Module 6: Storing Tabular Data in Azure

Lab: Storing Event Registration Data in Azure Storage Tables

Exercise 1: Populating the Sign-In Form with Registrant Names

Task 1: Create an instance of the CloudTable class

1. On the Start screen, click the **Desktop** tile.
2. On the taskbar, click the **File Explorer** icon.
3. In the **This PC** window, go to **Allfiles (F):\Mod06\Labfiles\Starter\Contoso.Events**, and then double-click **Contoso.Events.sln.**
4. In the **Solution Explorer** pane, expand the **Roles** solution folder.
5. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
6. Double-click the **TableStorageHelper.cs** file.
7. In the **TableStorageHelper** class, find the method with the following signature:

```
IEnumerable<string> GetRegistrantNames(string eventKey);
```

8. Remove the following single line of code in the class:

```
return Enumerable.Empty<string>();
```

9. At the end of the **GetRegistrantNames** method and before the closing curly braket, create a **CloudTable** instance:

```
CloudTable table = _tableClient.GetTableReference("EventRegistrations");
```

Task 2: Retrieve strongly-typed registration records by partition key

1. At the end of the **GetRegistrantsNames** method and before the closing curly braket, store the **eventKey** in a **string** variable named **partitionKey**:

```
string partitionKey = eventKey;
```

2. Create a **string** filter by using the **TableQuery.GenerateFilterCondition**, as shown below:

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, partitionKey);
```

3. At the end of the **GetRegistrantsNames** method and before the closing curly bracket, create a new instance of the **TableQuery** class and use the fluent **Where** method with your filter to generate a query:

```
TableQuery<Registration> query = new TableQuery<Registration>().Where(filter);
```

4. Pass the generated query into the **ExecuteQuery** method of the table variable by using the **Registration** model class as the generic parameter:

```
IEnumerable<Registration> registrations = table.ExecuteQuery<Registration>(query);
```

Task 3: Use LINQ-to-Objects to project a list of registrant names

1. At the end of the **GetRegistrantsNames** method and before the closing curly bracket, add a statement without the closing semi-colon to store the registrations in a variable of the same type named **names**:

```
IEnumerable<string> names = registrations
```

2. Append the lambda-syntax query with a fluent method to order the result by **LastName**:

```
.OrderBy(r => r.LastName)
```

3. Append the query further with a fluent method to order the result by **FirstName**:

```
.ThenBy(r => r.FirstName)
```

4. Finalize the query with a projection method that uses the **String.Format** static method to format the string with **LastName**, followed by a command, then a space, and then the **FirstName**:

```
.Select(r => String.Format("{1}, {0}", r.FirstName, r.LastName));
```

5. At the end of the **GetRegistrantsNames** method and before the closing curly bracket, add the following statement:

```
return names;
```

> **Results:** After completing this exercise, you will have queried entities by row key or partition key from Table storage.

Exercise 2: Updating the Events Website to use Storage Tables

Task 1: Update the register controller action to store the registration record

1. In the **Solution Explorer** pane, expand the **Roles** solution folder.
2. In the **Solution Explorer** pane, expand the **Contoso.Events.Web** project.
3. In the **Contoso.Events.Web** project, expand the **Controllers** folder.
4. Double-click the **RegisterController.cs** file.
5. In the **RegisterController** class, find the method with the following signature:

```
private Guid StoreRegistration(dynamic registration)
```

6. Remove the single line of code in the class:

```
return Guid.Empty;
```

7. At the end of the **StoreRegistration** method and before the closing curly bracket, get the connection string by using the **ConfigurationManager.AppSettings** property and the **Microsoft.WindowsAzure.Storage.ConnectionString** value as the parameter:

```
string connectionString = ConfigurationManager.AppSettings["Microsoft.WindowsAzure.Storage.ConnectionString"];
```

8. Use the **CloudStorageAccount.Parse** static method with the connection string as the parameter to get the storage account:

```
var storageAccount = Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

9. At the end of the **StoreRegistration** method and before the closing curly bracket, create a *CloudTableClient* variable by using the **CreateCloudTableClient** method of the storage account:

```
var tableClient = storageAccount.CreateCloudTableClient();
```

10. By using the **GetTableReference** method of the *CloudTableClient* variable and **"EventRegistrations"** as the parameter, create a *CloudTable* variable:

```
var table = tableClient.GetTableReference("EventRegistrations");
```

11. At the end of the **StoreRegistration** method and before the closing curly bracket, create a new **TableOperation** by using the **TableOperation.Insert** static method and the dynamic registration as the parameter:

```
var operation = TableOperation.Insert(registration);
```

12. By using the *CloudTable* variable, invoke the **Execute** method by passing the **TableOperation** as the parameter:

```
table.Execute(operation);
```

13. At the end of the **StoreRegistration** method and before the closing curly bracket, parse the **registration.RowKey** string as a **System.Guid** by using the **Guid.Parse** static method :

```
Guid rowKey = Guid.Parse(registration.RowKey);
```

14. Return the *rowKey* variable as the result of the **StoreRegistration** method.

```
return rowKey;
```

Task 2: Update the register ViewModel to retrieve the dynamic stub registration from the table

1. In the **Solution Explorer** pane, expand the **Shared** solution folder.
2. In the **Solution Explorer** pane, expand the **Contoso.Events.ViewModels** project.
3. Double-click the **RegisterViewModel.cs** file.
4. In the **RegisterViewModel** class, locate the method with the following signature:

```
RegisterViewModel(string eventKey)
```

5. At the end of the **RegisterViewModel** constructor and before the closing curly bracket, get the connection string using the **ConfigurationManager.AppSettings** property and the **Microsoft.WindowsAzure.Storage.ConnectionString** value as the parameter:

```
string connectionString = ConfigurationManager.AppSettings["Microsoft.WindowsAzure.Storage.ConnectionString"];
```

6. Use the **CloudStorageAccount.Parse** static method with the connection string as the parameter to get the storage account:

```
var storageAccount = Microsoft.WindowsAzure.Storage.CloudStorageAccount.Parse(connectionString);
```

7. At the end of the **RegisterViewModel** constructor and before the closing curly bracket, create a *CloudTableClient* variable by using the **CreateCloudTableClient** method of the storage account:

```
var tableClient = storageAccount.CreateCloudTableClient();
```

8. Create a *CloudTable* variable by using the **GetTableReference** method of the *CloudTableClient* variable and **"EventRegistrations"** as the parameter:

```
var table = tableClient.GetTableReference("EventRegistrations");
```

9. At the end of the **RegisterViewModel** constructor and before the closing curly bracket, store the **eventKey** in a *string* variable named **partitionKey**:

```
string partitionKey = String.Format("Stub_{0}", this.Event.EventKey);
```

10. Create a **string** filter by using the **TableQuery.GenerateFilterCondition**

```
string filter = TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, partitionKey);
```

11. At the end of the **RegisterViewModel** constructor and before the closing curly bracket, create a new instance of the **TableQuery** class and generate a query by using the fluent **Where** method with your filter :

```
TableQuery query = new TableQuery().Where(filter);
```

12. Pass the generated query into the **ExecuteQuery** method of the table variable:

```
IEnumerable<DynamicTableEntity> tableEntities = table.ExecuteQuery(query);
```

13. At the end of the **RegisterViewModel** constructor and before the closing curly bracket, select the single element in the enumerable of **DynamicTableEntity** objects:

```
DynamicTableEntity tableEntity = tableEntities.SingleOrDefault();
```

14. Set the **RegistrationStub** property to the *DyanmicTableEntity* variable:

```
this.RegistrationStub = DynamicEntity.GenerateDynamicItem(tableEntity);
```

> **Results:** After completing this exercise, you will have used the Azure Storage SDK to retrieve and create entities.

