**Implement Azure Storage tables**

Azure Storage is a non-relational (NoSQL) entity storage service on Microsoft Azure. When you create a storage account, it includes the Table service alongside the Blob and Queue services. Table services can be accessed through a URL format. It looks like this:

*http://<storage account name>.table.core.windows.net/<table name>.*

There are many forms of NoSQL databases:

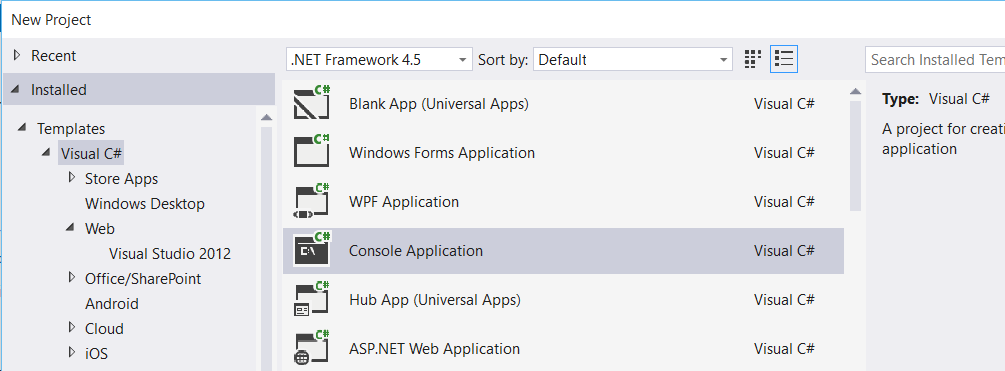
* Key-value stores that organize data with a unique key per record and often allow for jagged entries where each row might not have a complete set of values.
* Document databases that are similar to key-value stores with semi-structured, easy-to-query documents. Usually, information is stored in JavaScript Object Notation (JSON) format.
* Columnar stores that are used to organize large amounts of distributed information.
* Graph databases that do not use columns and rows; instead, they use a graph model for storage and query, usually for large amounts of highly distributed data.

Table storage is a key-value store that uses a partition key to help with scale out distribu­tion of data and a row key for unique access to a particular entry. Together, these keys are used to uniquely identify a record in the account.

**Using basic CRUD operations**

**Creating a table**

1. Create a C# console application.



2. In your app.config file, add an entry under the Configuration element, replacing the account name and key with your own storage account details:

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />

</startup>

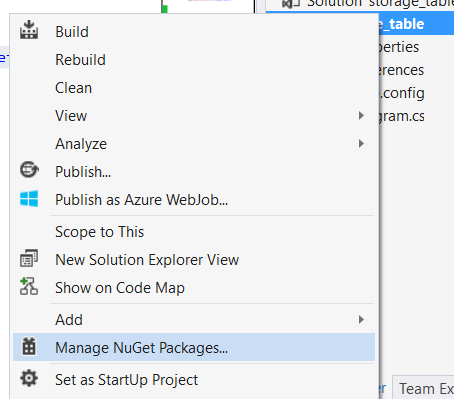
<appSettings>

<add key="StorageConnectionString" value="DefaultEndpointsProtocol=https;AccountName=storagefinal1;AccountKey=tFwfGr4sFCtpmf6KjC8WgQpy1vSDFOvavw9ncK2uv5iJni2T7emxrCm/XeDJddB4J1m6eNzG7JBhPeRfdDQlYg==" />

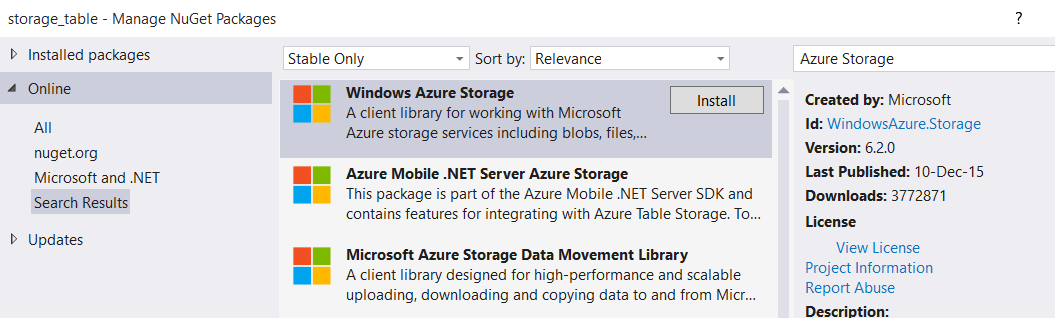
</appSettings>

</configuration>

1. Use NuGet Package Manager to obtain the Microsoft.WindowsAzure.Storage.dll.



Search for “Azure Storage”



1. Add the following using statements to the top of your Program.cs file:

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Auth;

