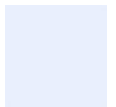




**Microsoft®**

Azure Storage



**Hands-on  
lab**

Azure provides different mechanisms to store your data. Each mechanism has a proper use and a recommended scenario. How to choose the right storage service by scenario? How to work with each different storage service on Azure?

This lab is a guidance for building Windows Azure data management services into your applications. Learn to choose the correct data service for each scenario and how to write applications for resiliency and robustness in the cloud with blobs, tables and queues.

This document supports a preliminary release of a software product that may be changed substantially prior to final commercial release. This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright 2012 © Microsoft Corporation. All rights reserved.

Microsoft *<list other MS trademarks used alphabetically>* are trademarks of the Microsoft group of companies.

*<This is where mention of specific, contractually obligated to, third party trademarks should be listed.>*

All other trademarks are property of their respective owners.

# Introduction

## Estimated time to complete this lab

60 minutes, Log on the machine with user TR20ATTENDEE, password P@ssw0rd.

## Objectives

After completing this lab, you will be able to:

- Set up and manage storage accounts
- Get walkthroughs on how and when to use blobs, tables or queues
- Analyze storage usage to improve application performance and design

## Prerequisites

Before working on this lab, you must have:

- Azure account

## Overview of the lab

Azure provides different mechanisms to store your data. Each mechanism has a proper use and a recommended scenario. How to choose the right storage service by scenario? How to work with each different storage service on Azure?

This lab is a guidance for building Windows Azure data management services into your applications. Learn to choose the correct data service for each scenario and how to write applications for resiliency and robustness in the cloud with blobs, tables and queues.

## Scenario

Cloud computing enables new scenarios for applications requiring scalable, durable, and highly available storage for their data. Azure Storage is massively scalable, so you can store and process hundreds of terabytes of data to support the big data scenarios required by scientific, financial analysis, and media applications. Or you can store the small amounts of data required for a small business website. Wherever your needs fall, you pay only for the data you're storing. Azure Storage is elastic, so you can design applications for a large global audience, and scale those applications as needed - both in terms of the amount of data stored and the number of requests made against it. You pay only for what you use, and only when you use it. Azure Storage uses an auto-partitioning system that automatically load-balances your data based on traffic. This means that as the demands on your application grow, Azure Storage automatically allocates the appropriate resources to meet them. Azure Storage supports clients using a diverse set of operating systems (including Windows and Linux) and a variety of programming languages (including .NET, Java, and C++) for convenient development. Azure Storage also exposes data resources via simple REST APIs, which are available to any client capable of sending and receiving data via HTTP/HTTPS. This guide targets the .NET Framework 4.5 (and above) and Azure .NET Storage Client Library 2.x (and above). The recommended version is Storage Client Library 4.x, which is available via NuGet or as part of the Azure SDK for .NET.

## Virtual machine technology

This lab is completed using virtual machines that run on Windows Server® 2012 R2 Hyper-V™ technology. To log on to the virtual machines, press CTRL+ALT+END and enter your logon credentials.

## Computers in this lab

This lab uses computers as described in the following table. Before you begin the lab, you must ensure that the virtual machines are started and then log on to the computers.

Virtual Machine	Role
TR20HOL	TR20Attendee

◇ All user accounts in this lab use the password **P@ssw0rd!**

### Note regarding pre-release software

Portions of this lab include software that is not yet released, and as such may still contain active or known issues. While every effort has been made to ensure this lab functions as written, unknown or unanticipated results may be encountered as a result of using pre-release software.

### Note regarding user account control

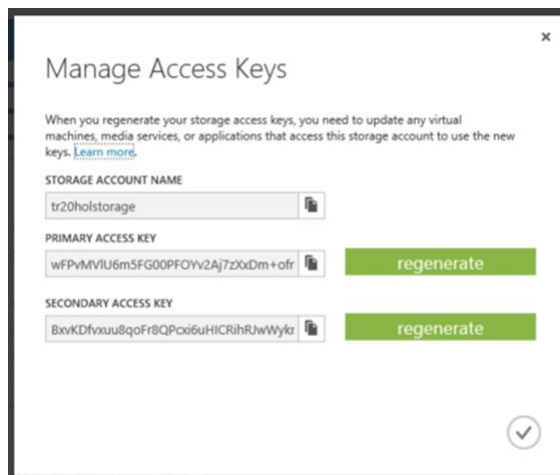
Some steps in this lab may be subject to user account control. User account control is a technology which provides additional security to computers by requesting that users confirm actions that require administrative rights. Tasks that generate a user account control confirmation are denoted using a shield icon. If you encounter a shield icon, confirm your action by selecting the appropriate button in the dialog box that is presented.

