Module 7: Storing and Consuming Files from Azure Storage

Lab: Storing Generated Documents in Azure Storage Blobs

Exercise 1: Implementing Azure Storage Blobs

Task 1: Sign in to the Azure Portal

1. On the Start screen, click the **Internet Explorer** tile.
2. Go to **https://portal.azure.com <https://portal.azure.com>**
3. Type the email address of your Microsoft account.
4. Click **Continue**.
5. Type the password for your Microsoft account.
6. Click **Sign In**.

Task 2: Create a container by using the Portal

1. In the navigation pane on the left side of the screen, scroll down, and then click **More Services**.
2. In the **Browse** blade that displays, click **Storage accounts**.
3. In the **Storage accounts** blade that displays, view the list of Storage instances.
4. In the list of Storage instances, locate the storage account with the prefix **stor20532**.
5. Click the name of the storage account to go to its dashboard.
6. In the **stor20532*[Your Name Here]*** blade that displays, click the **Blobs** tile.
7. In the **Blob service** blade that displays, view the list of your containers.
8. At the top of the blade, click the **Container** button.
9. In the **New container** blade that displays, perform the following steps:

   a. In the **Name** box, type **example**.

   b. In the **Access Type** list, select **Container**.

   c. Click the **Create** button to create your container.

Task 3: Obtain the Storage Account connection string

1. In the **Storage account** blade, record the name of your *storage account*.
2. Click the **Settings** button at the top of the blade.
3. In the **Settings** section, select the **Access keys** option.
4. 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.

5. 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.
6. 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 storage account};AccountKey=ODQYiL8AJuqxDYnwA54u88KRHN3JayY/ns+hfjAiBqHXjDd4xQRfIzAYG2SQ9ZJryDLFUD5hSc6Yk8m3L02D2w==;

Task 4: Add and access blob files in your container

1. On the Start screen, locate and click the **Visual Studio 2017** tile.

   > **Note:** You might have to use the down arrow to locate the Visual Studio 2017 tile on your Start screen.

2. On the **View** menu, click **Cloud Explorer**.

3. Expand the **Storage Accounts** node.

   > **Note:** If you have not previously indicated that you want Visual Studio to remember your credentials, you will be prompted to enter your Microsoft account username and password to continue.

4. Expand the **stor20532*[Your Name]*** account node under the **Storage** node.

5. Expand the **Blob Containers** node under your storage account's node.

6. Double-click **example**.

7. In the **example [Container]** tab, click the *Upload Blob* button.

   > **Note:** The icon on the upload button includes an arrow that is pointing upward to a horizontal line.

8. In the **Upload New File** dialog box, perform the following steps:

   a. Click the **Browse** button.

   b. Go to the **(F):\Mod07\LabFiles\Starter** directory.

   c. Click the **samplefile** text document.

   d. Click **Open**.

   e. Leave the **folder** option set to it's default (blank) value.

   e. Click the **OK** button to complete the dialog.

9. Switch to the **Internet Explorer** window.

10. In a new tab, type the following URL by replacing **[*storage account*]** with the name of your storage account:

    ```
    https://[storage account].blob.core.windows.net/example/samplefile.txt
    ```

    > **Note:** In Visual Studio, you can also right click the samplefile.txt file and select **Copy URL** and paste it in your new tab.

11. Verify that the text **This is your sample file!** displays in the browser.

    > **Results:** After completing this exercise, you will have created a blob container by using the Portal and viewed the blobs in the container.

Exercise 2: Populating the Container with Files and Media

Task 1: Open the blob helper in the Web App worker project

1. On the Start screen, click the **Desktop**.

2. On the taskbar, click **File Explorer**.

3. In the Libraries window, go to **Allfiles (F):\Mod07\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 **BlobStorageHelper.cs** file.

Task 2: Add Word documents to the container after they are created

1. In the **BlobStorageHelper** class, find the method with the following signature:

```
public Uri CreateBlob(MemoryStream stream, string eventKey)
```

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

```
return null;
```

3. At the end of the **CreateBlob** method and before the closing curly bracket, create a new **CloudBlobContainer** for the **signin** container.

```
CloudBlobContainer container = _blobClient.GetContainerReference("signin");
```

4. Call the **CreateIfNotExists** method of the *CloudBlobContainer* variable to ensure that the container exists**:**

```
container.CreateIfNotExists();
```

5. At the end of the **CreateBlob** method and before the closing curly bracket, create a new variable named *blobName*:

```
string blobName;
```

