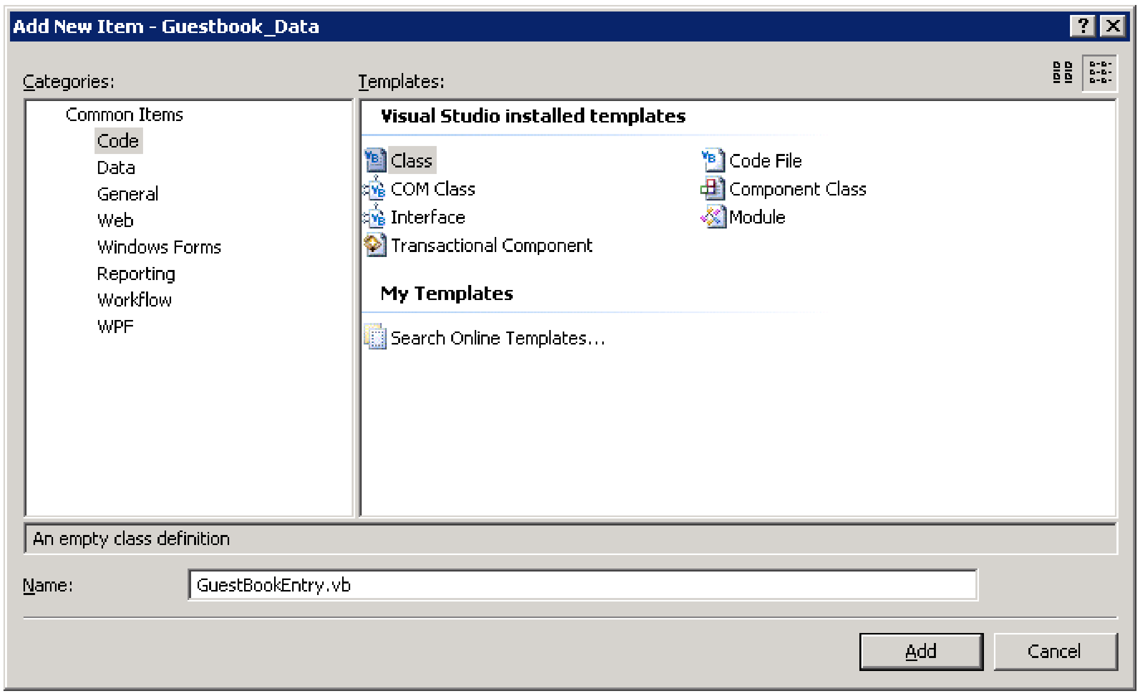
Search Online Templates...

An empty class definition

Name: GuestBookEntry.cs

Figure 11

Adding the GuestBookEntry class (C#)


Add New Item - Guestbook_Data

Categories:

- Common Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - Reporting
  - Workflow
  - WPF

Templates:

Visual Studio installed templates

Class Code File
COM Class Component Class
Interface Module
Transactional Component

My Templates

Search Online Templates...

An empty class definition

Name: GuestBookEntry.vb

Figure 12

Adding the GuestBookEntry class (Visual Basic)

7. Open the **GuestBookEntry.cs** file (for Visual C# projects) or **GuestBookEntry.vb** file (for Visual Basic projects). Add the following namespace declaration to import the types contained in the **Microsoft.WindowsAzure.StorageClient** namespace.

C#

```
using Microsoft.WindowsAzure.StorageClient;
```

Visual Basic

```
Imports Microsoft.WindowsAzure.StorageClient
```

8. Update the declaration of the **GuestBookEntry** class to make it public and derive from the **Microsoft.Samples.ServiceHosting.StorageClient.TableStorageEntity** class.

C#

```
public class GuestBookEntry :
   Microsoft.WindowsAzure.StorageClient.TableServiceEntity
{
}
```

Visual Basic

```
Public Class GuestBookEntry
    Inherits Microsoft.WindowsAzure.StorageClient.TableServiceEntity
```

Note: 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.

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

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntry constructor – C#*)

C#

```
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());
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntry constructor – Visual Basic*)

Visual Basic

```
Public Sub New()
  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())
End Sub
```

Note: 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.

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

    (Code Snippet – *Introduction to Windows Azure - Ex01 Table Schema Properties – C#*)

C#

```
public string Message { get; set; }
public string GuestName { get; set; }
public string PhotoUrl { get; set; }
public string ThumbnailUrl { get; set; }
```

(Code Snippet – *Introduction to Windows Azure - Ex01 Table Schema Properties – Visual Basic*)

Visual Basic

```
Private privateMessage As String
Private privateGuestName As String
Private privatePhotoUrl As String
Private privateThumbnailUrl As String

Public Property Message() As String
  Get
    Return privateMessage
  End Get
  Set(ByVal value As String)
    privateMessage = value
  End Set
End Property

Public Property GuestName() As String
  Get
    Return privateGuestName
  End Get
  Set(ByVal value As String)
    privateGuestName = value
  End Set
End Property

Public Property PhotoUrl() As String
  Get
    Return privatePhotoUrl
  End Get
  Set(ByVal value As String)
    privatePhotoUrl = value
  End Set
End Property

Public Property ThumbnailUrl() As String
  Get
    Return privateThumbnailUrl
  End Get
  Set(ByVal value As String)
    privateThumbnailUrl = value
  End Set
End Property
```

11. Save the **GuestBookEntry.cs** file (for Visual C# projects) or **GuestBookEntry.vb** file (for Visual Basic projects).
12. Next, you need to create the context class required to access the *GuestBook* table using ADO.NET 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** (for C# projects) or **GuestBookDataContext.vb** (for Visual Basic projects) and click **Add**.
13. In the new class file, add the following namespace declarations to import the types contained in the **Microsoft.WindowsAzure** and **Microsoft.WindowsAzure.StorageClient** namespaces.

C#

```
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;
```

Visual Basic

```
Imports Microsoft.WindowsAzure
Imports Microsoft.WindowsAzure.StorageClient
```

14. Update the declaration of the new class to make it public, inherit the **TableServiceContext** class and include a default constructor to initialize its base class with storage account information.

    (Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookDataContext Class – C#*)

    C#

    ```
    public class GuestBookDataContext : TableServiceContext
    {
      public GuestBookDataContext(string baseAddress, StorageCredentials 
    credentials)
        : base(baseAddress, credentials)
      { }
    }
    ```

    (Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookDataContext Class – Visual Basic*)

    Visual Basic

    ```
    Public Class GuestBookDataContext
      Inherits TableServiceContext

      Public Sub New(ByVal baseAddress As String, ByVal credentials As 
    StorageCredentials)
        MyBase.New(baseAddress, credentials)
      End Sub
    End Class
    ```

    Note: You can find the **TableServiceContext** class in the storage client API. This class derives from **DataServiceContext** in ADO.NET Data Services and manages the credentials required to access your storage account as well as providing support for a retry policy for operations.

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

    (Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntry Property – C#*)

    C#

    ```
    public class GuestBookDataContext : TableServiceContext
    {
      ...
      public IQueryable<GuestBookEntry> GuestBookEntry
      {
        get
    ```

```
    {
      return this.CreateQuery<GuestBookEntry>("GuestBookEntry");
    }
  }
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntry Property – Visual Basic*)

Visual Basic

```
Public Class GuestBookDataContext
  Inherits TableServiceContext
  ...
  Public ReadOnly Property GuestBookEntry() As IQueryable(Of
GuestBookEntry)
    Get
      Return Me.CreateQuery(Of GuestBookEntry)("GuestBookEntry")
    End Get
  End Property

End Class
```

Note: 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.

16. Finally, you need to implement an object that can be bound to data controls in ASP.NET. In **Solution Explorer**, right-click **GuestBook_Data** and click **Add New Item**. In the **Add New Item** dialog, select **Code** in the categories list and choose the **Class** template. Enter the name **GuestBookEntryDataSource.cs** (for Visual C# projects) or **GuestBookEntryDataSource.vb** (for Visual Basic projects) and click **Add**.
17. In the new class file, add the following namespace declarations to import the types contained in the **Microsoft.WindowsAzure** and **Microsoft.WindowsAzure.StorageClient** namespaces.

C#

```
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;
```

Visual Basic

```
Imports Microsoft.WindowsAzure
Imports Microsoft.WindowsAzure.StorageClient
```

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

    (Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource Fields – C#*)

    C#

    ```
    public class GuestBookEntryDataSource
    {
      private static CloudStorageAccount storageAccount;
      private GuestBookDataContext context;
    }
    ```

    (Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource Fields – Visual Basic*)

    Visual Basic

    ```
    Public Class GuestBookEntryDataSource
      Private Shared storageAccount As CloudStorageAccount
      Private context As GuestBookDataContext
    End Class
    ```

19. Now, add a static (Shared in Visual Basic) 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 - Ex01 GuestBookEntryDataSource Static Constructor – C#*)

    C#

    ```
    public class GuestBookEntryDataSource
    {
      private static CloudStorageAccount storageAccount;
      private GuestBookDataContext context;

      static GuestBookEntryDataSource()
      {
        storageAccount = 
    CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

        CloudTableClient.CreateTablesFromModel(
            typeof(GuestBookDataContext),
            storageAccount.TableEndpoint.AbsoluteUri,
            storageAccount.Credentials);
      }
    }
    ```

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource Shared Constructor – Visual Basic*)

Visual Basic

```
Public Class GuestBookEntryDataSource
  Private Shared storageAccount As CloudStorageAccount
  Private context As GuestBookDataContext

  Shared Sub New()
    storageAccount =
CloudStorageAccount.FromConfigurationSetting("DataConnectionString")

    CloudTableClient.CreateTablesFromModel(GetType(GuestBookDataContext),
storageAccount.TableEndpoint.AbsoluteUri, storageAccount.Credentials)
  End Sub
End Class
```

Note: 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.

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

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource Constructor – C#*)

C#

```
public GuestBookEntryDataSource()
{
  this.context = new
GuestBookDataContext(storageAccount.TableEndpoint.AbsoluteUri,
storageAccount.Credentials);
  this.context.RetryPolicy = RetryPolicies.Retry(3,
TimeSpan.FromSeconds(1));
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource Constructor – Visual Basic*)

Visual Basic

```
Public Sub New()
  Me.context = New
GuestBookDataContext(storageAccount.TableEndpoint.AbsoluteUri,
storageAccount.Credentials)
```

```
  Me.context.RetryPolicy = RetryPolicies.Retry(3, TimeSpan.FromSeconds(1))
End Sub
```

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

    (Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource Select – C#*)

    C#

```
public IEnumerable<GuestBookEntry> Select()
{
  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 - Ex01 GuestBookEntryDataSource Select – Visual Basic*)

Visual Basic

```
Public Function [Select]() 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
```

Note: The **Select** method retrieves today's guest book entries by constructing a LINQ statement using the current date as the partition key. The web role uses this method to bind to a data grid and display the guest book.

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

    (Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource AddGuestBookEntry – C#*)

    C#

```
public void AddGuestBookEntry(GuestBookEntry newItem)
{
  this.context.AddObject("GuestBookEntry", newItem);
  this.context.SaveChanges();
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource AddGuestBookEntry – Visual Basic*)

Visual Basic

```
Public Sub AddGuestBookEntry(ByVal newItem As GuestBookEntry)
  Me.context.AddObject("GuestBookEntry", newItem)
  Me.context.SaveChanges()
End Sub
```

Note: This method adds a new **GuestBookEntry** object to the data context and then calls **SaveChanges** to write the entity to storage.

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

(Code Snippet – *Introduction to Windows Azure - Ex01 GuestBookEntryDataSource UpdateImageThumbnail – C#*)

C#

```
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 - Ex01 GuestBookEntryDataSource UpdateImageThumbnail – Visual Basic*)

Visual Basic

```
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
```

Note: 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.

24. Save the **GuestBookEntryDataSource.cs** file (for Visual C# projects) or **GuestBookEntryDataSource.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 the 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. Finally, you configure the storage account used by the Web role.

1. Add a reference to the storage client API assembly. In **Solution Explorer**, right-click the **GuestBook_WebRole** project node and point to **Add Reference**, switch to the **.NET** Tab, select the **Microsoft.WindowsAzure.StorageClient** assembly and click **OK**.
2. Now, add a reference to the **GuestBook_Data** project. Once again, in **Solution Explorer**, right-click the **GuestBook_WebRole** project node and select **Add Reference**, switch to the **Project** tab this time, select the **GuestBook_Data** project and click **OK**.
3. 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**.
4. 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\Ex01-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**.

   Note: 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.

5. 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**.
6. In the code-behind file, insert the following namespace declarations.

   (Code Snippet – *Introduction to Windows Azure - Ex01 Web Role Namespace Declarations – C#*)

C#

```
using System.Net;
using GuestBook_Data;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.ServiceRuntime;
using Microsoft.WindowsAzure.StorageClient;
```

(Code Snippet – *Introduction to Windows Azure - Ex01 Web Role Namespace Declarations – Visual Basic*)

Visual Basic

```
Imports System.Net
Imports GuestBook_Data
Imports Microsoft.WindowsAzure
Imports Microsoft.WindowsAzure.ServiceRuntime
Imports Microsoft.WindowsAzure.StorageClient
```

7. Declare the following member fields in the **_Default** class.

(Code Snippet – *Introduction to Windows Azure - Ex01 Web Role Member Fields – C#*)

C#

```
private static bool storageInitialized = false;
private static object gate = new Object();
private static CloudBlobClient blobStorage;
```

(Code Snippet – *Introduction to Windows Azure - Ex01 Web Role Member Fields – Visual Basic*)

Visual Basic

```
Private Shared storageInitialized As Boolean = False
Private Shared gate As New Object()
Private Shared blobStorage As CloudBlobClient
```

8. Locate the **SignButton_Click** event handler in the code-behind file and insert the following code.

(Code Snippet – *Introduction to Windows Azure - Ex01 SignButton_Click – C#*)

C#

```
protected void SignButton_Click(object sender, EventArgs e)
{
  if (FileUpload1.HasFile)
  {
    InitializeStorage();

    // upload the image to blob storage
    CloudBlobContainer container =
blobStorage.GetContainerReference("guestbookpics");
    string uniqueBlobName = string.Format("image_{0}.jpg",
Guid.NewGuid().ToString());
    CloudBlockBlob blob = container.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() };
    GuestBookEntryDataSource ds = new GuestBookEntryDataSource();
    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 - Ex01 SignButton_Click – Visual Basic*)

Visual Basic

```
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 container As CloudBlobContainer =
blobStorage.GetContainerReference("guestbookpics")
    Dim uniqueBlobName As String = String.Format("image_{0}.jpg",
Guid.NewGuid().ToString())
    Dim blob As CloudBlockBlob =
container.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 GuestBookEntryDataSource()
    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
```

Note: 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, which allows 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 **GuestBookEntryDataSource** class to save the entry to table storage using the .NET Client Library for ADO.NET Data Services.

Finally, it data binds the guest book entries list to refresh its contents.

9. Update the body of the **Timer1_Tick** method with the code shown (highlighted) below.

(Code Snippet – *Introduction to Windows Azure - Ex01 Timer1_Tick– C#*)

C#

```
protected void Timer1_Tick(object sender, EventArgs e)
{
  DataList1.DataBind();
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 Timer1_Tick– Visual Basic*)

Visual Basic

```
Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As EventArgs)
Handles Timer1.Tick
  DataList1.DataBind()
End Sub
```

Note: A timer on the page periodically refreshes the contents of the guest book entries list.

10. Locate the **Page_Load** event handler and update its body with the following (highlighted) code to enable the page refresh timer.