## Exercise 1: Setup Azure Account

In this exercise, we are going to setup a new azure account.

### Create a new Azure Storage Account

1. If you do not have a trial Azure pass, please ask a lab proctor.
2. Go to Azure management portal at <https://manage.windowsazure.com>.
3. Click on Storage.
4. Click on +New.
5. Click on "Data Services", after on "Storage" and on "Quick Create".
6. In URL, type a unique Url, such as tr21storage<youralias> (must be all lower case and use numbers and letters only).
7. Select East US for the location.
8. Click at "Create storage account".
9. Wait until processing is completed successfully.
10. Click at Manage Access Keys to see the all enabled access keys. These keys are important they'll be used to access Windows Azure storage resources.



### Setup Visual Studio Solution

Body text style for task overview.

1. Open Visual Studio 2013
2. When prompted, sign in using the Microsoft Account attached to the Azure Pass you were provided with to do the lab.
3. Click at File menu
4. Click at New Project
5. Select Other Project Types

6. Choose Blank Solution
7. Name the solution as “Azure.StorageServices”.
8. Click Ok

Create three solution folders, with the solution opened:

9. Right click over solution folder
10. Click at Add
11. Click at New Solution Folder
12. Name this new solution as “Blob”
13. Repeat the steps creating other two new solution folders named “Queue” and “Table”

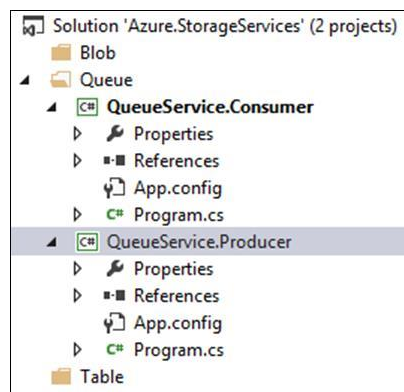
♦ **IMPORTANT:** You do not have to type in the code in the exercises below. All the code has been copied into text files that you can find at `C:\LabFiles` on the virtual machine.

## Queue

Azure Queue storage is a service for storing large numbers of messages that can be accessed from anywhere in the world via authenticated calls using HTTP or HTTPS. A single queue message can be up to 64 KB in size, and a queue can contain millions of messages, up to the total capacity limit of a storage account.

### 1) How to create new projects to consume an Azure Queue

- a. Right click solution folder named “Queue”
- b. Click at Add
- c. Click at New Project
- d. Add a new Console Application named “QueueService.Consumer”
- e. Repeat the steps to create another Console Application named “QueueService.Producer”
- f. Your solution must look like this



g.

### 2) How to add binary references to Windows Azure storage

- a. Open Package Manager Console (View -> Other Windows -> Package Manager Console)
- b. Type “install-package windowsazure.storage -project QueueService.Producer”, press Enter

- c. Wait until Windows Azure storage binaries installation is complete
  - d. Type “install-package windowsazure.storage -project QueueService.Consumer”, press Enter
  - e. Wait until Windows Azure storage binaries installation is complete
- 3) How to post messages into a queue at Windows Azure
- a. Open Program.cs at QueueService.Producer project
  - b. Add using statements as below:

```
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Auth;
using Microsoft.WindowsAzure.Storage.Queue;
```

- c. Add code to Program class to create an instance of Windows Azure cloud storage account

```
public static CloudStorageAccount GetStorageAccount() {

    string accountName = "<data account name>";
    string primaryKey = "<primary key>";

    StorageCredentials credentials = new StorageCredentials(accountName, primaryKey);
    CloudStorageAccount storageAccount = new CloudStorageAccount(credentials, true);

    return storageAccount;
}
```

Replace the "<data account name>" text with the Storage Account Name created on first step.

Replace the "<primary key>" text with the Primary Access Key created on first step.