Exercise 3: Verifying that the Events Website is Using Azure Storage Tables for Registrations

Task 1: Create a Storage Account Instance

1. On the Start screen, click the **Internet Explorer** tile.
2. Go to **https://portal.azure.com <https://portal.azure.com>**

3. Enter the email address of your Microsoft account. Click **Continue**.
4. Enter the password for your Microsoft account.
5. Click **Sign In**.
6. In the navigation pane on the left side of the Azure Portal, scroll down, and then click **More Services**.
7. In the **Browse** blade that displays, click **Storage accounts**.
8. In the **Storage accounts** blade that displays, view your list of storage account instances.
9. At the top of the **Storage accounts** blade, click the **Add** button.
10. In the **Create storage account** blade that displays, perform the following steps:

    a. In the **Name** box, provide a globally unique value.

    b. In the **Deployment model** section, ensure that the *Resource manager* option is selected.

    c. In the **Account kind** list, ensure that the *General purpose* option is selected.

    d. In the **Performance** section, ensure that the *Standard* option is selected.

    e. Click on the **Replication** list and select the **Locally-Redundant Storage (LRS)** option.

    f. In the **Storage service encryption** section, ensure that the *Disabled* option is selected.

    g. In the **Resource group** section, select the **Use existing** option.

    h. In the **Resource group** section, locate the dialog box and provide the value **20532**.

    i. In the **Location** list, select the region closest to your current location.

    j. Ensure that the **Pin to dashboard** option is selected.

    k. Click **Create**.
11. Once the **Storage account** instance is created, the blade for the new instance will open automatically.
12. In the **Storage account** blade, record the name of your *storage account*.
13. In the **Settings** section, select the **Access keys** option.
14. In the **Access keys** blade, locate a key that you wish to use.

    > **Note:** you can use any of the keys listed for this lab.
15. For the access key you selected, click the three ellipsis (...) button to the right of the key. Once clicked, select the **View connection string** option.
16. In the **View connection string** dialog, record your connection string for the access key you selected.

    > **Note:** This connection string will be used in various parts of this lab.
    >
    > **Example:** DefaultEndpointsProtocol=https;AccountName={your name here};
    > AccountKey=ODQYiL8AJuqxDYnwA54u88KRHN3JayY/ns+hfjAiBqHXjDd4xQRfIzAYG2SQ9ZJryDLFUD5hSc6Yk8m3L02D2w==;
17. Close the **View connection string** dialog.

Task 2: Run the data generation console project to populate the Azure storage table with data

1. In the **Solution Explorer** pane, expand the **Shared** solution folder.
2. In the **Solution Explorer** pane, expand the **Contoso.Events.Data.Generation** project.

3. Locate and open the **app.config** file in the project.
4. Within the **app.config** file, locate the following configuration setting:

```
<add key="StorageConnectionString" value="UseDevelopmentStorage=true" />
```

5. Update the setting by replacing the value of the **value** attribute (currently *UseDevelopmentStorage=true*) with your *Storage Account*'s connection string.
6. In the **Solution Explorer** pane, right-click the **Contoso.Events.Data.Generation** project, point to **Debug**, and then click **Start New Instance**.
7. Wait for debugging to complete (when the console window closes).

Task 3: Use Cloud Explorer in Visual Studio 2017 to view table storage registrations

1. On the **View** menu, click **Cloud Explorer**.
2. In the **Cloud Explorer** pane, locate the **Storage Accounts** node and click the arrow at the left side to expand the node.
3. If prompted for your account credentials, sign in by using your **Microsoft Account**.
4. Expand the node for your newly created *Storage Account* under the **Storage Accounts** node.
5. Expand the **Tables** node immediately under your *Storage Account*'s node.
6. Double-click the **EventRegistrations** table.
7. In the **EventRegistrations [Table]** tab, scroll through the entities.
8. Drill-down into the properties of a single entity by double-clicking on a row.
9. Exit out of the **Edit Entity** dialog box by clicking the **Cancel** button.
10. Close the **EventRegistrations [Table]** tab in Visual Studio.