    (Code Snippet – Introduction to Windows Azure - Ex01 Page_Load – C#)

C#

```
protected void Page_Load(object sender, EventArgs e)
{
  if (!Page.IsPostBack)
  {
    Timer1.Enabled = true;
  }
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 Page_Load – Visual Basic*)

Visual Basic

```
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
```

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

    (Code Snippet – *Introduction to Windows Azure - Ex01 InitializeStorage –C#*)

C#

```
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;
  }
}
```

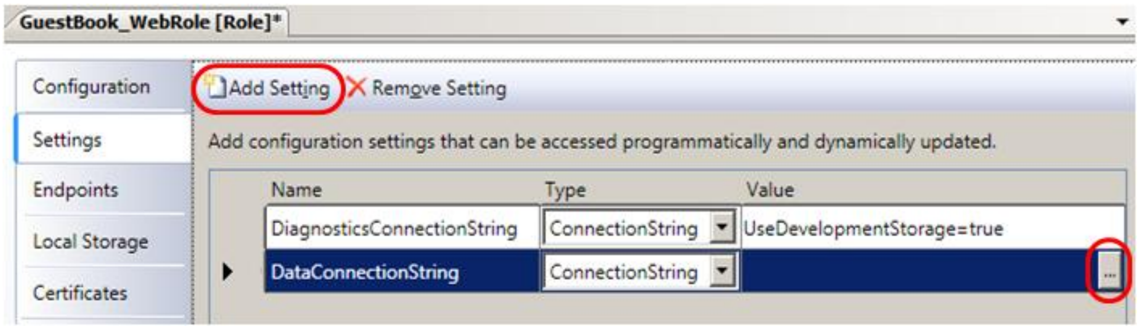
(Code Snippet – *Introduction to Windows Azure - Ex01 InitializeStorage – Visual Basic*)

Visual Basic

```
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
```

Note: 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.

12. 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.

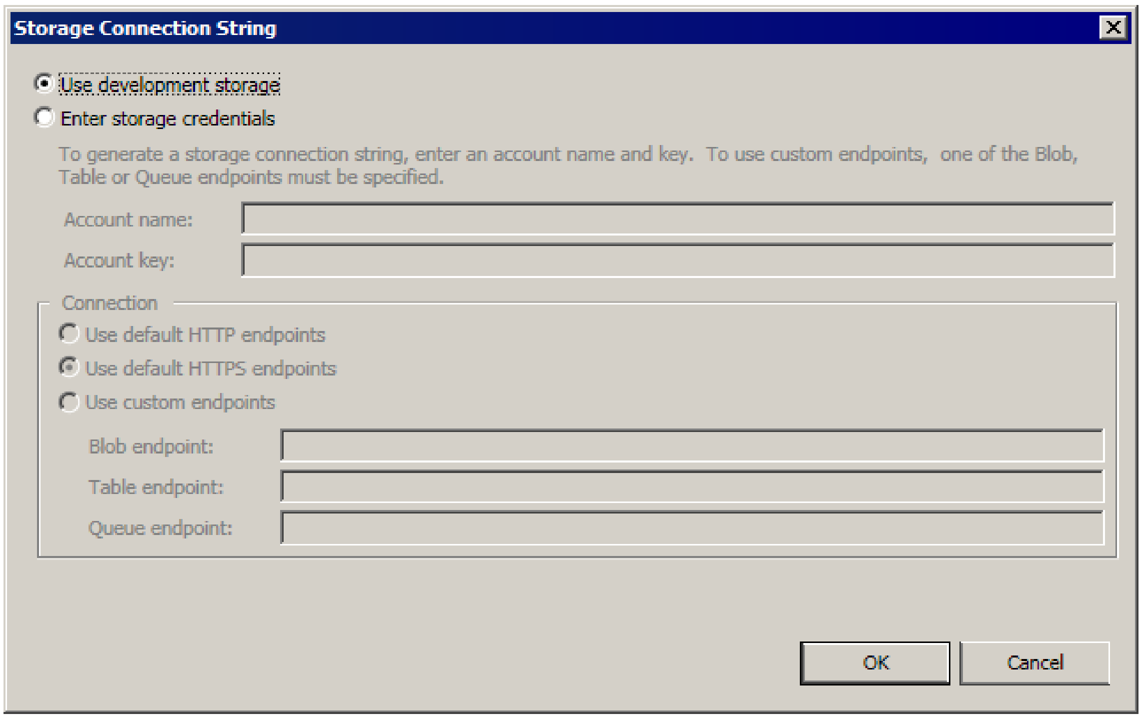


Add configuration settings that can be accessed programmatically and dynamically updated.

Name	Type	Value
DiagnosticsConnectionString	ConnectionString	UseDevelopmentStorage=true
DataConnectionString	ConnectionString	

Figure 13
Configuring the storage account settings

13. In the **Storage Connection String** dialog, choose the **Use development storage** option and click **OK**.


Storage Connection String

(●) Use development storage
( ) Enter storage credentials

To generate a storage connection string, enter an account name and key. To use custom endpoints, one of the Blob, Table or Queue endpoints must be specified.

Account name:
Account key:

Connection
( ) Use default HTTP endpoints
(●) Use default HTTPS endpoints
( ) Use custom endpoints

Blob endpoint:
Table endpoint:
Queue endpoint:

Figure 14

Setting "Use development storage" on Storage Connection String dialog

Note: A storage account is a unique endpoint for the Windows Azure blob, queue, and table services. You must create a storage account in the Developer Portal to use these services. In this exercise, you use development storage, 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 development storage.

To use development storage, 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;Ac
countKey=YourAccountKey" />
```

14. Press **CTRL + S** to save changes to the role configuration.
15. Finally, you need to set up the environment for the configuration publisher. In the **GuestBook_WebRole** project, open the **WebRole.cs** file (for Visual C# projects) or the **WebRole.vb** file (for Visual Basic projects) and insert the following code into the **OnStart** method immediately after the line that sets up a handler for the **RoleEnvironmentChanging** event.

C#

```
public override bool OnStart()
{
  DiagnosticMonitor.Start("DiagnosticsConnectionString");

  // Restart the role upon all configuration changes
  // Note: To customize the handling of configuration changes,
  // remove this line and register custom event handlers instead.
  // See the MSDN topic on “Managing Configuration Changes” for further
details
  // (http://go.microsoft.com/fwlink/?LinkId=166357).
  RoleEnvironment.Changing += RoleEnvironmentChanging;

Microsoft.WindowsAzure.CloudStorageAccount.SetConfigurationSettingPublisher
(( configName, configSetter) =>
  {

configSetter(Microsoft.WindowsAzure.ServiceRuntime.RoleEnvironment.GetConfi
gurationSettingValue(configName));
  });

  return base.OnStart();
}
```

Visual Basic

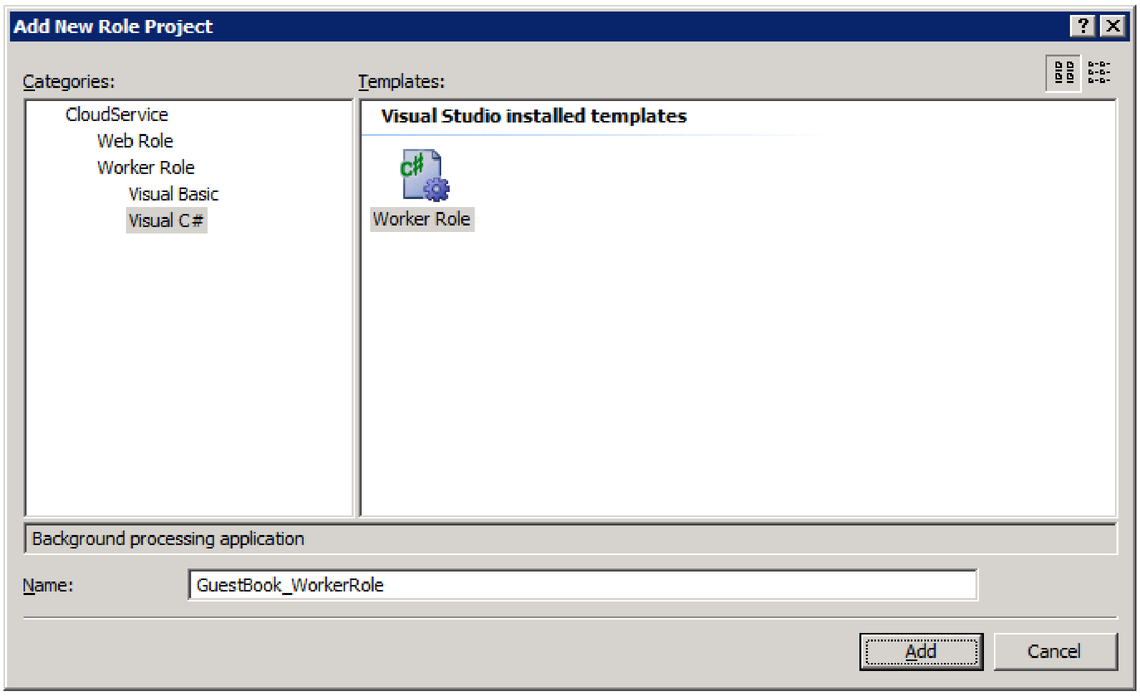
```
Public Overrides Function OnStart() As Boolean

  DiagnosticMonitor.Start("DiagnosticsConnectionString")

  '   Restart the role upon all configuration changes
  '   Note: To customize the handling of configuration changes,
  '   remove this line and register custom event handlers instead.
  '   See the MSDN topic on “Managing Configuration Changes” for further
details
  '   (http://go.microsoft.com/fwlink/?LinkId=166357).
  AddHandler RoleEnvironment.Changing, AddressOf RoleEnvironmentChanging


Microsoft.WindowsAzure.CloudStorageAccount.SetConfigurationSettingPublisher
( Function(configName, configSetter)
configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))

  Return MyBase.OnStart()
End Function
```

Task 4 – Creating a Worker Role to Process Images in the Background

A worker role runs in the background to provide services or execute time related tasks like a service process.

In this task, you create a worker role to read work items posted to a queue by the web role front-end. The worker role extracts the information about the guest book entry from the message and then retrieves the entry from table storage. It then fetches the associated image from blob storage and creates a thumbnail, which it also stores as a blob. Finally, it updates the entry to include the URL of the generated thumbnail.

1. In **Solution Explorer**, right-click the **Roles** node in the **GuestBook** project, point to **Add** and select **New Worker Role Project**.
2. In the **Add New Role Project** dialog, select the **Worker Role** category and choose the **Worker Role** template for the language of your choice (Visual C# or Visual Basic). Set the name of the worker role to **GuestBook_WorkerRole** and click **Add**.


Add New Role Project

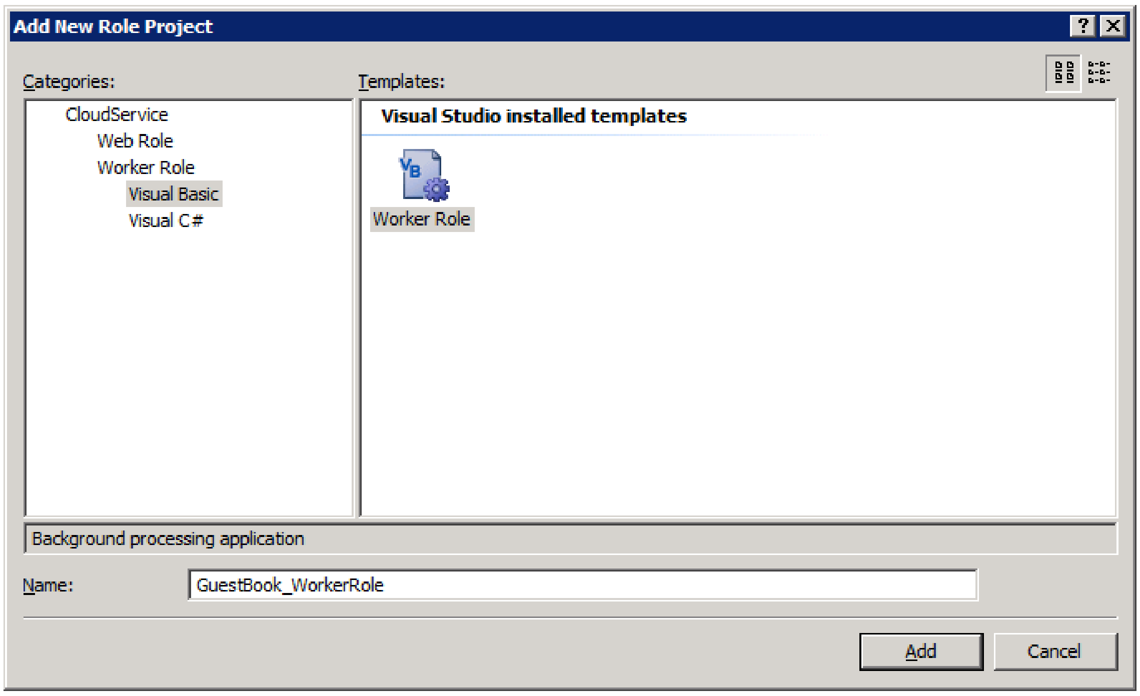
Categories: CloudService / Web Role / Worker Role / Visual Basic / Visual C#

Templates: Visual Studio installed templates — Worker Role

Background processing application

Name: GuestBook_WorkerRole

Figure 15
Adding a worker role to the GuestBook application (Visual C#)


Add New Role Project

Categories:
CloudService
Web Role
Worker Role
Visual Basic
Visual C#

Templates:
Visual Studio installed templates
Worker Role

Background processing application

Name: GuestBook_WorkerRole

Figure 16

Adding a worker role to the GuestBook application (Visual Basic)

3. In the new worker role project, add a reference to the data model project. In **Solution Explorer**, right-click the **GuestBook_WorkerRole** project and select **Add Reference**, switch to the **Projects** tab, select the **GuestBook_Data** project and click **OK**.
4. Next, add a reference to the **System.Drawing** assembly. To do this, in **Solution Explorer**, right-click the **GuestBook_WorkerRole** project, select **Add Reference**, switch to the **.NET** tab, select the **System.Drawing** component and click **OK**.
5. Repeat the procedure in the previous step to add a reference to the storage client API assembly, this time choosing the **Microsoft.WindowsAzure.StorageClient** component instead.
6. In the **GuestBook_WorkerRole** project, open the **WorkerRole.cs** file (for Visual C# projects) or **WorkerRole.vb** file (for Visual Basic projects).
7. In the worker role file, add the followings namespace declarations.

(Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole Namespaces – C#*)

C#

```
using System.Drawing;
using System.IO;
using GuestBook_Data;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;
```

(Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole Namespaces – Visual Basic*)

Visual Basic

```
Imports System.Drawing
Imports System.IO
Imports GuestBook_Data
Imports Microsoft.WindowsAzure
Imports Microsoft.WindowsAzure.StorageClient
```

8. Add member fields to the **WorkerRole** class for the blob container and the queue, as shown below.

   (Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole Fields – C#*)

   C#

```
private CloudQueue queue;
private CloudBlobContainer container;
```

   (Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole Fields – Visual Basic*)

   Visual Basic

```
Private Shared queue As CloudQueue
Private Shared container As CloudBlobContainer
```

9. Replace the body of the **OnStart** method with the following code.

   (Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole OnStart – C#*)

   C#

```
public override bool OnStart()
{
  DiagnosticMonitor.Start("DiagnosticsConnectionString");

  // Restart the role upon all configuration changes
  RoleEnvironment.Changing += RoleEnvironmentChanging;

  // read storage account configuration settings
  CloudStorageAccount.SetConfigurationSettingPublisher((configName,
configSetter) =>
  {
    configSetter(RoleEnvironment.GetConfigurationSettingValue(configName));
  });
  var storageAccount =
CloudStorageAccount.FromConfigurationSetting("DataConnectionString");

  // initialize blob storage
  CloudBlobClient blobStorage = storageAccount.CreateCloudBlobClient();
  container = blobStorage.GetContainerReference("guestbookpics");
  // initialize queue storage
  CloudQueueClient queueStorage = storageAccount.CreateCloudQueueClient();
  queue = queueStorage.GetQueueReference("guestthumbs");

  Trace.TraceInformation("Creating container and queue...");

  bool storageInitialized = false;
  while (!storageInitialized)
  {
    try
    {
      // create the blob container and allow public access
      container.CreateIfNotExist();
      var permissions = container.GetPermissions();
      permissions.PublicAccess = BlobContainerPublicAccessType.Container;
      container.SetPermissions(permissions);

      // create the message queue
      queue.CreateIfNotExist();
      storageInitialized = true;
    }
    catch (StorageClientException e)
    {
      if (e.ErrorCode == StorageErrorCode.TransportError)
      {
        Trace.TraceError("Storage services initialization failure. "
          + "Check your storage account configuration settings. If running
locally, "
          + "ensure that the Development Storage service is running.
Message: '{0}'", e.Message);
        System.Threading.Thread.Sleep(5000);
      }
      else
      {
        throw;
      }
    }
  }

  return base.OnStart();
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole OnStart – Visual Basic*)

Visual Basic

```
Public Overrides Function OnStart() As Boolean

  DiagnosticMonitor.Start("DiagnosticsConnectionString")

  ' Restart the role upon all configuration changes
  AddHandler RoleEnvironment.Changing, AddressOf RoleEnvironmentChanging

  ' read storage account configuration settings
  CloudStorageAccount.SetConfigurationSettingPublisher(Function(configName,
configSetter)
configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))
  Dim storageAccount =
CloudStorageAccount.FromConfigurationSetting("DataConnectionString")

  ' initialize blob storage
  Dim blobStorage = storageAccount.CreateCloudBlobClient()
  container = blobStorage.GetContainerReference("guestbookpics")

  ' initialize queue storage
  Dim queueStorage = storageAccount.CreateCloudQueueClient()
  queue = queueStorage.GetQueueReference("guestthumbs")

  Trace.TraceInformation("Creating container and queue...")