- d. Add code to create the queue if it does not exist.

```
private static CloudQueue CreateQueueIfNotExists(string queueName) {

    var account = GetStorageAccount();

    CloudQueueClient queueClient = account.CreateCloudQueueClient();
    CloudQueue queue = queueClient.GetQueueReference(queueName);

    bool created = queue.CreateIfNotExists();

    return queue;
}
```

- e. Add code to queue messages.

```
private static void QueueMessage(CloudQueue queue, string message) {

    TimeSpan timeToLive = TimeSpan.FromDays(1);
```

```
queue.AddMessage(new CloudQueueMessage(message), timeToLive);  
}
```

- f. Replace the main method.

```
static void Main(string[] args) {  
  
    const string QUEUENAME = "tr20queue";  
  
    CloudQueue queue = CreateQueueIfNotExists(QUEUENAME);  
  
    while (true) {  
  
        string message = Guid.NewGuid().ToString();  
        QueueMessage(queue, message);  
        Console.WriteLine("Message queued! - " + message);  
    }  
}
```

4) Retrieve messages from the queue at Windows Azure

- a. Open Program.cs at QueueService.Consumer project  
b. Add using statements as below:

```
using Microsoft.WindowsAzure.Storage;  
using Microsoft.WindowsAzure.Storage.Auth;  
using Microsoft.WindowsAzure.Storage.Queue;
```

- c. Add code to create an instance of Windows Azure cloud storage account

```
public static CloudStorageAccount GetStorageAccount() {  
  
    string accountName = "<data account name>";  
    string primaryKey = "<primary key>";  
  
    StorageCredentials credentials = new StorageCredentials(accountName, primaryKey);  
    CloudStorageAccount storageAccount = new CloudStorageAccount(credentials, true);  
  
    return storageAccount;  
}
```

Replace the "<data account name>" text with the Storage Account Name created on first step.

Replace the "<primary key>" text with the Primary Access Key created on first step.

- d. Add code to create the queue if it does not exist.



```
private static CloudQueue CreateQueueIfNotExists(string queueName) {

    var account = GetStorageAccount();

    CloudQueueClient queueClient = account.CreateCloudQueueClient();
    CloudQueue queue = queueClient.GetQueueReference(queueName);

    bool created = queue.CreateIfNotExists();

    return queue;
}
```

- e. Add code to retrieve messages from queue

```
private static void RetrieveMessages(CloudQueue queue) {

    const int maxMessagesToGet = 10;
    var visibilityTimeout = TimeSpan.FromSeconds(10);

    Console.WriteLine("Reading queue...");
    IEnumerable<CloudQueueMessage> messages = queue.GetMessages(maxMessagesToGet,
visibilityTimeout);

    foreach (CloudQueueMessage message in messages) {

        string messageContent = message.AsString;
        Console.WriteLine("Message's content: " + messageContent);
        queue.DeleteMessage(message);
    }
}
```

- f. Replace the main method.

```
static void Main(string[] args) {

    const string QUEUENAME = "tr20queue";

    CloudQueue queue = CreateQueueIfNotExists(QUEUENAME);

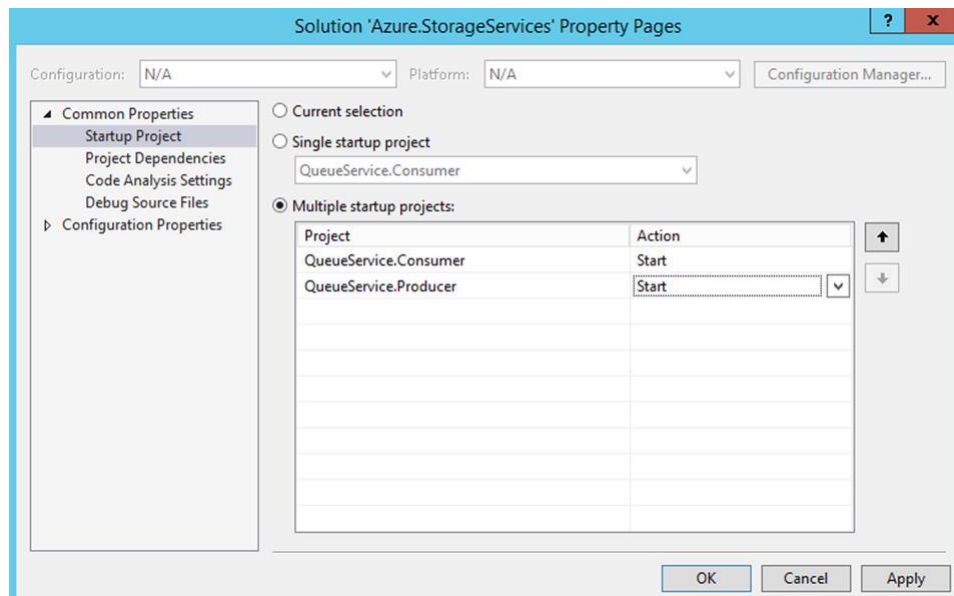
    while (true) {

        RetrieveMessages(queue);
    }
}
```