6. Use the **String.Format** static method to create a string, and then assign the string to the *blobName* variable:

```
blobName = String.Format("{0}_SignIn_{1:ddmmyyyss}.docx", eventKey, DateTime.UtcNow);
```

7. At the end of the **CreateBlob** method and before the closing curly bracket, create a block blob reference by using the **GetBlockBlobReference** method and the *blobName* variable as the parameter:

```
ICloudBlob blob = container.GetBlockBlobReference(blobName);
```

8. Use the **Seek** method of the *MemoryStream* variable to set the position of the stream to the beginning and offset the position by the value of **0**:

```
stream.Seek(0, SeekOrigin.Begin);
```

9. Use the **UploadFromStream** method of the **ICloudBlob** interface to upload the stream to the referenced blob:

```
blob.UploadFromStream(stream);
```

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

```
return blob.Uri;
```

> **Results:** After completing this exercise, you will have used the Azure Storage SDK to manage blobs and containers in your storage account.

Exercise 3: Retrieving Files and Media from the Container

Task 1: Download documents from blob storage and stream to the client

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 **DownloadViewModel.cs** file.
4. In the **DownloadViewModel** class, find the method with the following signature:

```
public async Task<DownloadPayload> GetStream()
```

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

```
return await Task.FromResult<DownloadPayload>(null);
```

6. At the end of the **GetStream** method and before the closing curly bracket, create a new **CloudBlobClient** variable for the *_storageAccount* variable:

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

7. Create a new **CloudBlobContainer** instance for the **signin** container by using the *CloudBlobClient* variable:

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

8. Call the **CreateIfNotExists** method of the *CloudBlobContainer* variable to ensure that the container exists:

```
container.CreateIfNotExists();
```

9. At the end of the **GetStream** method and before the closing curly bracket, create a block blob reference by using the **GetBlockBlobReference** method and the *_blobId* variable as the parameter:

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

10. Use the **OpenReadAsync** method of the *ICloudBlob* variable to create a **Stream** and store it in a variable:

```
Stream blobStream = await blob.OpenReadAsync();
```

11. At the end of the **GetStream** method and before the closing curly bracket, create a new instance of the **DownloadPayload** class:

```
DownloadPayload payload = new DownloadPayload();
```

12. Assign the **Stream** variable to the **DownloadPayload** variable's **Stream** property:

```
payload.Stream = blobStream;
```

13. Assign the *ICloudBlob* variable's **Properties.ContentType** value to the *DownloadPayload* variable's **ContentType** property:

```
payload.ContentType = blob.Properties.ContentType;
```

14. Return the *DownloadPayload* variable:

```
return payload;
```

Task 2: Generate the Test 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: Download generated sign-in sheets from the blob storage

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.Management** entry and change it's *Action* from **None** to **Start**.

   c. 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 **Administration** solution folder.
7. In the **Solution Explorer** pane, expand the **Contoso.Events.Management** 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 **Roles** solution folder.
12. In the **Solution Explorer** pane, expand the **Contoso.Events.Web** project.
13. Locate and open the **web.config** file in the project.
14. Within the **web.config** file, locate the following configuration setting:

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

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

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

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

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

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

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

23. Update the setting by replacing the value of the **value** attribute (currently *UseDevelopmentStorage=true*) with your *Storage Account*'s connection string.
24. On the **Debug** menu, click **Start Debugging**.
25. On the home page for the **Contoso Events Administration** web application, click the button to view the list of events.
26. Click the **Sign-In Sheet** button for any event in the list.
27. View the sign-in page that notifies you that a sign-in sheet is being generated.
28. Wait for one to two minutes, and then refresh the sign-in sheet page.
29. Click **Sign-In Sheet** to download the sign-in sheet from the server.

> **Results:** After completing this exercise, you will have downloaded blobs from your storage account by using the Azure Storage SDK.

Exercise 4: Specifying Permissions for the Container

Task 1: Modify Container Access using Cloud Explorer

1. On the desktop, click the **Contoso.Events - Visual Studio 2017** window.
2. On the **Debug** menu, click **Stop Debugging**.
3. On the **View** menu, click **Cloud Explorer**.
4. Expand the **Storage Accounts** node.
5. Expand the node for the storage account used in this lab under the **Storage Accounts** node.
6. Expand the **Blob Containers** node under the node **stor20532*[Your Name Here]***.
7. Select the **signin** container, and then click **Properties Tab** below.
8. In the **Properties** pane, locate **Public Read Access**.
9. Ensure that the value of the **Public Read Access** property is set to **Off**.