  Dim storageInitialized = False
  Do While (Not storageInitialized)
    Try
      ' create the blob container and allow public access
      container.CreateIfNotExist()
      Dim permissions = container.GetPermissions()
      permissions.PublicAccess = BlobContainerPublicAccessType.Container
      container.SetPermissions(permissions)

      ' create the message queue
      queue.CreateIfNotExist()
      storageInitialized = True
      Catch e As StorageClientException
      If (e.ErrorCode = StorageErrorCode.TransportError) Then
        Trace.TraceError("Storage services initialization failure. " _
          & "Check your storage account configuration settings. If running
locally, " _
          & "ensure that the Development Storage service is running.
Message: '{0}'", e.Message)
        System.Threading.Thread.Sleep(5000)
      Else
        Throw
      End If
    End Try
  Loop

  Return MyBase.OnStart()

End Function
```

10. Replace the body of the **Run** method with the code shown below.

    (Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole Run – C#*)

C#

```
public override void Run()
{
  Trace.TraceInformation("Listening for queue messages...");

  while (true)
  {
    try
    {
      // retrieve a new message from the queue
      CloudQueueMessage msg = queue.GetMessage();
      if (msg != null)
      {
        // parse message retrieved from queue
        var messageParts = msg.AsString.Split(new char[] { ',' });
        var uri = messageParts[0];
        var partitionKey = messageParts[1];
        var rowkey = messageParts[2];
        Trace.TraceInformation("Processing image in blob '{0}'.", uri);

        // download original image from blob storage
        CloudBlockBlob imageBlob = container.GetBlockBlobReference(uri);
        MemoryStream image = new MemoryStream();
        imageBlob.DownloadToStream(image);
        image.Seek(0, SeekOrigin.Begin);

        // create a thumbnail image and upload into a blob
        string thumbnailUri =
String.Concat(Path.GetFileNameWithoutExtension(uri), "_thumb.jpg");
        CloudBlockBlob thumbnailBlob =
container.GetBlockBlobReference(thumbnailUri);
        thumbnailBlob.UploadFromStream(CreateThumbnail(image));

        // update the entry in table storage to point to the thumbnail
        var ds = new GuestBookEntryDataSource();
        ds.UpdateImageThumbnail(partitionKey, rowkey,
thumbnailBlob.Uri.AbsoluteUri);

        // remove message from queue
        queue.DeleteMessage(msg);

        Trace.TraceInformation("Generated thumbnail in blob '{0}'.",
thumbnailBlob.Uri);
      }
      else
      {
        System.Threading.Thread.Sleep(1000);
      }
    }
    catch (StorageClientException e)
    {
      Trace.TraceError("Exception when processing queue item. Message:
'{0}'", e.Message);
      System.Threading.Thread.Sleep(5000);
    }
  }
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 WorkerRole Run – Visual Basic*)

Visual Basic

```
Public Overrides Sub Run()
  Trace.TraceInformation("Listening for queue messages...")
  Do
    Try
      ' retrieve a new message from the queue
      Dim msg As CloudQueueMessage = queue.GetMessage()
      If msg IsNot Nothing Then
        ' parse message retrieved from queue
        Dim messageParts = msg.AsString.Split(New Char() {","c})
        Dim uri = messageParts(0)
        Dim partitionKey = messageParts(1)
        Dim rowkey = messageParts(2)
        Trace.TraceInformation("Processing image in blob '{0}'.", uri)

        ' download original image from blob storage
        Dim imageBlob As CloudBlockBlob =
container.GetBlockBlobReference(uri)
        Dim image As New MemoryStream()
        imageBlob.DownloadToStream(image)
        image.Seek(0, SeekOrigin.Begin)

        ' create a thumbnail image and upload into a blob
        Dim thumbnailUri As String =
String.Concat(Path.GetFileNameWithoutExtension(uri), "_thumb.jpg")
        Dim thumbnailBlob As CloudBlockBlob =
container.GetBlockBlobReference(thumbnailUri)
        thumbnailBlob.UploadFromStream(CreateThumbnail(image))

        ' update the entry in table storage to point to the thumbnail
        Dim ds = New GuestBookEntryDataSource()
        ds.UpdateImageThumbnail(partitionKey, rowkey,
thumbnailBlob.Uri.AbsoluteUri)

        ' remove message from queue
        queue.DeleteMessage(msg)

        Trace.TraceInformation("Generated thumbnail in blob '{0}'.",
thumbnailBlob.Uri)
      Else
        System.Threading.Thread.Sleep(1000)
      End If
    Catch e As StorageClientException
      Trace.TraceError("Exception when processing queue item. Message:
'{0}'", e.Message)
      System.Threading.Thread.Sleep(5000)
    End Try
  Loop

End Sub
```

11. Finally, add the following method to the **WorkerRole** class to create thumbnails from a given image.

   (Code Snippet – *Introduction to Windows Azure - Ex01 CreateThumbnail - C#*)

C#

```
private Stream CreateThumbnail(Stream input)
{
  var orig = new Bitmap(input);
  int width;
  int height;

  if (orig.Width > orig.Height)
  {
    width = 128;
    height = 128 * orig.Height / orig.Width;
  }
  else
  {
    height = 128;
    width = 128 * orig.Width / orig.Height;
  }

  var thumb = new Bitmap(width, height);

  using (Graphics graphic = Graphics.FromImage(thumb))
  {
    graphic.InterpolationMode =
System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;
    graphic.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
    graphic.PixelOffsetMode =
System.Drawing.Drawing2D.PixelOffsetMode.HighQuality;
    graphic.DrawImage(orig, 0, 0, width, height);
    var ms = new MemoryStream();
    thumb.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
    ms.Seek(0, SeekOrigin.Begin);
    return ms;
  }
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 CreateThumbnail - Visual Basic*)

Visual Basic

```
Private Function CreateThumbnail(ByVal input As Stream) As Stream
  Dim orig As New Bitmap(input)
  Dim width As Integer
  Dim height As Integer

  If orig.Width > orig.Height Then
    width = 128
    height = 128 * orig.Height / orig.Width
  Else
    height = 128
    width = 128 * orig.Width / orig.Height
  End If
  Dim thumb As New Bitmap(width, height)

  Using graphic = Graphics.FromImage(thumb)
    graphic.InterpolationMode =
Drawing2D.InterpolationMode.HighQualityBicubic
    graphic.SmoothingMode = Drawing2D.SmoothingMode.AntiAlias
    graphic.PixelOffsetMode = Drawing2D.PixelOffsetMode.HighQuality
    graphic.DrawImage(orig, 0, 0, width, height)
    Dim ms As New MemoryStream()
    thumb.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
    ms.Seek(0, SeekOrigin.Begin)
    Return ms
  End Using
End Function
```

12. The worker role also uses Windows Azure storage services and you need to configure your storage account settings, just as you did in the case of the web role. To create the storage account setting, in **Solution Explorer**, expand the **Roles** node in the **GuestBook** project, double-click **GuestBook_WorkerRole** 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. In the **Storage Connection String** dialog, choose the **Use development storage** option and click **OK**. Press **CTRL + S** to save your changes.

Task 5 – Using Queues to Dispatch Jobs to the Worker Role

In this task, you update the front-end web role to dispatch work items to the queue so that the worker role can process guest book entries and generate thumbnails for the images.

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. Declare a new member field in the **_Default** class for the queue client.

C#

```
private static CloudQueueClient queueStorage;
```

Visual Basic

```
Private Shared queueStorage As CloudQueueClient
```

3. Update the **SignButton_Click** event handler to queue a work item to generate a thumbnail for the uploaded image. To do this, insert the following (highlighted) code immediately following the code that creates a new entry in table storage.

   (Code Snippet – *Introduction to Windows Azure - Ex01 Queue Work Item – C#*)

C#

```
protected void SignButton_Click(object sender, EventArgs e)
{
  if (FileUpload1.HasFile)
  {
    InitializeStorage();

    ...

    // 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() };
    GuestBookEntryDataSource ds = new GuestBookEntryDataSource();
    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}",
uniqueBlobName, 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 - Ex01 Queue Work Item – Visual Basic*)

Visual Basic

```
Protected Sub SignButton_Click(ByVal sender As Object, ByVal e As
EventArgs) Handles SignButton.Click
  If FileUpload1.HasFile Then
    InitializeStorage()

    ...

    ' 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 GuestBookEntryDataSource()
    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}",
uniqueBlobName, 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
```

Note: 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.

4. Finally, update the **InitializeStorage** method to create the queue by inserting the following (highlighted) code immediate after the code that configures the blob container for public access.

   (Code Snippet – *Introduction to Windows Azure - Ex01 Create Queue – C#*)

C#

```
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)
    {
       ...
    }

    storageInitialized = true;
  }
}
```

(Code Snippet – *Introduction to Windows Azure - Ex01 Create Queue – Visual Basic*)

Visual Basic

```
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 Try

    storageInitialized = True
  End SyncLock
End Sub
```

Note: The updated code creates a queue that the web role uses to submit new jobs to the worker role.

Verification

In this task, you test the GuestBook application in the Development Fabric. The Development Fabric, or simple devfabric, is a simulated environment for developing and testing Windows Azure applications in your machine.

1. Press **F5** to execute the service. The service builds and then launches the local development fabric. To show the development fabric UI, right-click its icon located in the system tray and select **Show Development Fabric UI**.

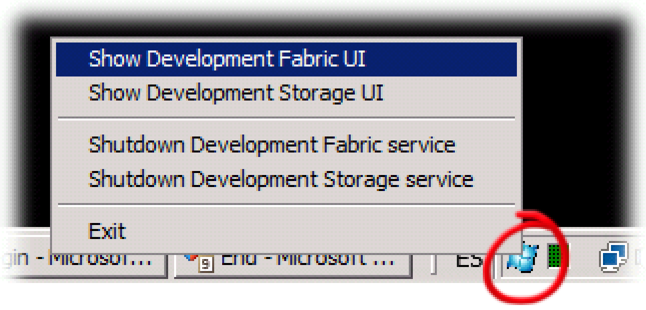

Show Development Fabric UI
Show Development Storage UI
Shutdown Development Fabric service
Shutdown Development Storage service
Exit

Figure 17
Showing the development fabric UI

2. Switch to Internet Explorer to view the GuestBook application.

3. 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.

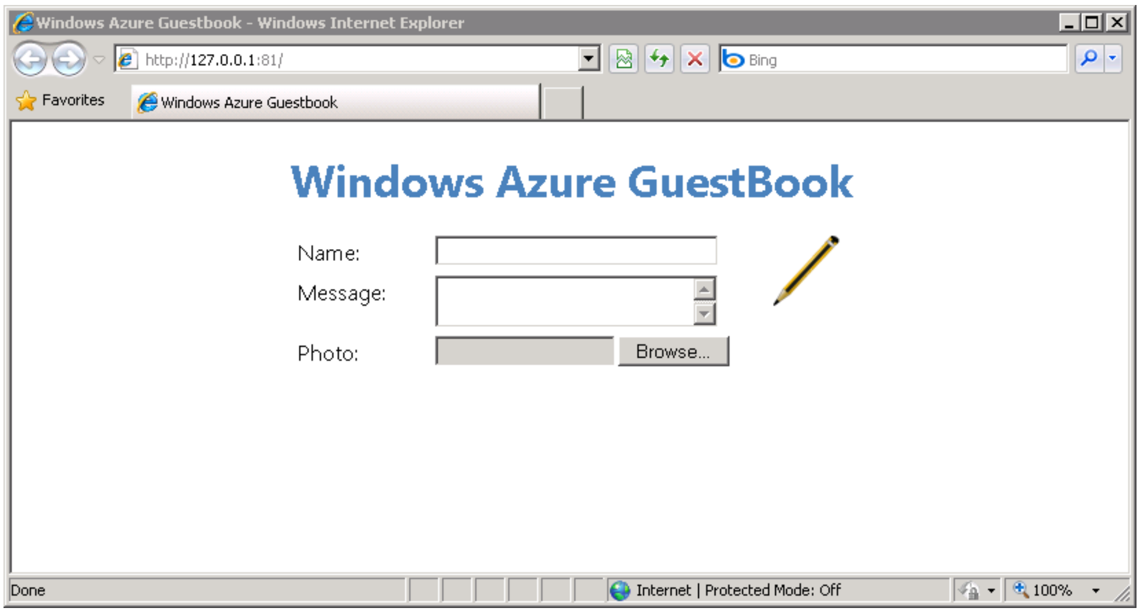


Figure 18

Windows Azure GuestBook home page

It is a good idea to choose a large hi-res image because the guestbook service will resize it.

Once you submit an entry, the web role creates a new entry 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.

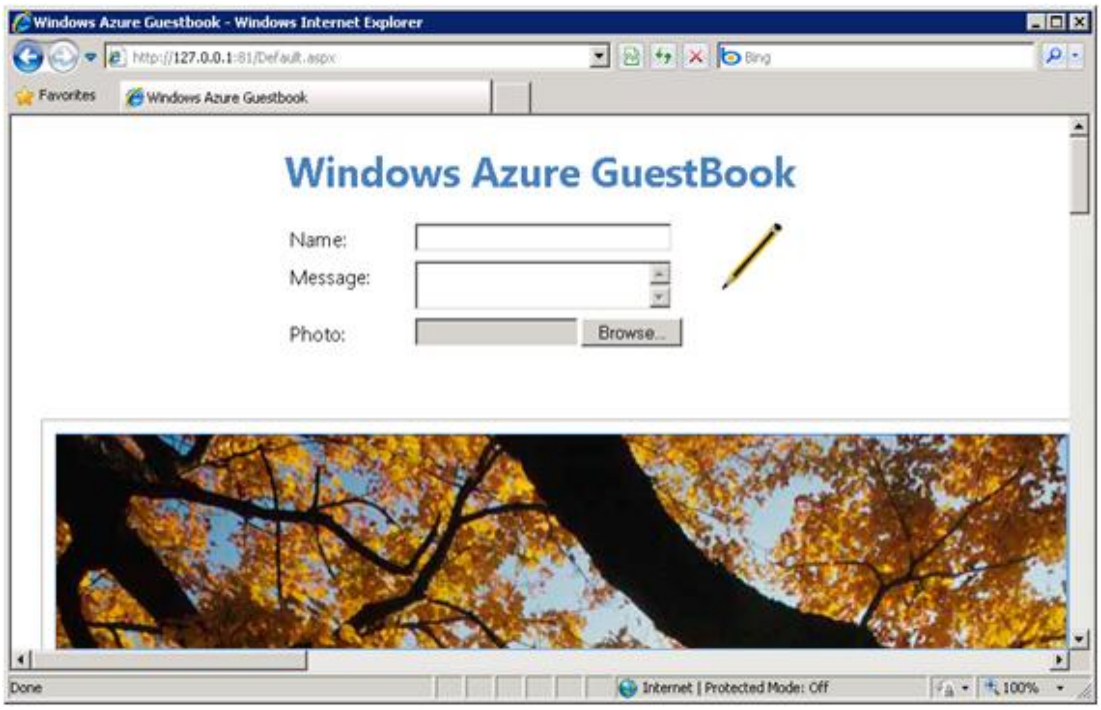

Windows Azure Guestbook - Windows Internet Explorer
http://127.0.0.1:81/Default.aspx
Bing
Favorites
Windows Azure Guestbook
Windows Azure GuestBook
Name:
Message:
Photo:
Browse...
Done
Internet | Protected Mode: Off
100%

Figure 19
GuestBook showing the image with its original size

After a few seconds, the page refreshes and displays the thumbnail that the worker role generated instead.