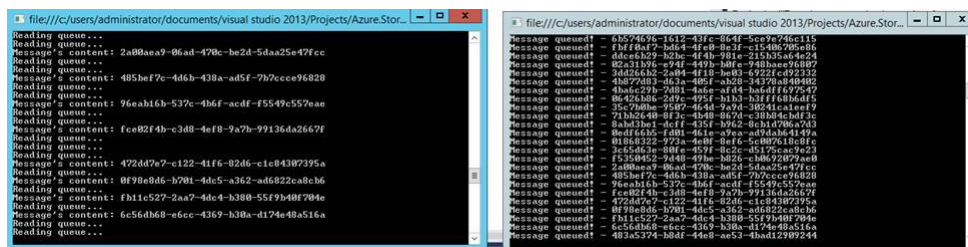
## 5) Testing sample

- a. Right click on the solution
- b. Click at Properties
- c. Select Startup Project tab
- d. Mark Multiple startup projects

- e. Define both QueueService.Consumer and QueueService.Producer projects with Start action:



- f. Click Ok
- g. Press F5 to run your application
- h. It should be like this:



After a few minutes, close the windows.

## Blob

Azure Blob storage is a service for storing large amounts of unstructured data, such as text or binary data, that can be accessed from anywhere in the world via HTTP or HTTPS. You can use Blob storage to expose data publicly to the world, or to store application data privately.

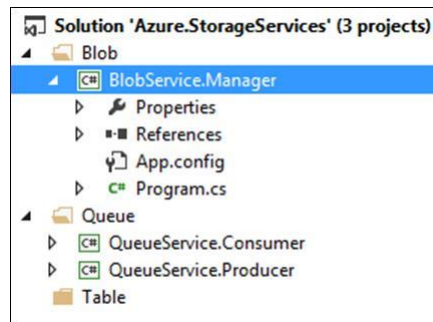
Common uses of Blob storage include:

- Serving images or documents directly to a browser
- Storing files for distributed access
- Streaming video and audio
- Performing secure backup and disaster recovery

- Storing data for analysis by an on-premises or Azure-hosted service

1) How to create new project to consume an Azure Blob

- Right click solution folder named “Blob”
- Click at Add
- Click at New Project
- Add a new Console Application named “BlobService.Manager”
- Your solution might look like this



2) How to add binary references to Windows Azure storage

- Open Package Manager Console (View -> Other Windows -> Package Manager Console)
- Type “install-package windowsazure.storage -project BlobService.Manager”, press Enter
- Wait until Windows Azure storage binaries installation is complete

3) How to upload files into a Windows Azure Blob

- Open Program.cs at BlobService.Manager project
- Add using statements as below:

```
using Microsoft.WindowsAzure.Storage;  
using Microsoft.WindowsAzure.Storage.Auth;  
using Microsoft.WindowsAzure.Storage.Blob;  
using System.IO;  
using System.Diagnostics;
```

- Add code Program class to create an instance of Windows Azure cloud storage account

```
public static CloudStorageAccount GetStorageAccount() {  
  
    string accountName = "<data account name>";  
    string primaryKey = "<primary key>";  
  
    StorageCredentials credentials = new StorageCredentials(accountName, primaryKey);  
}
```

```
CloudStorageAccount storageAccount = new CloudStorageAccount(credentials, true);

return storageAccount;
}
```

Replace the "<data account name>" text with the Storage Account Name created on first step.

Replace the "<primary key>" text with the Primary Access Key created on first step.