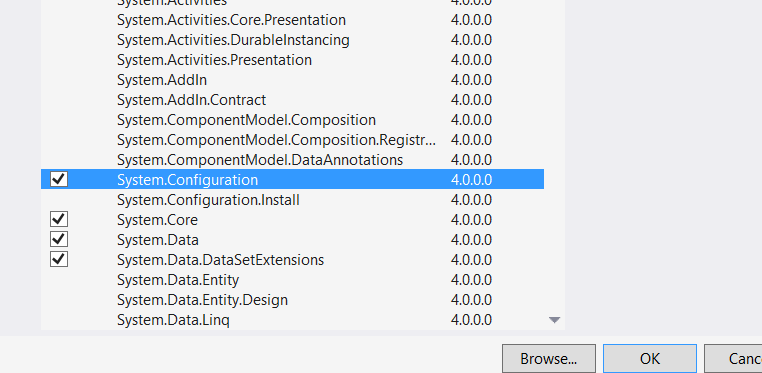
using Microsoft.WindowsAzure.Storage.Table;

using Microsoft.WindowsAzure;

using System.Configuration;

also add one reference

**System.Configuration**



1. Type the following command to retrieve your connection string in the Main function of Program.cs:

static void Main(string[] args)

{

var storageAccount = CloudStorageAccount.Parse(ConfigurationManager.AppSettings["StorageConnectionString"]);

}

1. Use the following command to create a table if one doesn’t already exist:

static void Main(string[] args)

{

var storageAccount = CloudStorageAccount.Parse(ConfigurationManager.AppSettings["StorageConnectionString"]);

CloudTableClient tableClient = storageAccount.CreateCloudTableClient();

CloudTable table = tableClient.GetTableReference("customers");

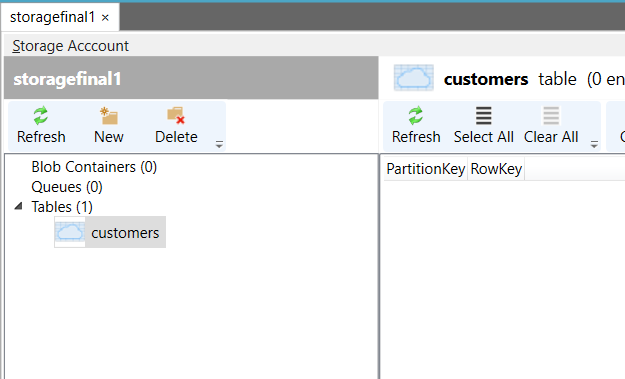
table.CreateIfNotExists();

}

Now run the Visual Studio Program or Press F5.

Run the “Azure Storage Explorer” tool

Enter Storage name & key



**Inserting records**

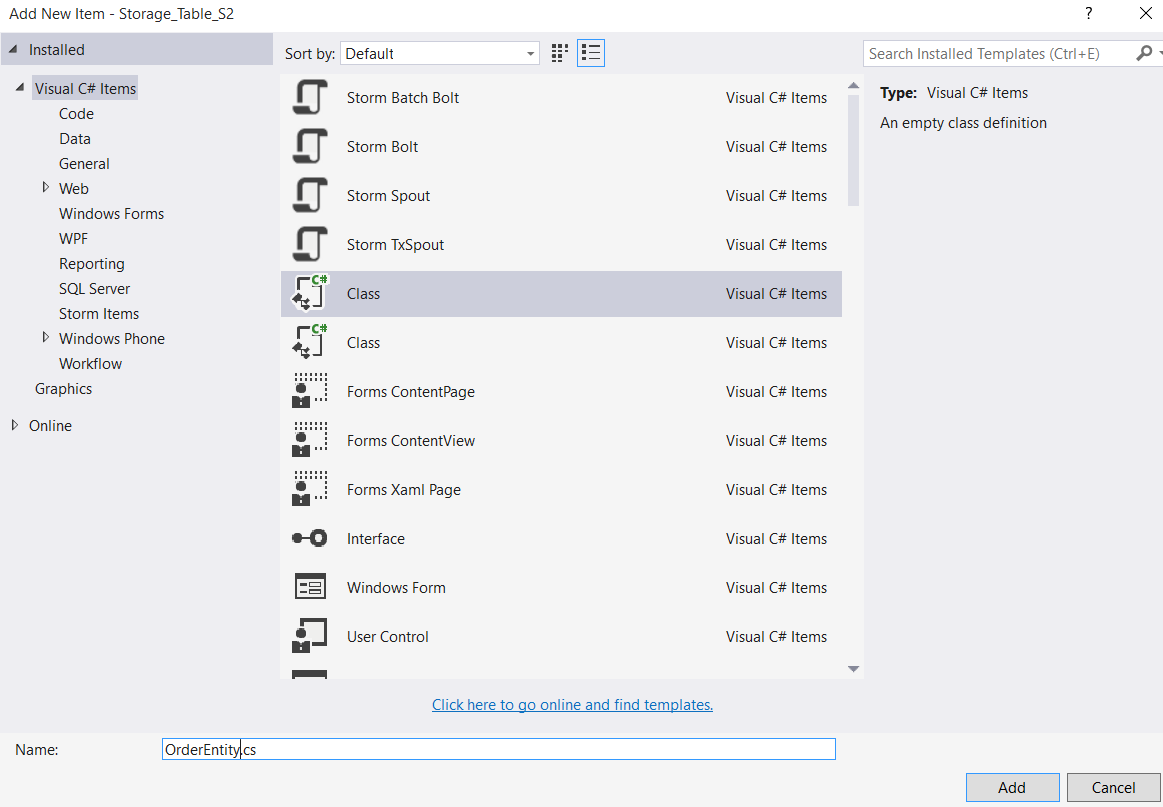
To add entries to a table, you create objects based on the TableEntity base class and serialize them into the table using the Storage Client Library. The following properties are provided for you in this base class:

* **Partition Key** Used to partition data across storage infrastructure
* **Row Key** Unique identifier in a partition
* **Timestamp** Time of last update maintained by Azure Storage
* **ETag** Used internally to provide optimistic concurrency

The combination of partition key and row key must be unique within the table. This combi­nation is used for load balancing and scaling, as well as for querying and sorting entities.

Follow these steps to add code that inserts records:

1. Add a class to your project. Name it as OrderEntity.cs



Open OrderEntity.cs file & add one reference:

using Microsoft.WindowsAzure.Storage.Table;

public class OrderEntity : TableEntity

{

public OrderEntity(string customerName, String orderDate)

{

this.PartitionKey = customerName;

this.RowKey = orderDate;

}

public OrderEntity() { }

public string OrderNumber { get; set; }

public string Status { get; set; }

}

1. Add the following code to the console program to insert a record:

Add references:

using Microsoft.WindowsAzure;

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Auth;

using Microsoft.WindowsAzure.Storage.Table;

public class Program

{

static void Main(string[] args)

{

CloudStorageAccount storageAccount = new CloudStorageAccount(new StorageCredentials("storagefinal1", "tFwfGr4sFCtpmf6KjC8WgQpy1vSDFOvavw9ncK2uv5iJni2T7emxrCm/XeDJddB4J1m6eNzG7JBhPeRfdDQlYg=="), true);

CloudTableClient tableClient = storageAccount.CreateCloudTableClient();

CloudTable table = tableClient.GetTableReference("orders");

table.CreateIfNotExists();

OrderEntity newOrder = new OrderEntity("Archer", "20160119");

newOrder.OrderNumber = "101";

newOrder.Status = "shipped";

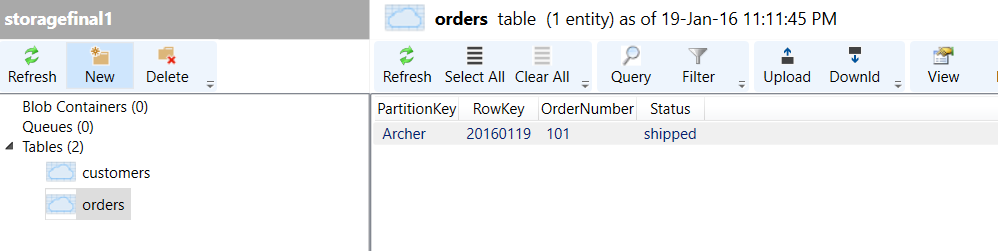
TableOperation insertOperation = TableOperation.Insert(newOrder);

table.Execute(insertOperation);

}

}

Now run the project



**Inserting multiple records in a transaction**

You can group inserts and other operations into a single batch transaction. All operations in the batch must take place on the same partition. You can have up to 100 entities in a batch. The total batch payload size cannot be greater than 4 MB.

class Program

{

static void Main(string[] args)

{

CloudStorageAccount storageAccount = new CloudStorageAccount(new StorageCredentials("storagefinal1", "tFwfGr4sFCtpmf6KjC8WgQpy1vSDFOvavw9ncK2uv5iJni2T7emxrCm/XeDJddB4J1m6eNzG7JBhPeRfdDQlYg=="), true);

CloudTableClient tableClient = storageAccount.CreateCloudTableClient();

CloudTable table = tableClient.GetTableReference("orders");

table.CreateIfNotExists();

TableBatchOperation batchOperation = new TableBatchOperation();

OrderEntity newOrder1 = new OrderEntity("Lana", "20141217");

newOrder1.OrderNumber = "102";

newOrder1.Status = "pending";

OrderEntity newOrder2 = new OrderEntity("Lana", "20141218");

newOrder2.OrderNumber = "103";

newOrder2.Status = "open";

OrderEntity newOrder3 = new OrderEntity("Lana", "20141219");

newOrder3.OrderNumber = "103";

newOrder3.Status = "shipped";

batchOperation.Insert(newOrder1);

batchOperation.Insert(newOrder2);

batchOperation.Insert(newOrder3);

table.ExecuteBatch(batchOperation);

}

}