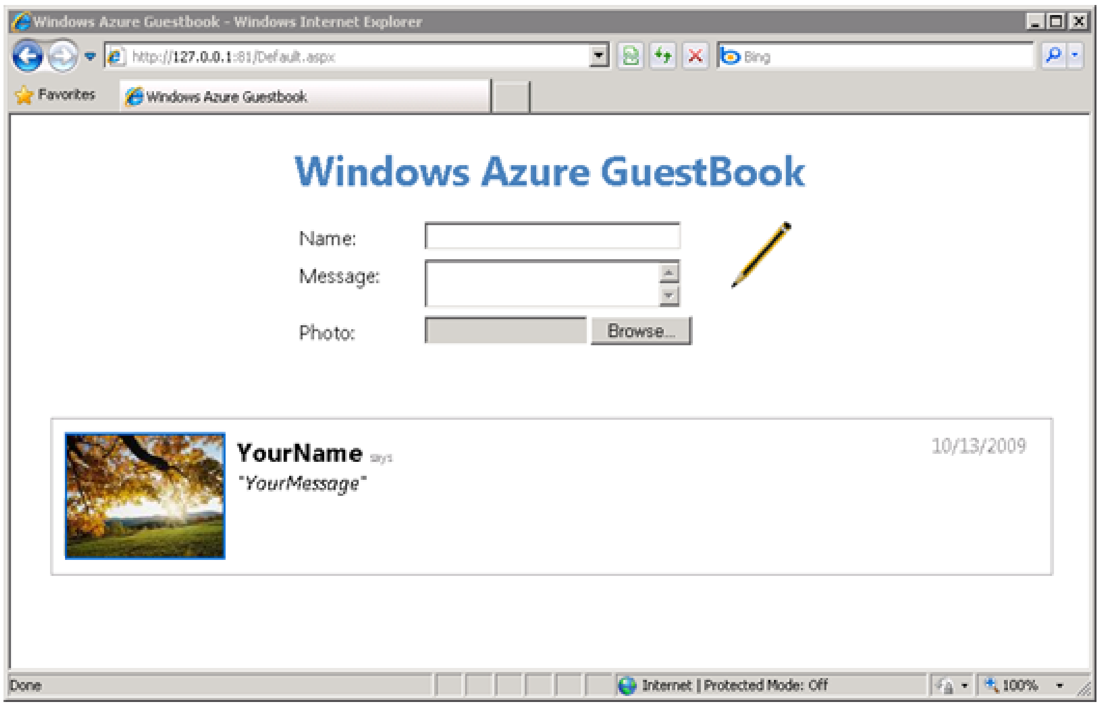


Figure 20

GuestBook showing the generated thumbnail

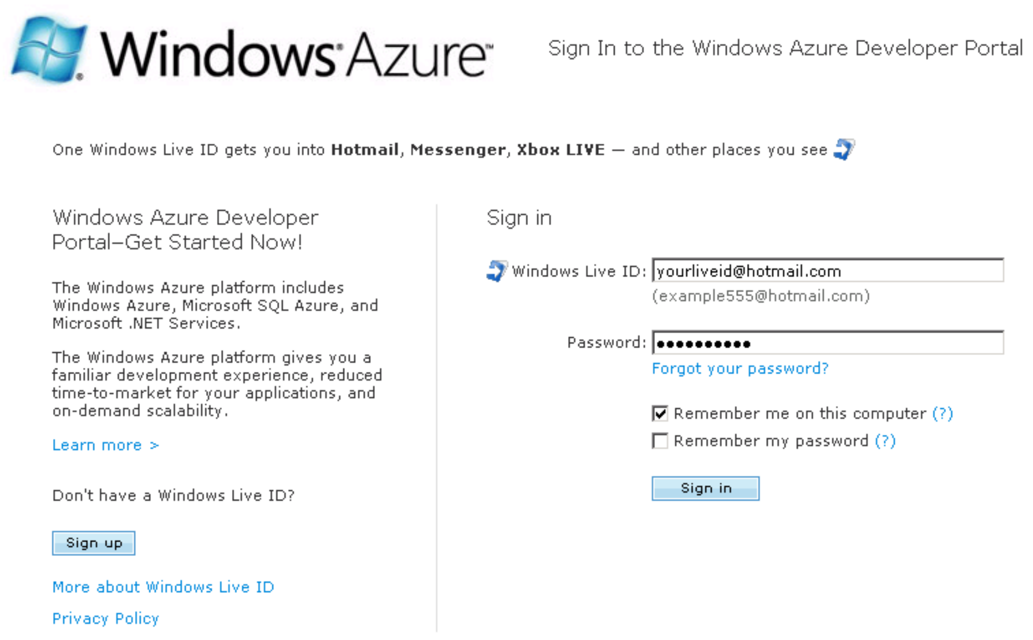
Deploying a Windows Azure Application

In this exercise, you deploy the application created in the previous exercise to Windows Azure using the Windows Azure Developer Portal. First, you register to obtain a Windows Azure developer account and provision the required service components. Next, you upload the application package to the staging area and configure it. You then execute the application in the staging area to verify its operation. Finally, you promote the application to production.

Task 1 – Creating a Windows Azure Account

In this task, you create a new Windows Azure account and redeem an invitation token to enable the creation of hosted services.

4. Navigate to http://windows.azure.com using a Web browser and sign in using your Windows Live ID.


Windows Azure Sign In to the Windows Azure Developer Portal

One Windows Live ID gets you into **Hotmail**, **Messenger**, **Xbox LIVE** — and other places you see

Windows Azure Developer Portal–Get Started Now!

The Windows Azure platform includes Windows Azure, Microsoft SQL Azure, and Microsoft .NET Services.

The Windows Azure platform gives you a familiar development experience, reduced time-to-market for your applications, and on-demand scalability.

Learn more >

Don't have a Windows Live ID?

Sign up

More about Windows Live ID

Privacy Policy

Sign in

Windows Live ID: yourliveid@hotmail.com
(example555@hotmail.com)

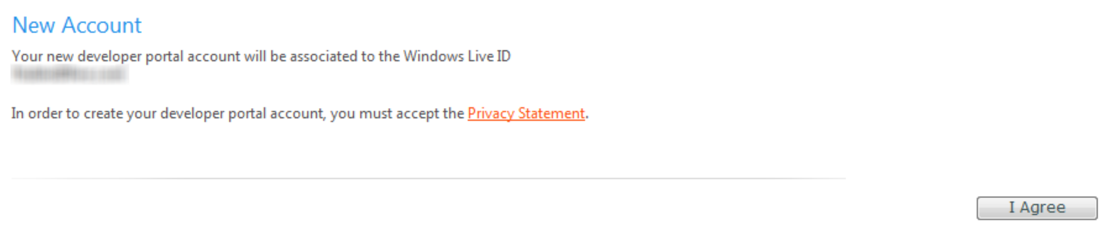
Password: ●●●●●●●●●●
Forgot your password?

[✓] Remember me on this computer (?)
☐ Remember my password (?)

Sign in

Figure 21
Signing in to the Windows Azure portal

5. If this is your first visit to the Windows Azure Developer Portal, create a new account associated to your Windows Live ID. Review the **Privacy Statement** and click **I Agree** to proceed.


New Account

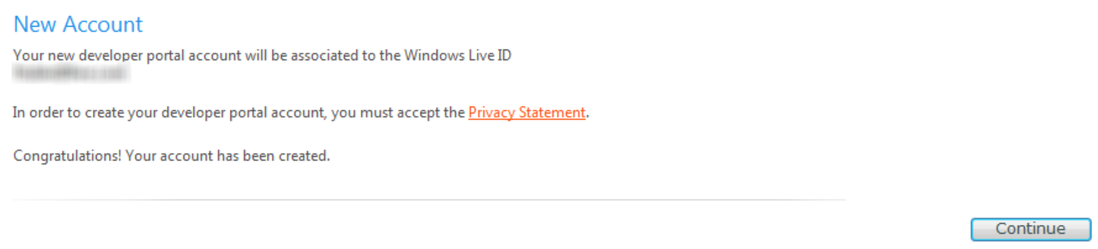
Your new developer portal account will be associated to the Windows Live ID
[redacted]

In order to create your developer portal account, you must accept the Privacy Statement.

I Agree

Figure 22
Creating a new Windows Azure Developer Portal account

6. After creating the account, click **Continue** to proceed.


New Account

Your new developer portal account will be associated to the Windows Live ID
[redacted]

In order to create your developer portal account, you must accept the Privacy Statement.

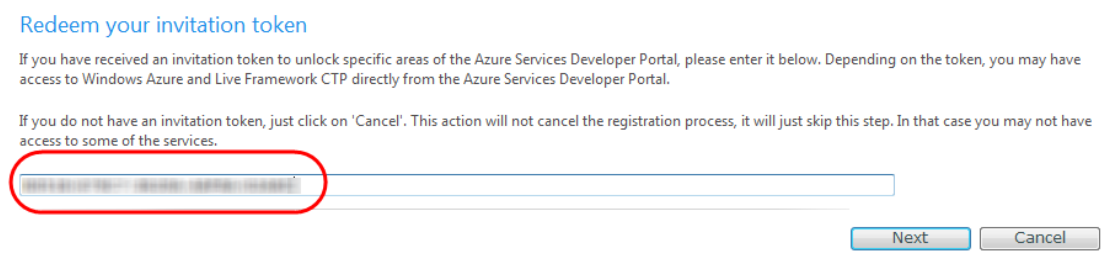
Congratulations! Your account has been created.

Continue

Figure 23
Successful creation of the account

7. Enter your invitation token and click **Next** to redeem it.

> **Note:** Invitation tokens are required to enable services at the Windows Azure Developer Portal. To obtain an invitation token, you will need to register at http://connect.microsoft.com to obtain a product key for the Windows Azure Platform.



Redeem your invitation token

If you have received an invitation token to unlock specific areas of the Azure Services Developer Portal, please enter it below. Depending on the token, you may have access to Windows Azure and Live Framework CTP directly from the Azure Services Developer Portal.

If you do not have an invitation token, just click on 'Cancel'. This action will not cancel the registration process, it will just skip this step. In that case you may not have access to some of the services.

[redacted]

Figure 24

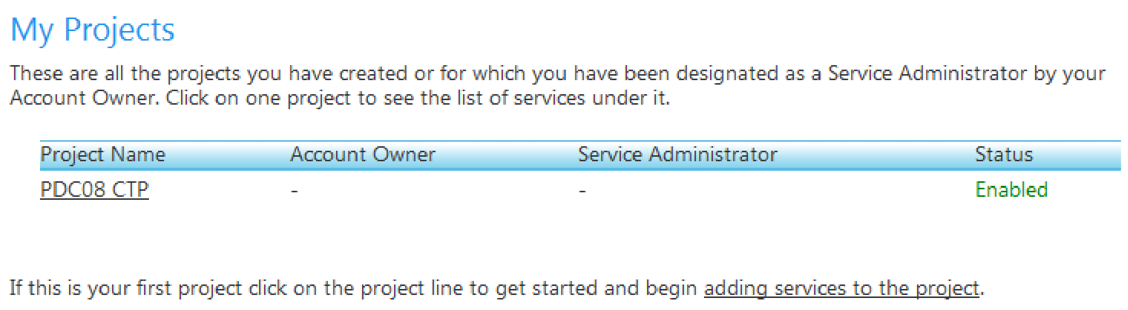
Redeeming an invitation token to enable the creation of hosted services

Task 2 – Creating a Storage Account and a Hosted Service Component

The application you deploy in this exercise requires both compute and storage services. In this task, you create a new Windows Azure storage account to allow the application to persist its data. In addition, you define a hosted service component to execute application code.

1. In **My Projects**, locate the project where you plan to deploy the application and click the **Project Name** link to view the list of services under it.

> **Note:** The items in this list are projects associated with your Windows Live ID. Following commercial launch, you will see every project that you have purchased listed here.



My Projects

These are all the projects you have created or for which you have been designated as a Service Administrator by your Account Owner. Click on one project to see the list of services under it.

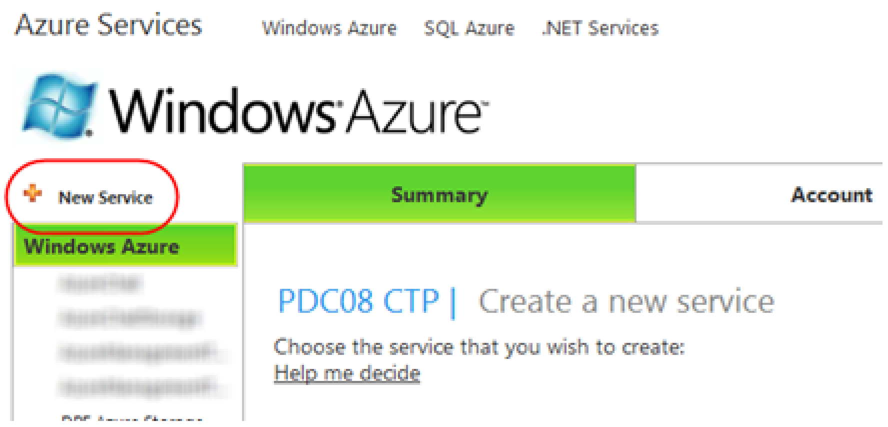
Project Name	Account Owner	Service Administrator	Status
PDC08 CTP	-	-	Enabled

If this is your first project click on the project line to get started and begin adding services to the project.

Figure 25

Available projects in your Windows Azure account

2. First, you create the storage account that the application will use to store its data. In the Windows Azure **Summary** page, click **New Service**.