- d. Add code to create a blob container if it does not exist.

```
private static CloudBlobContainer GetContainer(string containerName) {

    var account = GetStorageAccount();

    CloudBlobClient blobClient = account.CreateCloudBlobClient();
    CloudBlobContainer container = blobClient.GetContainerReference(containerName);

    bool created = container.CreateIfNotExists();

    return container;
}
```

- e. Add code to upload bytes

```
public static void UploadBytes(CloudBlobContainer container, string reference, byte[] data) {

    CloudBlockBlob blob = container.GetBlockBlobReference(reference);

    using (var content = new MemoryStream(data)) {
        blob.UploadFromStream(content);
    }
}
```

- f. Add code to download bytes

```
public static byte[] DownloadBytes(CloudBlobContainer container, string reference) {

    byte[] returnValue = null;

    CloudBlockBlob blob = container.GetBlockBlobReference(reference);

    if (!blob.Exists())
        throw new System.IndexOutOfRangeException("Blob not found!");

    using (MemoryStream stream = new MemoryStream()) {

        blob.DownloadToStream(stream);
        returnValue = stream.ToArray();
    }
}
```

```
return returnValue;  
}
```


- g. Replace the main method.

```
static void Main(string[] args) {  
  
    const string BLOBCONTAINERNAME = "tr20blob";  
  
    var container = GetContainer(BLOBCONTAINERNAME);  
  
    Console.WriteLine("Uploading file...");  
    UploadBytes(container, "Reference#1",  
File.ReadAllBytes(@"c:\temp\visualstudiologo.png"));  
    Console.WriteLine("File uploaded! Press any key...");  
    Console.ReadKey();  
  
    Console.WriteLine("Downloading file...");  
    var bytes = DownloadBytes(container, "Reference#1");  
    Console.WriteLine("File downloaded! Press any key...");  
    Console.ReadKey();  
  
    Console.WriteLine("Opening file...");  
    File.WriteAllBytes(@"c:\temp\visualstudiologoDownloaded.png", bytes);  
    Process.Start(new ProcessStartInfo(@"c:\temp\visualstudiologoDownloaded.png"));  
}
```

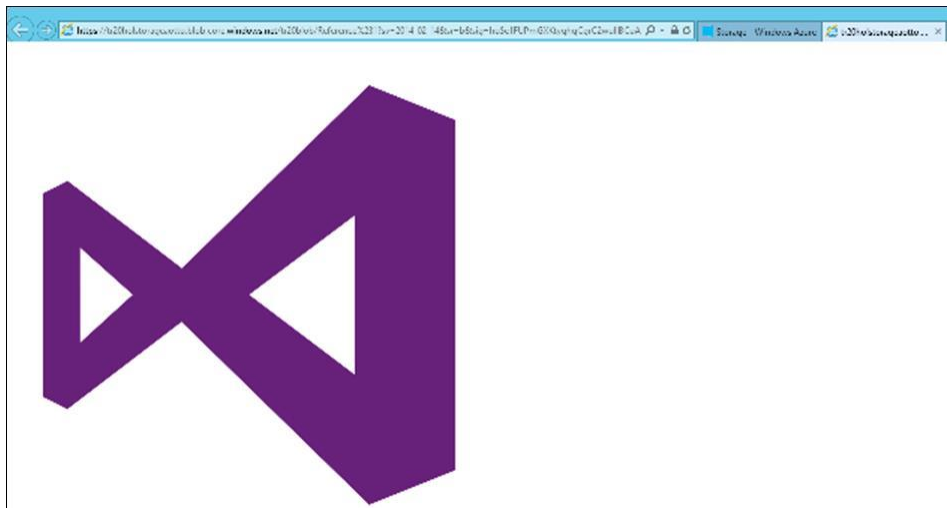
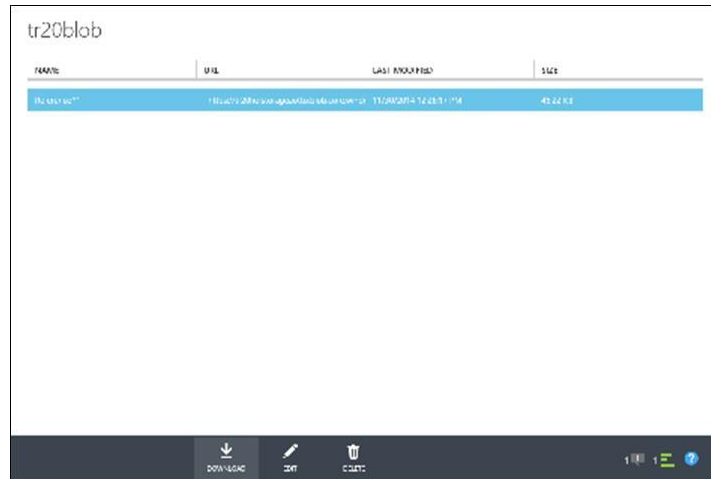
- h. Right click on BlobService.Manager project
- i. Click Set Startup Projects
- j. In the Action column, set BlobService.Manager to Start.
- k. In the Action column, set QueueService.Consumer and QueueService.Producer to None.
- l. Press F5 to run your project