Task 2: Generate temporary SAS tokens by using the SDK

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 **DownloadViewModel.cs** file.
4. In the **DownloadViewModel** class, find the method with the following signature:

```
public async Task<string> GetSecureUrl()
```

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

```
return await Task.FromResult<string>(String.Empty);
```

6. At the end of the **GetSecureUrl** method and before the closing curly bracket, create a new **CloudBlobClient** for the ***_storageAccount*** variable.

```
CloudBlobClient blobClient = _storageAccount.CreateCloudBlobClient();
```

7. Create a new **CloudBlobContainer** for the **signin** container by using the *CloudBlobClient* variable.

```
CloudBlobContainer container = blobClient.GetContainerReference("signin");
```

8. Call the **CreateIfNotExists** method of the *CloudBlobContainer* variable to ensure that the container exists**:**

```
container.CreateIfNotExists();
```

9. At the end of the **GetSecureUrl** method and before the closing curly bracket, create a new instance of the **SharedAccessBlobPolicy** class:

```
SharedAccessBlobPolicy blobPolicy = new SharedAccessBlobPolicy();
```

10. Set the *SharedAccessBlobPolicy* variable's **SharedAccessExpiryTime** property to 15 minutes from the current time:

```
blobPolicy.SharedAccessExpiryTime = DateTime.Now.AddHours(0.25d);
```

11. Set the *SharedAccessBlobPolicy* variable's **Permissions** property to **SharedAccessBlobPermissions.Read**:

```
blobPolicy.Permissions = SharedAccessBlobPermissions.Read;
```

12. At the end of the **GetSecureUrl** method and before the closing curly bracket, create a new instance of the **BlobContainerPermissions** class:

```
BlobContainerPermissions blobPermissions = new BlobContainerPermissions();
```

13. Add the newly created **SharedAccessBlobPolicy** to the *BlobContainerPermissions* variable's **SharedAccessPolicy** with the key **"ReadBlobPolicy"**:

```
blobPermissions.SharedAccessPolicies.Add("ReadBlobPolicy", blobPolicy);
```

14. Set the *BlobContainerPermissions* variable's **PublicAccess** property to **BlobContainerPublicAccessType.Off**:

```
blobPermissions.PublicAccess = BlobContainerPublicAccessType.Off;
```

15. At the end of the **GetSecureUrl** method and before the closing curly bracket, call the asynchronous **SetPermissionsAsync** method of the *CloudBlobContainer* variable by using the *BlobContainerPermissions* variable as the parameter:

```
await container.SetPermissionsAsync(blobPermissions);
```

16. Invoke the **GetSharedAccessSignature** method of the *CloudBlobContainer* variable by using a new instance of the **SharedAccessBlobPolicy** class as the first parameter and the **"ReadBlobPolicy"** key as the second parameter:

```
string sasToken = container.GetSharedAccessSignature(new   SharedAccessBlobPolicy(), "ReadBlobPolicy");
```

17. At the end of the **GetSecureUrl** method and before the closing curly bracket, create a block blob reference by using the **GetBlockBlobReference** method and the ***_blobId*** variable as the parameter:

```
ICloudBlob blob = container.GetBlockBlobReference(_blobId);
```

18. Take the **Uri** property of the *ICloudBlob* variable and store it in a new *Uri* variable.

```
Uri blobUrl = blob.Uri;
```

19. At the end of the **GetSecureUrl** method and before the closing curly bracket, concatenate the **AbsoluteUri** of the *Uri* variable and the *sasToken* variable:

```
string secureUrl = blobUrl.AbsoluteUri + sasToken;
```

20. Return the string variable as the result of the method:

```
return secureUrl;
```

Task 3: Download documents from a protected container by using the SAS token

1. On the **Debug** menu, click **Start Debugging**.
2. On the home page for the **Contoso Events Administration** web application, click the button to view the list of events.
3. Click the **Sign-In Sheet** button for any event in the list.
4. View the sign-in page which notifies you that a sign-in sheet is being generated.
5. Wait for one or two minutes, and then refresh the sign-in sheet page.
6. Click **Sign-In Sheet** to download the sign-in sheet from the server by using the blob Url.
7. Close the **Internet Explorer** application.
8. Close the **Contoso.Events - Visual Studio** application.

> **Results:** After completing this exercise, you will have modified the permissions of the containers and generated SAS tokens for the containers.