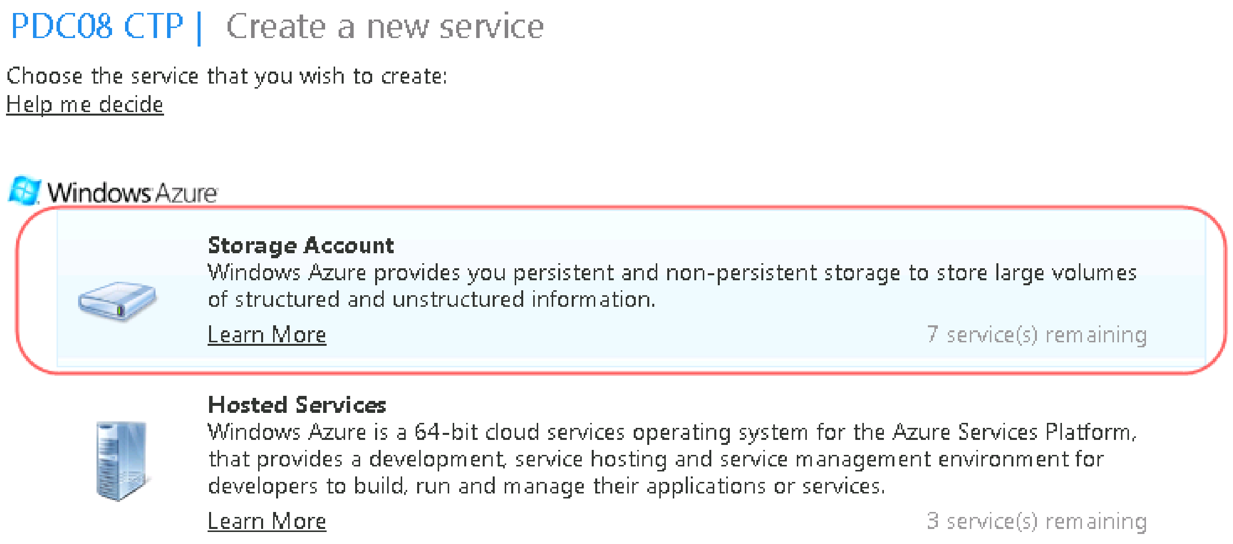
PDC08 CTP | Create a new service

Choose the service that you wish to create:
Help me decide

Figure 26

Creating a new Windows Azure service

3. In the Windows Azure **Summary** page, click **Storage Account**.



PDC08 CTP | Create a new service

Choose the service that you wish to create:
Help me decide

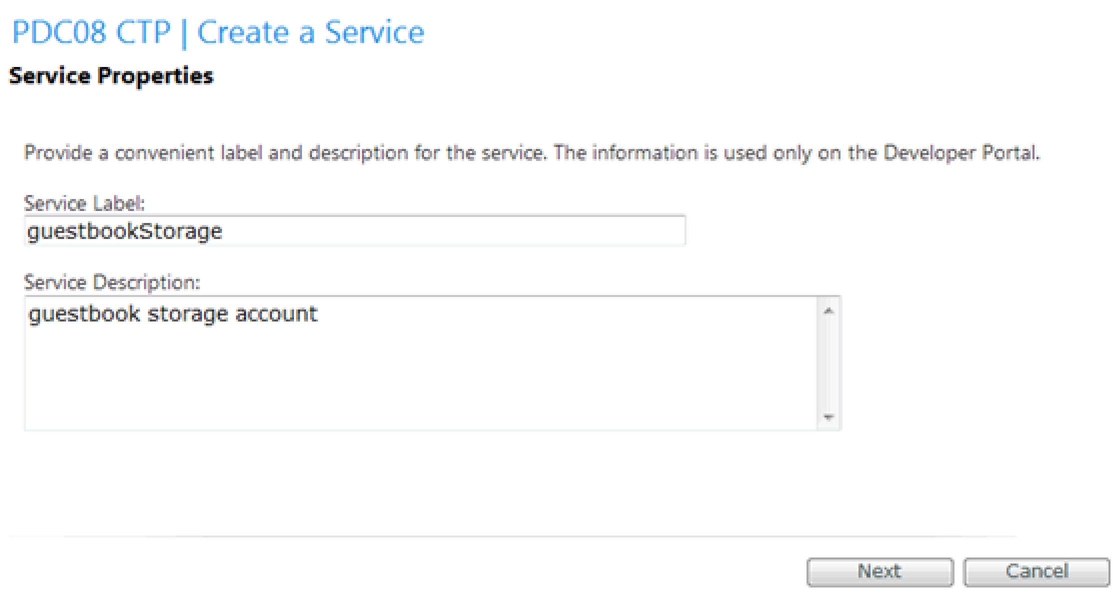
Storage Account
Windows Azure provides you persistent and non-persistent storage to store large volumes of structured and unstructured information.
Learn More 7 service(s) remaining

Hosted Services
Windows Azure is a 64-bit cloud services operating system for the Azure Services Platform, that provides a development, service hosting and service management environment for developers to build, run and manage their applications or services.
Learn More 3 service(s) remaining

Figure 27

Creating a new Windows Azure storage account

4. Enter the **Service Component Label** and **Service Component Description** and click **Next** to proceed. The values you enter here allow you to identify the account storage component in the portal user interface.


PDC08 CTP | Create a Service

Service Properties

Provide a convenient label and description for the service. The information is used only on the Developer Portal.

Service Label:
guestbookStorage

Service Description:
guestbook storage account

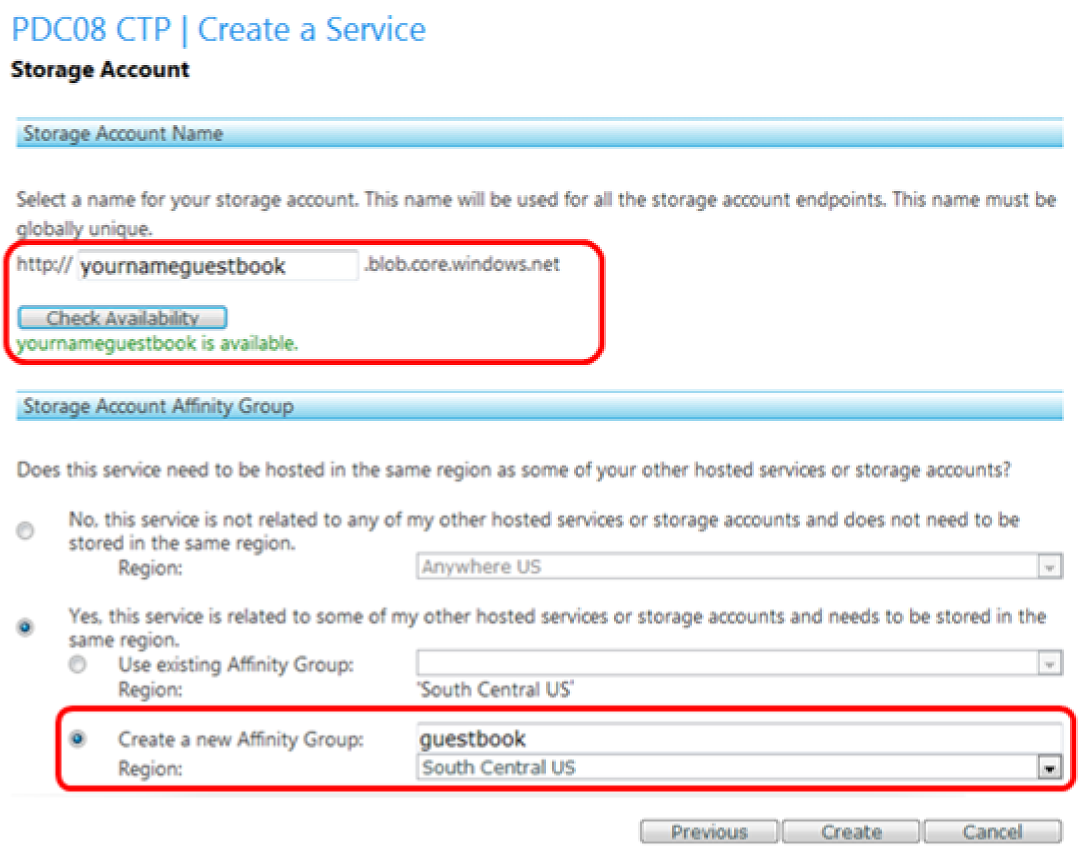
Figure 28

Configuring the service component properties

5. In the **Storage Account Name** section, enter the name for your storage account, for example, ***<yourname>guestbook***, where *<yourname>* is a unique name. Windows Azure uses this value to generate the endpoint URLs for the storage account services. To ensure that the name is valid, click **Check Availability** to verify that the name satisfies the naming rules and is currently available.

> **Note:** The name used for the storage account corresponds to a DNS name and is subject to standard DNS naming rules. Moreover, the name is publicly visible and must therefore be unique.

6. In the **Storage Account Affinity Group** section, click "**Yes, this service is related to some of my other hosted services or storage accounts and needs to be stored in the same region.**" and then select the "**Create a new Affinity Group Region**" option. Set the value of the affinity group to the label **guestbook** and choose the region where you wish your service to run, most likely, the one that is closest to your current location.


PDC08 CTP | Create a Service

Storage Account

Storage Account Name

Select a name for your storage account. This name will be used for all the storage account endpoints. This name must be globally unique.

http:// yournameguestbook .blob.core.windows.net

yournameguestbook is available.

Storage Account Affinity Group

Does this service need to be hosted in the same region as some of your other hosted services or storage accounts?

No, this service is not related to any of my other hosted services or storage accounts and does not need to be stored in the same region.
Region: Anywhere US

[✓ Yes, this service is related to some of my other hosted services or storage accounts and needs to be stored in the same region.]
Use existing Affinity Group:
Region: 'South Central US'

[✓ Create a new Affinity Group:] guestbook
Region: South Central US

Figure 29

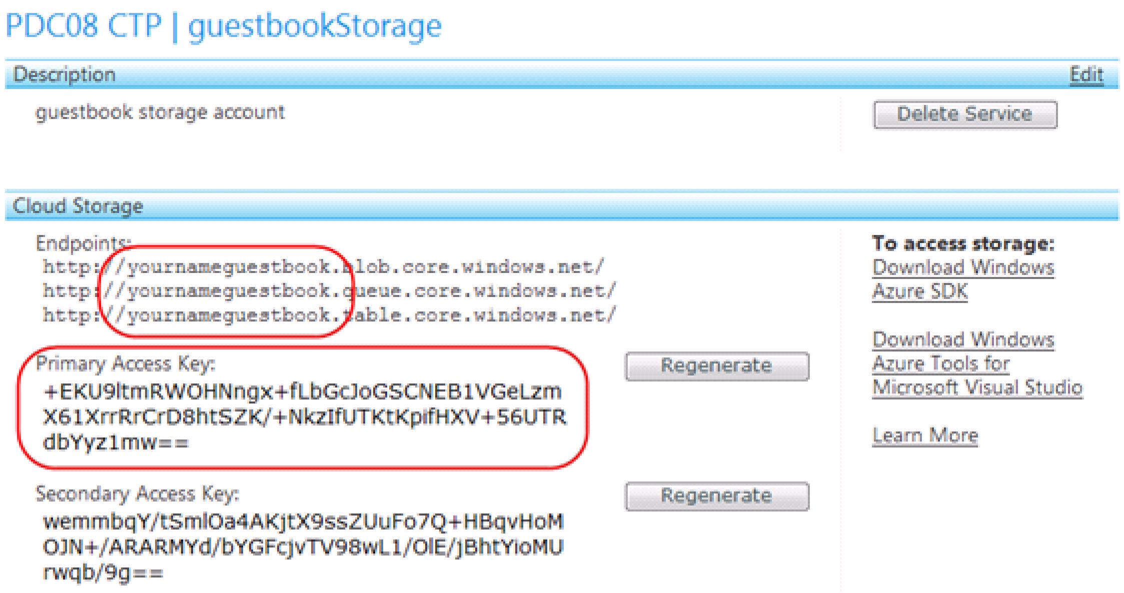
Configuring the storage account URL and affinity group

Note: The reason that you are creating a new affinity group is to deploy both the hosted service and storage account to the same location, thus ensuring high bandwidth and low latency between the application and the data it depends on.

7. Click **Create** to register your new storage account. Wait until the account provisioning process completes and updates the **Summary** page. Notice the available **Endpoints** and the **Access Keys** assigned to the storage account. Record the **Storage Account Name** and the value assigned to the **Primary Access Key**. You will use these values later on to configure the application.

Note: The **Endpoints** specify the URLs of the blob, queue, and table storage services for this account.

The **Primary Access Key** and **Secondary Access Key** both provide a shared secret that you can use to access storage. The secondary key gives the same access as the primary key and is used for backup purposes. You can regenerate each key independently in case either one is compromised.


PDC08 CTP | guestbookStorage

Description

guestbook storage account

Cloud Storage

Endpoints:

```
http://yournameguestbook.blob.core.windows.net/
http://yournameguestbook.queue.core.windows.net/
http://yournameguestbook.table.core.windows.net/
```

Primary Access Key:

+EKU9ltmRWOHNngx+fLbGcJoGSCNEB1VGeLzm
X61XrrRrCrD8htSZK/+NkzIfUTKtKpifHXV+56UTR
dbYyz1mw==

Secondary Access Key:

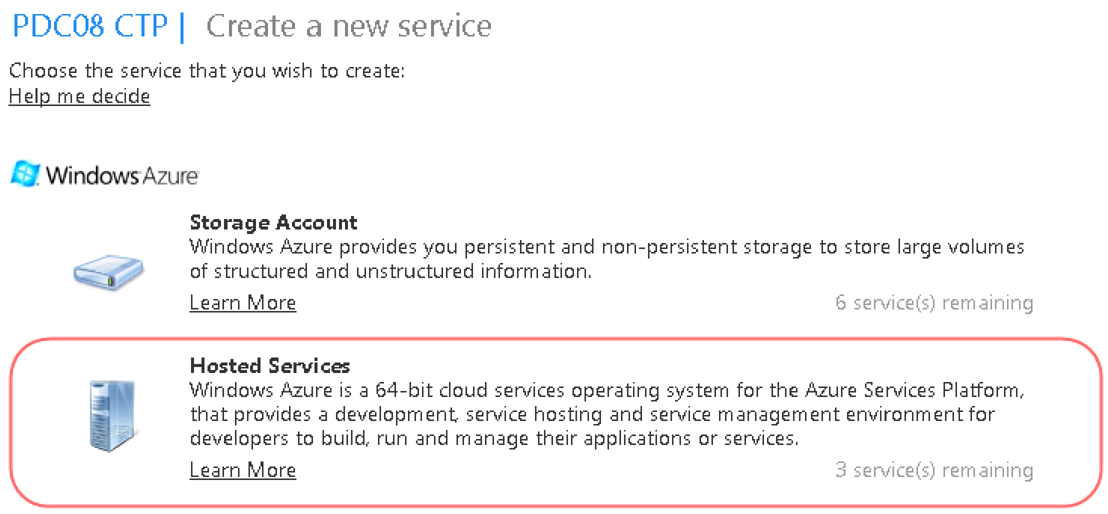
wemmbqY/tSmlOa4AKjtX9ssZUuFo7Q+HBqvHoM
OJN+/ARARMYd/bYGFcjvTV98wL1/OlE/jBhtYioMU
rwqb/9g==

Edit
Delete Service
Regenerate
Regenerate
To access storage:
Download Windows Azure SDK
Download Windows Azure Tools for Microsoft Visual Studio
Learn More

Figure 30

Successful creation of the storage account

8. Next, create the compute component that executes the application code. Click **New Service** once again to create a hosted services account for the application. In the **Project** section, click **Hosted Services**.


PDC08 CTP | Create a new service

Choose the service that you wish to create:

Help me decide

Windows Azure

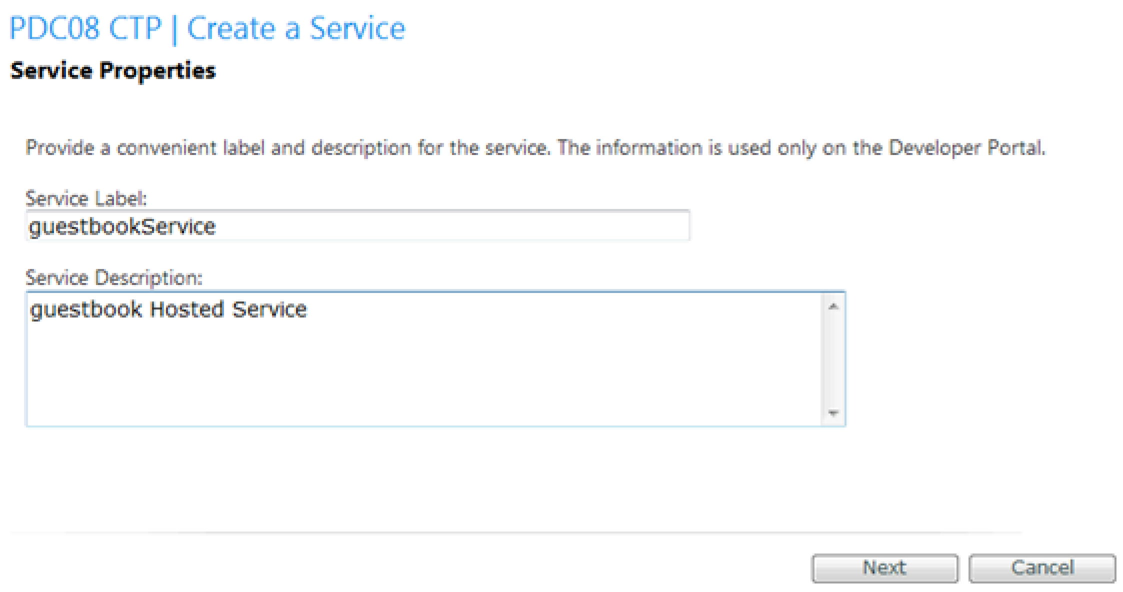
Storage Account
Windows Azure provides you persistent and non-persistent storage to store large volumes of structured and unstructured information.
Learn More — 6 service(s) remaining

Hosted Services
Windows Azure is a 64-bit cloud services operating system for the Azure Services Platform, that provides a development, service hosting and service management environment for developers to build, run and manage their applications or services.
Learn More — 3 service(s) remaining

Figure 31

Creating a new hosted service

9. Enter the **Service Component Label** and **Service Component Description** and click **Next** to proceed. The values you enter here allow you to identify the hosted service component in the portal user interface.