4) Access files at Windows Azure portal

- a. Open Windows Azure portal (<http://manage.windowsazure.com>)
- b. Click at Storage
- c. Select the storage created during the lab
- d. Click at containers

tr20holstorageaotto		
	DASHBOARD	MONITOR
CONFIGURE	CONTAINERS	IMPORT/EXPORT
NAME	URL	LAST MODIFIED
tr20blob	<a href="https://tr20holstorageaotto.blob.core.windows.net/tr20blob">https://tr20holstorageaotto.blob.core.windows.net/tr20blob</a>	11/30/2014 12:26:06 PM

- e. Click at tr20blob to see all uploaded files
- f. Select the item with name Reference#1
- g. Click at Download button



## Table

The Azure Table storage service stores large amounts of structured data. The service is a NoSQL datastore which accepts authenticated calls from inside and outside the Azure cloud. Azure tables are ideal for storing structured, non-relational data. Common uses of the Table service include:

- Storing TBs of structured data capable of serving web scale applications
- Storing datasets that don't require complex joins, foreign keys, or stored procedures and can be denormalized for fast access
- Quickly querying data using a clustered index

- Accessing data using the OData protocol and LINQ queries with WCF Data Service .NET Libraries

You can use the Table service to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

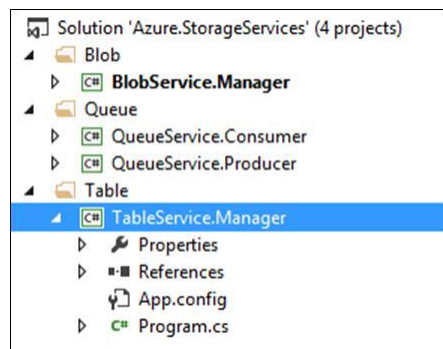
Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties. The number of tables that a storage account can contain is limited only by the storage account capacity limit.

An entity is a set of properties, similar to a database row. An entity can be up to 1MB in size. Each entity can include up to 252 properties to store data.

A property is a name-value pair. Each entity also has 3 system properties that specify a partition key, a row key, and a timestamp. Entities with the same partition key can be queried more quickly, and inserted/updated in atomic operations. An entity's row key is its unique identifier within a partition.

#### 1) How to create a new project to consume data on Azure tables

- Right click solution folder named Table
- Click at Add
- Click at New Project
- Add a new Console Application and name it "TableService.Manager"
- Your solution might look like this



#### 2) Add binary references to Windows Azure storage

- Open Package Manager Console (View->Other Windows->Package Manager Console)
- Type "install-package windowsazure.storage -project TableService.Manager", press Enter
- Wait until Windows Azure storage binaries installation is complete

3) Upload files into a Windows Azure Blob

- a. Add a new class named Lecture
- b. Right-click in the TableService.Manager -> add -> class
- c. In Add New Item – Table Service.Manager, click Visual C# Items, and click Class.
- d. Click Add.
- e. Rename Class1.ca to Lecture.cs.
- f. Open Lecture.cs at TableService.Manager project
- g. Add using statement as below:

```
using Microsoft.WindowsAzure.Storage.Table;
```

- h. Override the original content of Lecture class with the content listed below. Now, Lecture inherits from TableEntity. TableEntity is a base class to represent an object type for a table in the Table service.

```
public class Lecture : TableEntity {  
  
    public string Title { get; set; }  
  
    public string Description { get; set; }  
  
    public Lecture()  
        : base() {  
    }  
}
```