Task 4: Debug the web and worker projects to register for the event

1. In the **Solution Explorer** pane, right-click the **Contoso.Events** solution, and then click **Properties**.
2. Navigate to the **Startup Project** section located under the **Common Properties** header.
3. In the **Startup Project** section, locate and select the **Multiple startup projects** option.
4. Within the **Multiple startup projects** table, perform the following actions:

   a. Locate the **Contoso.Events.Web** entry and change it's *Action* from **None** to **Start**.

   b. Locate the **Contoso.Events.Worker** entry and change it's *Action* from **None** to **Start**.
5. Click the **OK** button to close the *Property* dialog.
6. In the **Solution Explorer** pane, expand the **Roles** solution folder.
7. In the **Solution Explorer** pane, expand the **Contoso.Events.Web** project.
8. Locate and open the **web.config** file in the project.
9. Within the **web.config** file, locate the following configuration setting:

```
<add key="Microsoft.WindowsAzure.Storage.ConnectionString" value="UseDevelopmentStorage=true" />
```

10. Update the setting by replacing the value of the **value** attribute (currently *UseDevelopmentStorage=true*) with your *Storage Account*'s connection string.
11. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
12. Locate and open the **app.config** file in the project.
13. Within the **app.config** file, locate the following configuration setting:

```
<add name="AzureWebJobsStorage" connectionString="UseDevelopmentStorage=true" />
```

14. Update the setting by replacing the value of the **connectionString** attribute (currently *UseDevelopmentStorage=true*) with your *Storage Account*'s connection string.
15. Within the **app.config** file, locate the following configuration setting:

```
<add name="AzureWebJobsDashboard" connectionString="UseDevelopmentStorage=true" />
```

16. Update the setting by replacing the value of the **connectionString** attribute (currently *UseDevelopmentStorage=true*) with your *Storage Account*'s connection string.
17. Within the **app.config** file, locate the following configuration setting:

```
<add key="StorageConnectionString" value="UseDevelopmentStorage=true" />
```

18. Update the setting by replacing the value of the **value** attribute (currently *UseDevelopmentStorage=true*) with your *Storage Account*'s connection string.
19. On the **Debug** menu, click **Start Debugging**.
20. On the home page of the web application, verify that it displays a list of events.
21. Click any of the events in the list.
22. On the event web page, click **Register Now**.
23. Fill out all of the fields in the registration form and click **Submit**.
24. Close the tab displaying the website.

Task 5: Use Cloud Explorer in Visual Studio 2017 to view the new table storage registration

1. Switch to the **Contoso.Events – Microsoft Visual Studio** window.
2. On the **View** menu, click **Cloud Explorer**.
3. Locate the **Storage Accounts** node and click the arrow at the left side.
4. If prompted for your account credentials, sign in by using your **Microsoft Account**.
5. Expand the node for your newly created *Storage Account* under the **Storage Accounts** node.
6. Expand the **Tables** node immediately under your *Storage Account*'s node.
7. Double-click the **EventRegistrations** table.
8. In the **EventRegistrations [Table]** tab, scroll through the entities.
9. In the yellow prompt asking if you would like to download the remaining entities, click **click here**.
10. Drill-down into the properties of a single entity by double-clicking on a row.

11. Exit out of the **Edit Entity** dialog box by clicking the **Cancel** button.

12. Click **Timestamp** header twice to sort entities in a descending order by their Timestamp.

13. Locate your new entity at the top of the table.

14. Switch to the **Contoso.Events – Microsoft Visual Studio** window.

15. Close **Contoso.Events – Microsoft Visual Studio**.

> **Results:** After completing this exercise, you will have used Visual Studio and the Azure to create a comprehensive development environment for Azure Storage.