PDC08 CTP | Create a Service

Service Properties

Provide a convenient label and description for the service. The information is used only on the Developer Portal.

Service Label: guestbookService

Service Description: guestbook Hosted Service

Figure 32

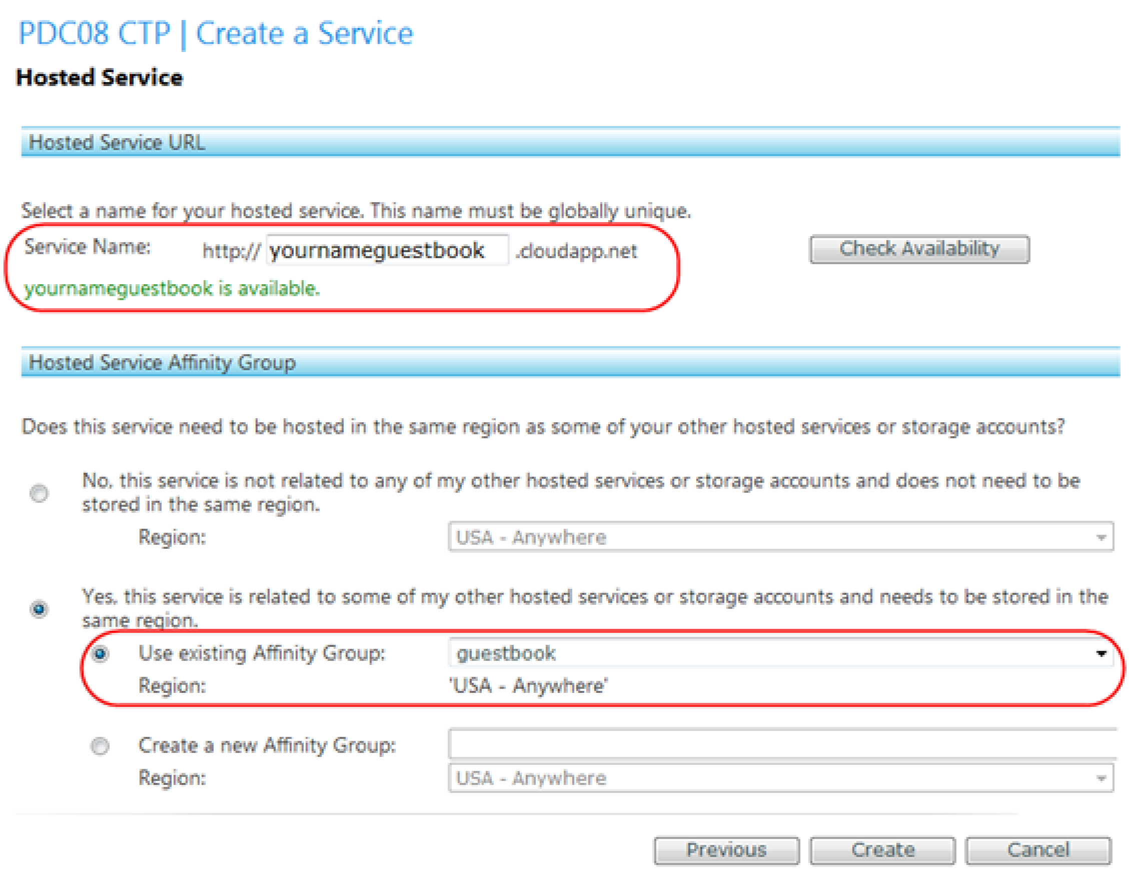
Configuring the service component properties

10. In the **Hosted Service URL** section, enter the name for your hosted service, for example, ***<yourname>guestbook***, where *<yourname>* is a unique name. Windows Azure uses this value to generate the endpoint URL of the hosted service. To ensure that the name is valid, click **Check Availability** to verify that the name satisfies the naming rules and is currently available.

Note: If possible, choose the same name for both the storage account and hosted service. However, you may need to choose a different name if the one you select is unavailable.

11. In the **Storage Account Affinity Group** section, click "**Yes, this service is related to some of my other hosted services or storage accounts and needs to be stored in the same region.**" and then select the "**Use existing Affinity Group Region**" option. Select **GuestBook** in the affinity group drop down list.

Note: By choosing **GuestBook** as the affinity group, you ensure that the hosted service is deployed to the same location as the storage account that you provisioned earlier.


PDC08 CTP | Create a Service

Hosted Service

Hosted Service URL

Select a name for your hosted service. This name must be globally unique.

Service Name: http:// yournameguestbook .cloudapp.net

yournameguestbook is available.

Hosted Service Affinity Group

Does this service need to be hosted in the same region as some of your other hosted services or storage accounts?

No, this service is not related to any of my other hosted services or storage accounts and does not need to be stored in the same region.

Region: USA - Anywhere

[✓ Yes, this service is related to some of my other hosted services or storage accounts and needs to be stored in the same region.]

[✓ Use existing Affinity Group:] guestbook

Region: 'USA - Anywhere'

Create a new Affinity Group:

Region: USA - Anywhere

Figure 33

Configuring the hosted service URL and affinity group

12. Click **Create** to register the hosted service and wait until the provisioning process completes.

Task 3 – Deploying the Application to the Windows Azure Developer Portal

A hosted service is a service that runs your code in the Windows Azure environment. It has two separate deployment states: staging and production. A staging deployment allows you to test your service in the Windows Azure environment before you deploy it to production.

In this task, you deploy the application code to the staging environment but first, you have to generate the package.

1. If it is not already open, launch Microsoft Visual Studio 2008 in elevated administrator mode, from **Start | All Programs | Microsoft Visual Studio 2008** by right clicking the **Microsoft Visual Studio 2008** shortcut and choosing **Run as Administrator**.
2. In the **File** menu, choose **Open** and then **Project/Solution**. In the **Open Project** dialog, browse to **Ex01-BuildingYourFirstWindowsAzureApp** in the **Source** folder of the lab and choose the folder for the language of your preference (Visual C# or Visual Basic). Select **End.sln** in the **End** folder and click **Open**. Alternatively, you may continue with the solution that you obtained after completing **Exercise 1**.

3. Generate the package to deploy to the cloud. To do this, right-click the **GuestBook** cloud project and select **Publish**. After a few seconds, Windows Explorer will open with the current folder set to the location where Visual Studio generated the package.
4. In the hosted service **Summary** page of the Windows Azure portal, click **Deploy** to upload the service package to the portal.

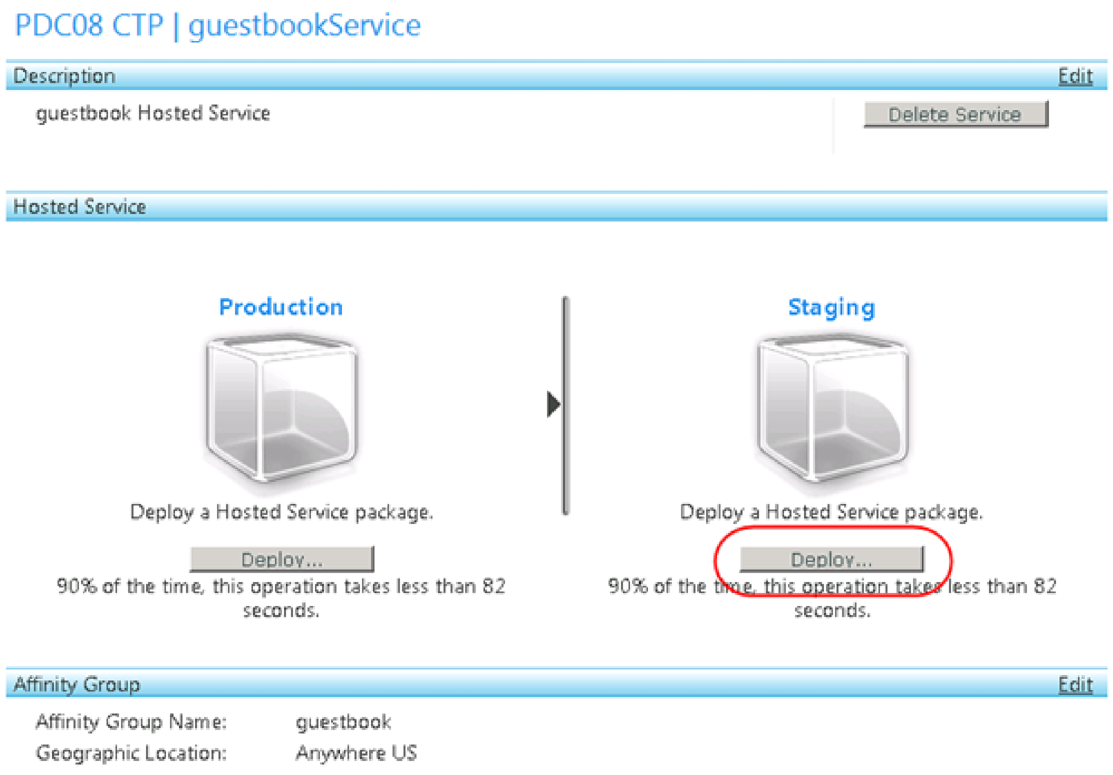

PDC08 CTP | guestbookService
Description
Edit
guestbook Hosted Service
Delete Service
Hosted Service
Production
Deploy a Hosted Service package.
Deploy...
90% of the time, this operation takes less than 82 seconds.
Staging
Deploy a Hosted Service package.
Deploy...
90% of the time, this operation takes less than 82 seconds.
Affinity Group
Edit
Affinity Group Name: guestbook
Geographic Location: Anywhere US

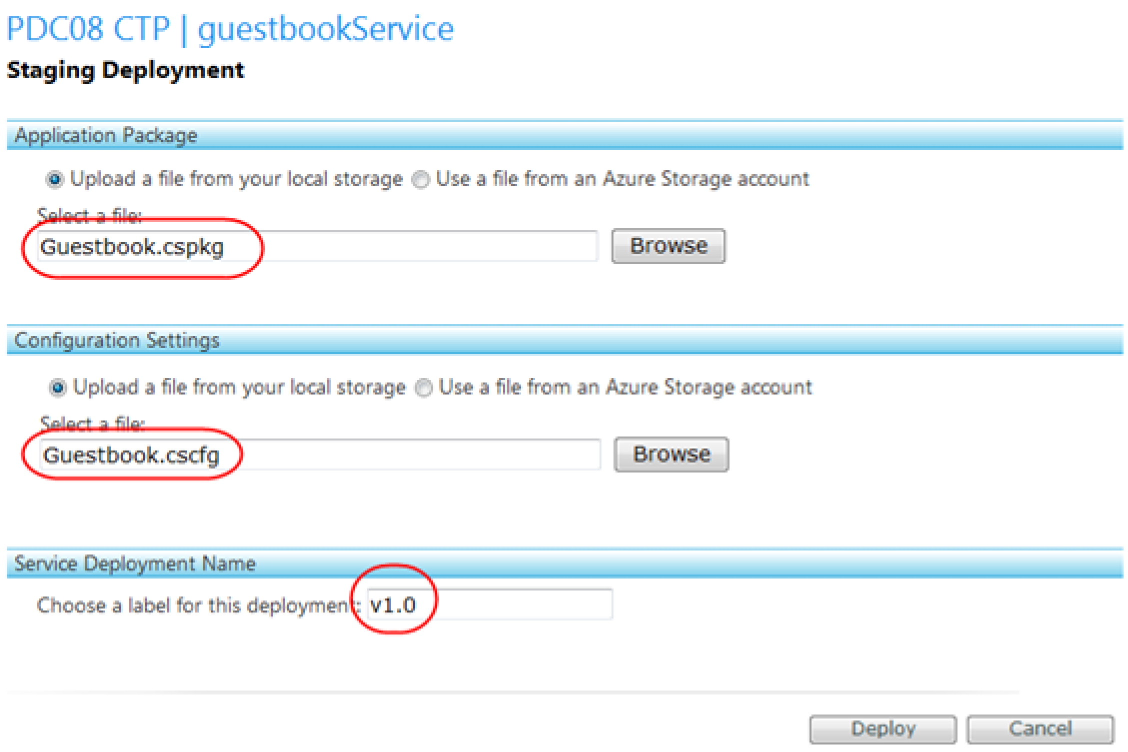
Figure 34
Hosted service summary page

5. In the **Staging Deployment** page, under the **App Package** section, click "**Upload a file from your local storage**" to select it. Click **Browse** and navigate to the folder where Visual Studio generated the package in Step 3 and select **GuestBook.cspkg**.
6. In the **Configuration Settings** section, once again choose "**Upload a file from your local storage**", click **Browse** and select **ServiceConfiguration.cscfg** in the same folder that you used in the previous step.

Note: The *.cscfg* file contains configuration settings for the application, including storage settings that you will update later in the exercise.

7. Finally, in the **Properties** section, enter a label to identify the deployment; for example, use **v1.0**.

Note: The portal displays the label in its user interface for staging and production, which allows you to identify the version currently deployed in each environment.


PDC08 CTP | guestbookService

Staging Deployment

Application Package

(●) Upload a file from your local storage ( ) Use a file from an Azure Storage account

Select a file: Guestbook.cspkg

Configuration Settings

(●) Upload a file from your local storage ( ) Use a file from an Azure Storage account

Select a file: Guestbook.cscfg

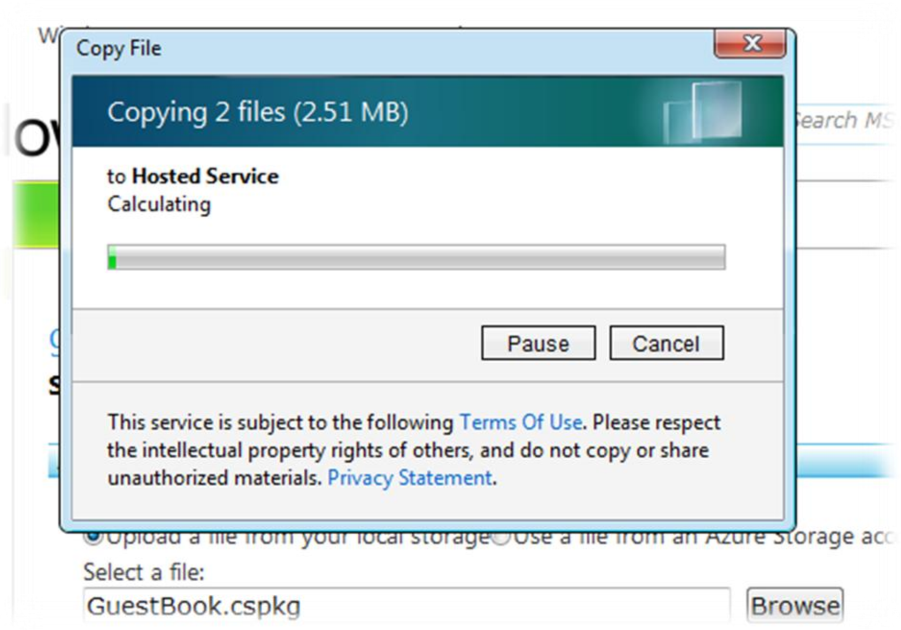
Service Deployment Name

Choose a label for this deployment: v1.0

Figure 35

Configuring the service package deployment

8. Click **Deploy** to start uploading the package to the Windows Azure Developer Portal.


Copy File

Copying 2 files (2.51 MB)

to **Hosted Service**
Calculating

This service is subject to the following Terms Of Use. Please respect the intellectual property rights of others, and do not copy or share unauthorized materials. Privacy Statement.