- i. Open Program.cs at TableService.Manager project
- j. Add using statements as below:

```
using Microsoft.WindowsAzure.Storage;  
using Microsoft.WindowsAzure.Storage.Auth;  
using Microsoft.WindowsAzure.Storage.Table;
```

- k. Add code to create an instance of Windows Azure cloud storage account

```
public static CloudStorageAccount GetStorageAccount() {  
  
    string accountName = "<data account name>";  
    string primaryKey = "<primary key>";  
  
    StorageCredentials credentials = new StorageCredentials(accountName, primaryKey);  
    CloudStorageAccount storageAccount = new CloudStorageAccount(credentials, true);  
  
    return storageAccount;  
}
```



Replace the "<data account name>" text with the Storage Account Name created on first step.

Replace the "<primary key>" text with the Primary Access Key created on first step.

- h. Add code to create a table if it does not exist.

```
private static CloudTable GetTable(string tableName) {  
  
    var client = GetStorageAccount().CreateCloudTableClient();  
  
    CloudTable table = client.GetTableReference(tableName);  
    var exists = table.CreateIfNotExists();  
  
    return table;  
}
```

- i. Add code to insert, update, delete and query all entities on Azure table storage.

```
public static void Insert(CloudTable table, Lecture lecture) {  
  
    TableOperation operation = TableOperation.Insert(lecture);  
    TableResult result = table.Execute(operation);  
}  
  
public static void Update(CloudTable table, Lecture lecture) {  
  
    TableOperation operation = TableOperation.Replace(lecture);  
    TableResult result = table.Execute(operation);  
}  
  
public static void Delete(CloudTable table, Lecture lecture) {  
  
    TableOperation operation = TableOperation.Delete(lecture);  
    TableResult result = table.Execute(operation);  
}  
  
public static IEnumerable<Lecture> GetAll(CloudTable table) {  
  
    TableQuery<Lecture> query = new TableQuery<Lecture>();  
    var returnValue = table.ExecuteQuery(query).ToList();  
  
    return returnValue;  
}  
  
public static void Print(Lecture lecture) {  
  
    Console.WriteLine();  
    Console.WriteLine("Lecture structure");  
    Console.WriteLine("PartitionKey: {0}", lecture.PartitionKey);  
    Console.WriteLine("RowKey: {0}", lecture.RowKey);  
    Console.WriteLine("Timestamp: {0}", lecture.Timestamp);  
    Console.WriteLine("ETag: {0}", lecture.ETag);  
}
```

```
Console.WriteLine("Title: {0}", lecture.Title);
Console.WriteLine("Description: {0}", lecture.Description);
Console.WriteLine();
}
```

- m. Override Main method with code to access and manipulate data on Azure table storage.

```
static void Main(string[] args) {

    const string TABLENAME = "tr20table";

    CloudTable table = GetTable(TABLENAME);

    Lecture newLecture = new Lecture() {
        PartitionKey = "Lecture",
        RowKey = Guid.NewGuid().ToString(),
        Title = "Developing Applications with Windows Azure Storage",
        Description = "Get guidance for building Windows Azure data management services into
your Microsoft.NET Applications"
    };
    Console.WriteLine("New instance of Lecture created.");

    Console.WriteLine("Inserting a new Lecture on Azure Table...");
    Insert(table, newLecture);
    Print(newLecture);
    Console.WriteLine("Lecture inserted! Press any key...");
    Console.ReadKey();

    Console.WriteLine("Updating a Lecture on Azure Table...");
    Update(table, newLecture);
    Print(newLecture);
    Console.WriteLine("Lecture updated! Press any key...");
    Console.ReadKey();

    Console.WriteLine("Querying entities at Azure");
    var all = GetAll(table);
    foreach (var item in all) {
        Print(item);
    }
    Console.WriteLine("Press any key...");
    Console.ReadKey();

    Console.WriteLine("Deleting a Lecture on Azure Table...");
    Delete(table, newLecture);
    Console.WriteLine("Lecture deleted! Press any key...");
    Console.ReadKey();
}
```

- n. Right Click the TableService.Manager project
- o. Click at Set as Startup Project
- p. Press F5 to run the project

## References

<http://azure.microsoft.com/en-us/documentation/services/storage/>

<http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-tables/>

<http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/>

<http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-queues/>

Mehner, Paul; Developing Cloud Applications with Windows Azure Storage; First Edition; Microsoft Press; 2013.