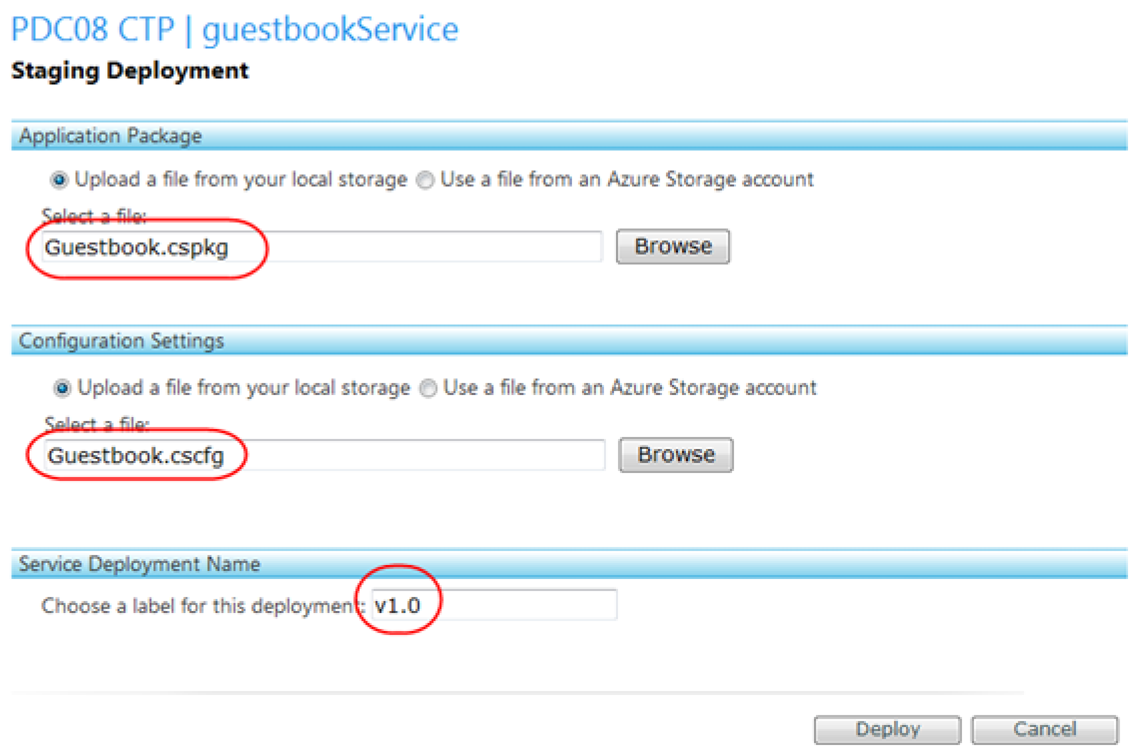
Select a file: GuestBook.cspkg

Figure 36

Uploading a service package to the Windows Azure Developer Portal

9. Wait until the deployment process finishes, which may take several minutes. At this point, you have already uploaded the package and it is in an **Allocated** state. Notice that the portal assigned this deployment a **Web Site URL** that includes a unique identifier. Shortly, you will access this URL to test the application and determine whether it operates correctly in the Windows Azure environment, but first you need to configure it.

Note: During deployment, Windows Azure analyzes the configuration file and copies the service to the correct number of machines, ready to start. Load balancers, network devices and monitoring are also configured during this time.


PDC08 CTP | guestbookService

Staging Deployment

Application Package

Upload a file from your local storage [✓] / Use a file from an Azure Storage account

Select a file: Guestbook.cspkg

Configuration Settings

Upload a file from your local storage [✓] / Use a file from an Azure Storage account

Select a file: Guestbook.cscfg

Service Deployment Name

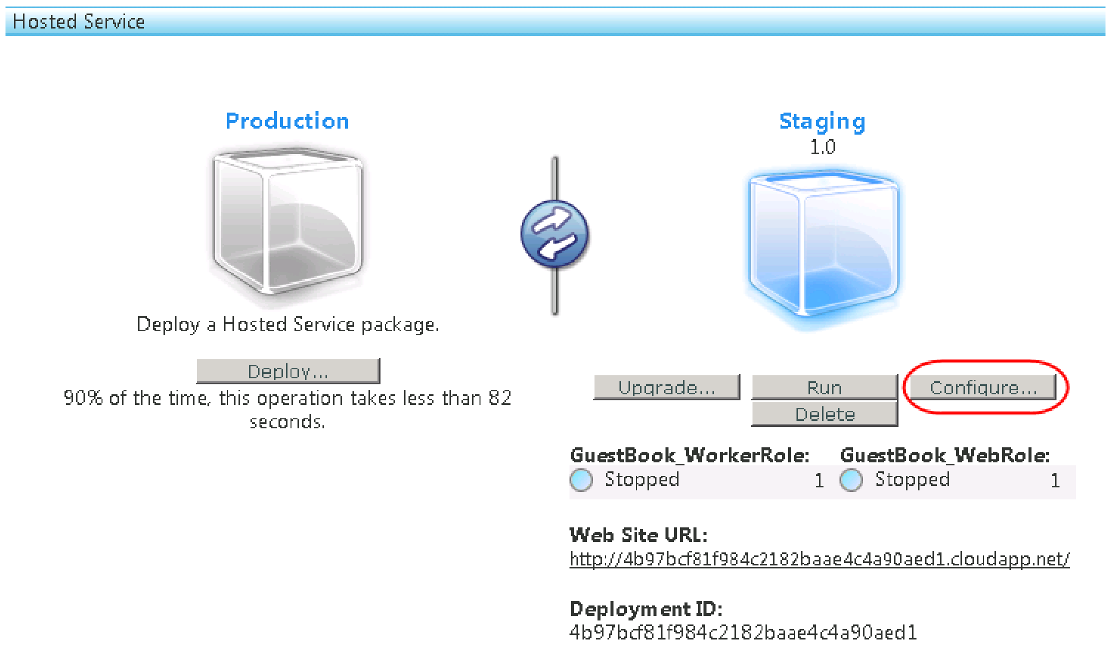
Choose a label for this deployment: v1.0

Figure 37
Package successfully deployed

Task 4 – Configuring the Application to Use the Storage Account

Before you can test the deployed application, you need to configure it. In this task, you define the storage account settings for the application.

1. In the **Summary** page, under the **Hosted Service** section, click **Configure** for the Staging environment.


Hosted Service
Production
Deploy a Hosted Service package.
Deploy...
90% of the time, this operation takes less than 82 seconds.
Staging
1.0
Upgrade...
Run
Delete
Configure...

GuestBook_WorkerRole: Stopped 1
GuestBook_WebRole: Stopped 1

Web Site URL:
http://4b97bcf81f984c2182baae4c4a90aed1.cloudapp.net/

Deployment ID:
4b97bcf81f984c2182baae4c4a90aed1

Figure 38

Configuring application settings

2. In the **Service Tuning** page, under **Configuration Settings**, locate the **GuestBook_WebRole** configuration settings and update the value of the **AccountName** setting by replacing the placeholder labeled [YOUR_ACCOUNT_NAME] with the **Storage Account Name** that you chose when you configured the storage account in Task 2. If you followed the recommendation, the name should follow the pattern ***<yourname>guestbook***, where *<yourname>* is a unique name.

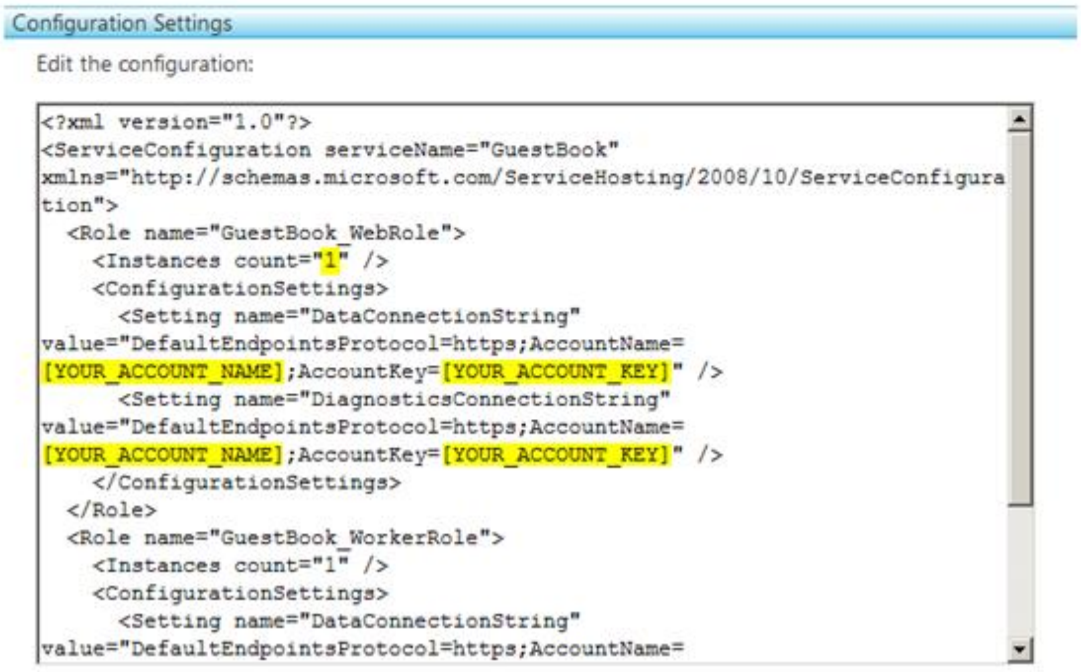
Note: If you continued using the solution created in Exercise 1, you will not find the placeholders mentioned above. In that case, update the configuration with the following:

```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="GuestBook"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfigur
ation">
  <Role name="GuestBook_WebRole">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="DataConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=[YOUR_ACCOUNT_NAME];Acco
untKey=[YOUR_ACCOUNT_KEY]" />
      <Setting name="DiagnosticsConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=[YOUR_ACCOUNT_NAME];Acco
untKey=[YOUR_ACCOUNT_KEY]" />
    </ConfigurationSettings>
  </Role>
  <Role name="GuestBook_WorkerRole">
    <Instances count="1" />
    <ConfigurationSettings>
```

```
      <Setting name="DataConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=[YOUR_ACCOUNT_NAME];Acco
untKey=[YOUR_ACCOUNT_KEY]" />
      <Setting name="DiagnosticsConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=[YOUR_ACCOUNT_NAME];Acco
untKey=[YOUR_ACCOUNT_KEY]" />
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

3. Next, update the **AccountSharedKey** setting by replacing the placeholder labeled [YOUR_ACCOUNT_KEY] with the **Primary Shared Key** value that you recorded earlier when you created the storage account in Task 2.
4. Finally, locate the **Instances** element and change the **count** attribute to a value of 2.

Note: The **Instances** setting controls the number of roles that Windows Azure starts and is used to scale the service. Currently, it is limited to a maximum of two instances. However, once the service is commercially available, you will be able to change it to any number that you are willing to pay for.



```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="GuestBook"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfigura
tion">
  <Role name="GuestBook_WebRole">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="DataConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=
[YOUR_ACCOUNT_NAME];AccountKey=[YOUR_ACCOUNT_KEY]" />
      <Setting name="DiagnosticsConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=
[YOUR_ACCOUNT_NAME];AccountKey=[YOUR_ACCOUNT_KEY]" />
    </ConfigurationSettings>
  </Role>
  <Role name="GuestBook_WorkerRole">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="DataConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=
```

Figure 39
Setting the storage account name and shared key

Note: The configuration is simply an XML document that contains the value of the settings declared in the service definition file and its initial content is determined by the

ServiceConfiguration.cscfg file that you uploaded earlier, when you deployed the package in Task 3.

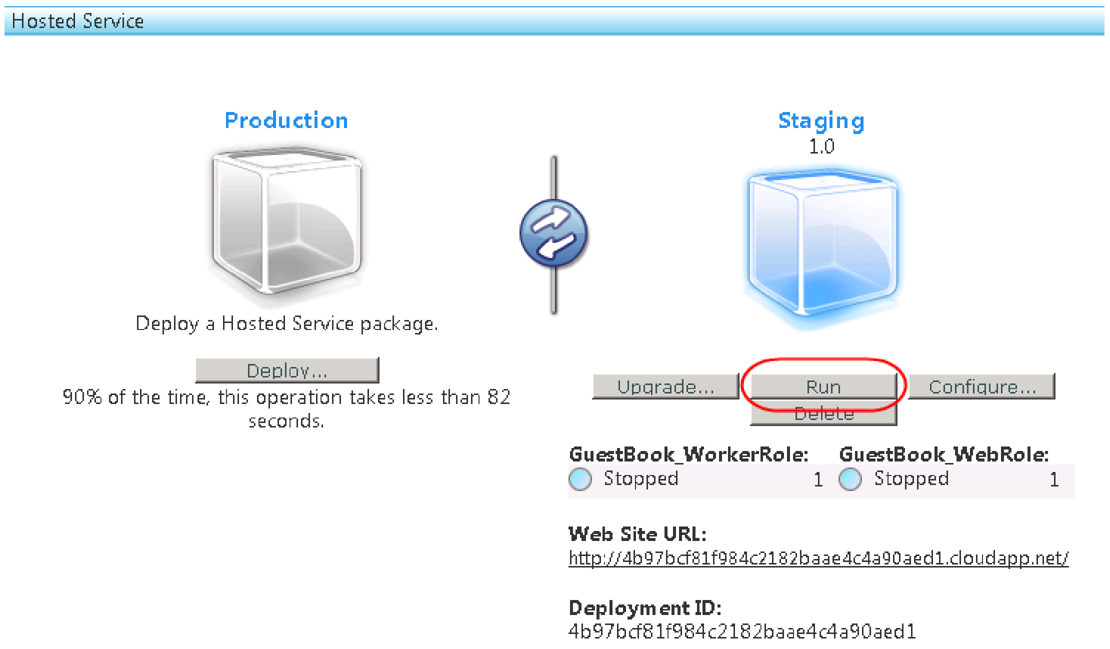
5. Repeat steps 2 - 3 to update the storage account settings for **GuestBook_WorkerRole.**
6. Click **Save** to update the configuration and wait for the hosted service to apply the new settings.

Note: The portal displays a legend "Package is updating..." while the settings are applied.

Task 5 – Testing the Application in the Staging Environment

In this task, you run the application in the staging environment and access its Web Site URL to test that it operates correctly.

1. In the **Summary** page, under the **Hosted Service** section, click **Run** for the deployment in the Staging environment.


Hosted Service
Production
Deploy a Hosted Service package.
Deploy...
90% of the time, this operation takes less than 82 seconds.
Staging
1.0
Upgrade...
Run
Delete
Configure...

GuestBook_WorkerRole: Stopped 1
GuestBook_WebRole: Stopped 1

Web Site URL:
http://4b97bcf81f984c2182baae4c4a90aed1.cloudapp.net/

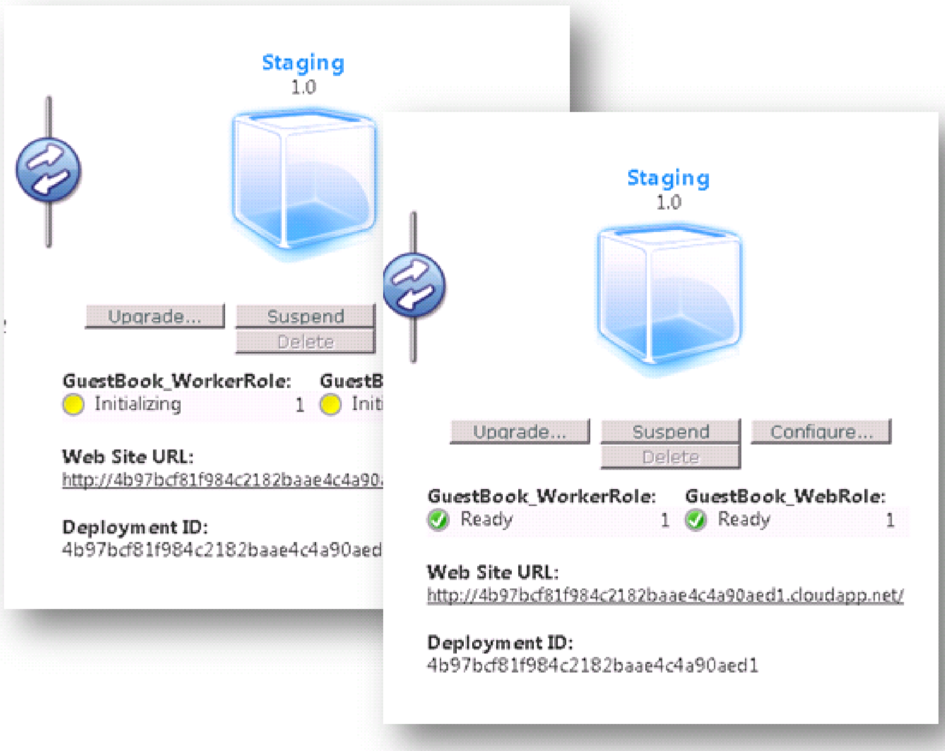
Deployment ID:
4b97bcf81f984c2182baae4c4a90aed1

Figure 40

Running the application in the staging environment

2. Wait until the status of the service is shown first as **Initializing** and then as **Ready.**

Note: The process may take several minutes during which Windows Azure provisions servers, load balancers and other network devices, as well as preparing to monitor the application. Once the application is running, it takes care of patching the OS, recovering from hardware failure, and ensuring that storage is available.



GuestBook_WorkerRole: Initializing 1 GuestB Initi

Web Site URL: http://4b97bcf81f984c2182baae4c4a90

Deployment ID: 4b97bcf81f984c2182baae4c4a90aed

GuestBook_WorkerRole: Ready 1 GuestBook_WebRole: Ready 1

Web Site URL: http://4b97bcf81f984c2182baae4c4a90aed1.cloudapp.net/

Deployment ID: 4b97bcf81f984c2182baae4c4a90aed1

Figure 41
Application initialization process

3. Once the service has started, click the **Web Site URL** link to open a new browser window that points to the application.

Note: The address URL is shown as *<guid>.cloudapp.net*, where *<guid>* is some random identifier. This is different from the address where the application will run once it is in production. Although the application executes in a staging area that is separate from the production environment, there is no actual physical difference between staging and production – it is simply a matter of where the load balancer is connected.

In the future, you will be able to have multiple “virtual” areas, for test, QA, pre-production, etc...

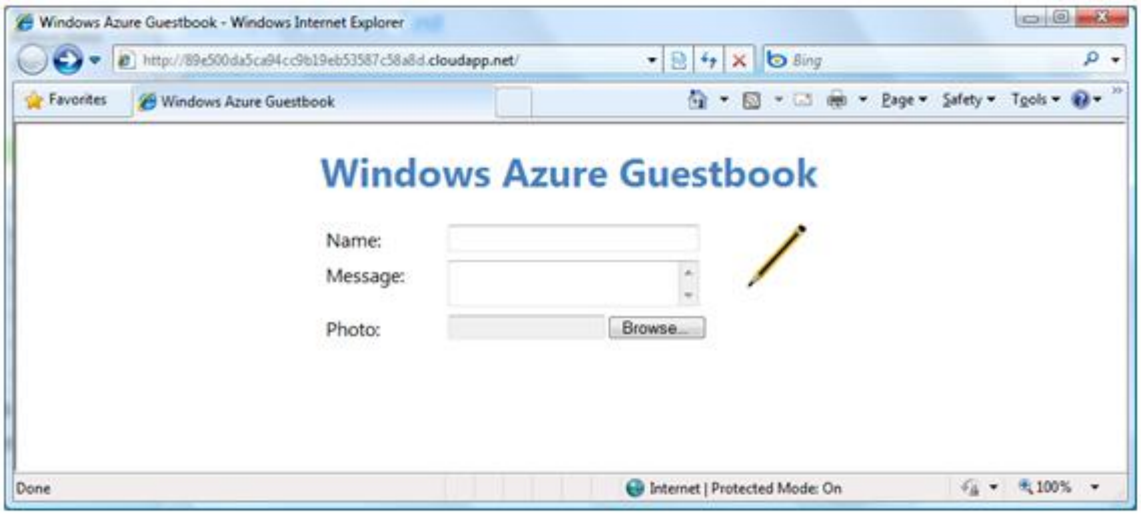

Windows Azure Guestbook - Windows Internet Explorer
http://89e500da5ca94cc9b19eb53587c58a8d.cloudapp.net/
Favorites
Windows Azure Guestbook
Windows Azure Guestbook
Name:
Message:
Photo:
Browse...
Done
Internet | Protected Mode: On
100%

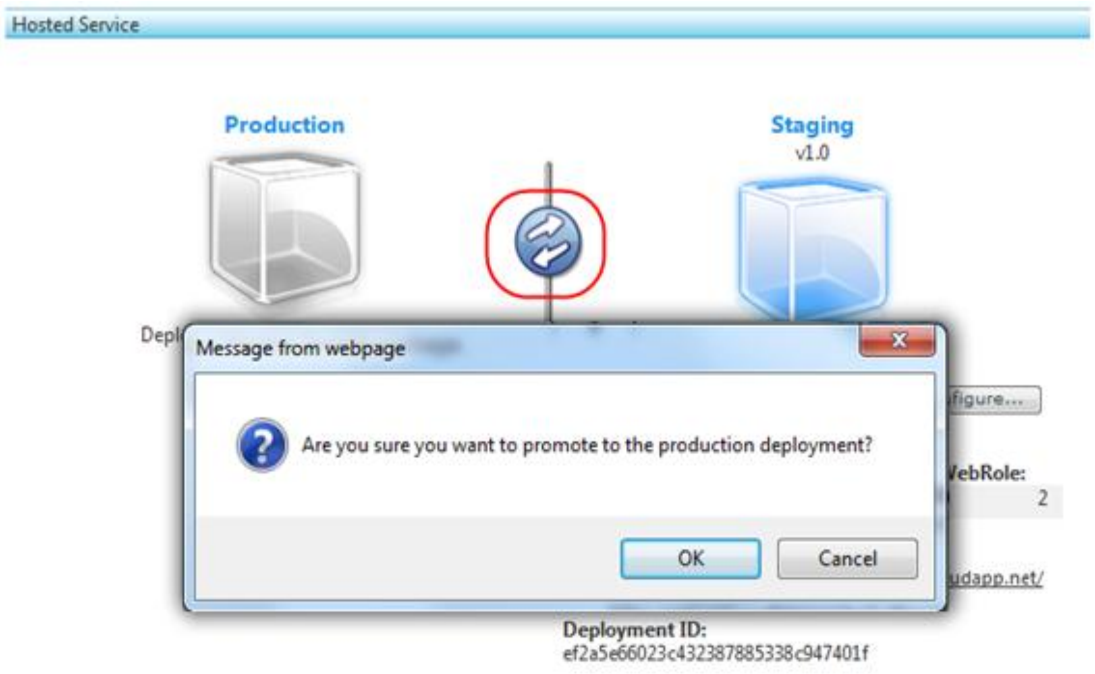
Figure 42
Application running in the staging environment

4. If you wish, you can test the application by signing the guestbook and uploading an image.

Task 6 – Promoting the Application to Production

Now that you have verified that the service is working correctly in the staging environment, you are ready to promote it to final production. When you deploy the application to production, Windows Azure reconfigures its load balancers so that the application is available at its production URL.

1. In the **Summary** page, under the **Hosted Service** section, click the **Swap** button (this is the circular button located between the Production and Staging areas). When prompted, click **OK** to confirm that you wish to promote to the production deployment.


Hosted Service

Production

Staging
v1.0

Message from webpage

Are you sure you want to promote to the production deployment?

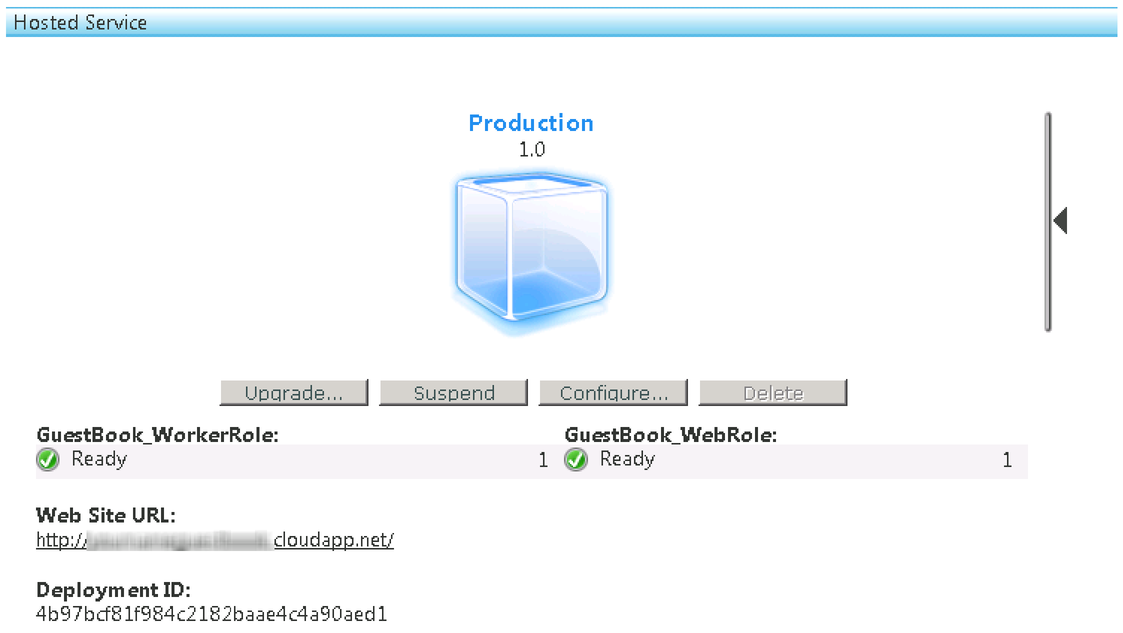
OK Cancel

Deployment ID:
ef2a5e66023c432387885338c947401f

Figure 43

Promoting the application to the production deployment

2. Wait for the promotion process to complete, which typically takes a few seconds. Notice that the Production service status is now shown as ***Ready*** and that the **Web Site URL** corresponds to the URL that you chose earlier, when you configured the hosted service in Task 2.


Hosted Service

Production
1.0

GuestBook_WorkerRole:
✓ Ready 1

GuestBook_WebRole:
✓ Ready 1

Web Site URL:
http://[illegible]cloudapp.net/

Deployment ID:
4b97bcf81f984c2182baae4c4a90aed1

Figure 44

Application successfully deployed to production

3. Click the **Web Site URL** link to open the production site in a browser window and notice the URL in the address bar.

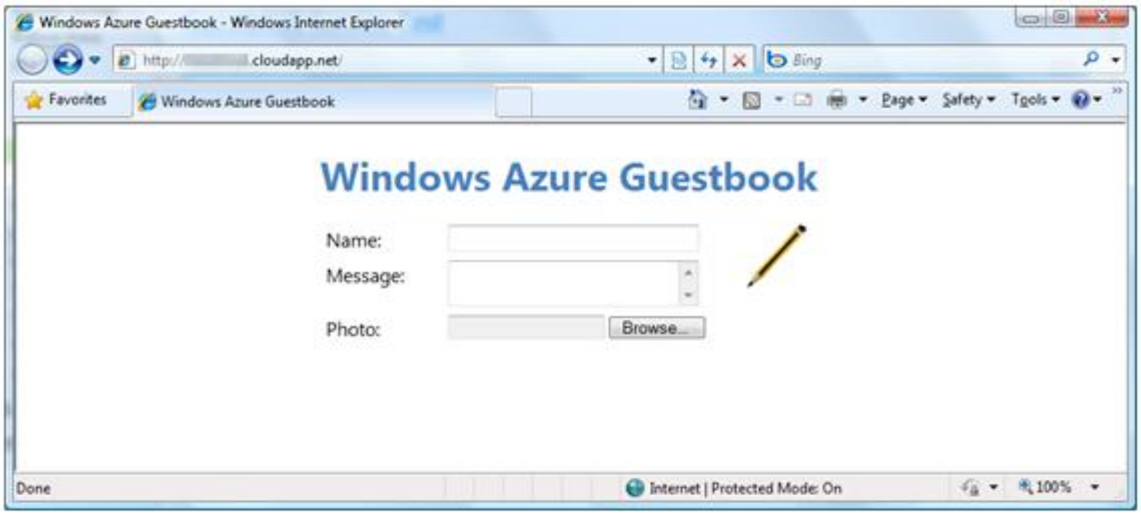

Windows Azure Guestbook - Windows Internet Explorer
http://[redacted].cloudapp.net/
Bing
Favorites
Windows Azure Guestbook
Page
Safety
Tools
Windows Azure Guestbook
Name:
Message:
Photo:
Browse...
Done
Internet | Protected Mode: On
100%

Figure 45
Application running in the production environment

Note: If you visit the production site shortly after its promotion, the DNS name might not be ready. If you encounter a DNS error (404), wait a few minutes and try again. Keep in mind that Windows Azure creates DNS name entries dynamically and that the changes might take few minutes to propagate.

Summary

By completing this, you have explored the basic elements of Windows Azure applications. You have seen that services consist of one or more web roles and worker roles. You have learnt about Windows Azure storage services and in particular, blob, table and queue services. Finally, you have explored a basic architectural pattern for cloud applications that allows front-end processes to communicate with back-end processes using queues.

Glossary

1. API – Application Programming Interface.
2. Azure - An operating system for the cloud developed by Microsoft Corporation that provides computation, storage, Application development and automated service management.
3. Blobs – Storage space in azure those are stored in a container. The blobs are of two types viz: page and block.
4. Block – it is a type of blob. It has semantics.
5. Cloud Computing- The term cloud means Internet; it got its name through its diagrammatic representation in the books. Computation in a virtual environment i.e. in the internet and performing partial or complete computational activities in the internet at a remote machine is called as cloud computing.
6. CTP – Community Technology Preview that provides two instances viz: Web roles and Worker roles.
7. Data center – Storage area managed for all the applications provided by Microsoft live.
8. Fabric controllers- Controls the Fabric and is comprised of 6 to 7 machines in order to manage and control the large group of machines i.e. fabric.
9. Fabric- the set of machines dedicated to Windows Azure is organized into a fabric.
10. HTTP – Hyper Text Transfer Protocol, for the client-server communication in the cloud.
11. IIS – Internet Information Services.
12. ISV – Independent Software Vendor, (ISV) could create an application that targets business users, an approach that's often referred to as Software as a Service (SaaS).
13. Live Services-Through the Live Framework provides access to data from Microsoft's live applications and others. The Live Framework also allows synchronizing this data across desktops and devices, finding and downloading applications, and more.
14. Load balances – They are used to manage the traffic and load in the cloud when the demand for particular service is high.
15. Microsoft .NET Services- Offers distributed infrastructure services to cloud-based and local applications.
16. Microsoft SQL Services- Provides data services in the cloud based on SQL Server.
17. Page- it is a type of blob. It is a random read/write.
18. Queues – The storage units in which the messages are placed.
19. Tables – The storage units in azure, which are the collection of entities. Entities must have a Partition Key and Row Key.
20. VM- Virtual Machine, Provides independent platform irrespective of Heterogeneity.
21. Web Role – The Web role instance that receives this request can write a message into a queue describing the work to be done
22. Windows Azure- Provides a Windows-based environment for running applications and storing data on servers in Microsoft data centers.
23. Worker Role – A Worker role instance that's waiting on this queue can then read the message and carry out the task it specifies.

References:

- www.windowsazure.com
- http://channel9.msdn.com/learn
- Hosting.com
- World Economic forum – weforum.org
- http://pluralsight.com
- INTRODUCING THE AZURE SERVICES PLATFORM, “AN EARLY LOOK AT WINDOWS AZURE, .NET SERVICES, SQL SERVICES, AND LIVE SERVICES” DAVID CHAPPELL.
- “INTRODUCING WINDOWS AZURE”, DAVID CHAPPELL.
- WWW.microsoft.com