Module 8: Designing a Communication Strategy by Using Queues and Service Bus
Lab: Using Queues and Service Bus to Manage Communication Between Web Applications in Azure

Exercise 1: Creating an Azure Service Bus Namespace

Task 1: Create the Service Bus namespace by using the Portal

Note: Service Bus functionality is not available yet in the new Portal. Because of this, the Classic Portal will be used for this lab.

1. On the Start screen, click the **Internet Explorer** tile.
2. Go to *https://manage.windowsazure.com*
3. In the email address box, type the email address of your Microsoft account.
4. In the password box, type the password for your Microsoft account.
5. Click **Sign In** In the navigation pane on the left side
6. In the navigation pane on the left side of the screen, scroll down, and then click **Service Bus**.
7. At the bottom of the screen, click the **Create** button.
8. In the **Create a Namespace** dialog box, perform the following steps:

   a. In the **Namespace Name** box, type **sb20532[*Your Name*]**.

   b. In the **Region** list, select the region that is closest to your location.

   c. In the **Type** list, select the **Messaging** option.

   d. In the **Messaging Tier** list, select the **Standard** option.

   e. Click the check mark button to create your namespace.

Note: It takes approximately 1-2 minutes to create your Service Bus namespace instance.

9. In the list of **Service Bus** namespaces, click the namespace that you just created.
10. At the bottom of the screen, click **Connection Information**.
11. Record the *RootManageSharedAccessKey* connection string from the **Access connection information** dialog box.

Note: You must record a connection string from the list of SAS items.

12. Close the **Access connection information** dialog box.
13. At the top of the screen, click the **Queues** tab.
14. At the bottom-left corner of the screen, click **New**.
15. If it is not automatically selected, select **App Services > Service Bus > Queue > Custom Create**.
16. In the **Create a Queue** dialog box, perform the following steps:

    a. In the **Queue Name** box, type **signin**.

    b. In the **Region** list, select the same region that you selected for the namespace.

    c. In the **Namespace** box, provide the value **sb20532[Your Name]**.

    d. Click next arrow to move to the next step in the wizard.

    e. Leave all fields as their default values.

    f. Click the check mark button to create the new queue.

Results: After completing this exercise, you will have created a Service Bus namespace and queue by using the Portal.

Exercise 2: Using Azure Queue Storage for Document Generation

Task 1: Update worker role to consume requests from the queue

1. On the Start screen, click **Desktop**.
2. On the taskbar, click the **File Explorer** icon.
3. In the Libraries window, go to **Allfiles (F):\Mod08\Labfiles\Starter\Contoso.Events**, and then double-click **Contoso.Events.sln.**
4. In the **Solution Explorer** pane, expand the **Roles** folder.
5. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
6. Double-click the **TableStorageQueueHelper.cs** file.
7. Add a using statement for the **System.Configuration** namespace to the top of the file:

   using System.Configuration;

8. At the end of the **TableStorageQueueHelper** constructor and before the closing curly bracket, store the **StorageAccount** property from the base class in a *CloudStorageAccount* variable:

   CloudStorageAccount storageAccount = base.StorageAccount;

9. Invoke the **CreateCloudQueueClient** method and assign the result to the *_queueClient* variable:

_queueClient = storageAccount.CreateCloudQueueClient();

10. Invoke the static **ConfigurationManager.AppSettings** property and assign the result to the *_signInQueueName* variable:

    _signInQueueName = ConfigurationManager.AppSettings["SignInQueueName"];

11. In the **TableStorageQueueHelper** class, find the method with the following signature:

    IQueueMessage<CloudQueueMessage> Receive()

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

    return new TableStorageQueueMessage(null);

13. At the end of the **Receive** method and before the closing curly bracket, create a new instance of the **CloudQueue** class by calling the **GetQueueReference** method of the *CloudQueueClient* variable by using the **string** name of the queue, as shown in the following code:

    CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);

14. Invoke the **CreateIfNotExists** method to ensure that the queue exists.

    queue.CreateIfNotExists();

15. At the end of the **Receive** method and before the closing curly bracket, invoke the **GetMessage** method of the **CloudQueue** class and store the result in a *CloudQueueMessage* variable, as shown in the following code:

    CloudQueueMessage message = queue.GetMessage();

16. Pass the *CloudQueueMessage* variable into the constructor of the **TableStorageQueueMessage** class and return the result:

    return new TableStorageQueueMessage(message);

17. At the end of the **CompleteMessage** method and before the closing curly bracket, create a new instance of the **CloudQueue** class by calling the **GetQueueReference** method of the *CloudQueueClient* variable by using the **string** name of the queue, as shown in the following code:

    CloudQueue queue = _queueClient.GetQueueReference(_signInQueueName);

18. Invoke the **CreateIfNotExists** method to ensure that the queue exists:

    queue.CreateIfNotExists();

19. At the end of the **CompleteMessage** method and before the closing curly bracket, invoke the **DeleteMessage** method by using the *CloudQueueMessage* variable as the parameter, as shown in the following code:

    queue.DeleteMessage(message);

Task 2: Update administration application to add requests to the queue

1. In the **Solution Explorer** pane, expand the **Shared** folder.
2. In the **Solution Explorer** pane, expand the **Contoso.Events.ViewModels** project.
3. Double-click the **SignInSheetViewModel.cs** file.
4. At the beginning of the **GenerateSignInSheetTableStorage** method and after the opening curly bracket, create a **CloudStorageAccount** instance by using the static **CloudStorageAccount. Parse** method and the table storage connection string, as shown in the following code:

CloudStorageAccount storageAccount = CloudStorageAccount.Parse(tableStorageConnectionString);

5. Create a new instance of the **CloudQueueClient** class by using the **CreateCloudQueueClient** method of the *CloudStorageAccount* variable:

   CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

6. After the above mentioned code, create a new instance of the **CloudQueue** class by invoking the **GetQueueReference** method of the *CloudQueueClient* variable by using the queue name string variable, as shown in the following code.:

   CloudQueue queue = queueClient.GetQueueReference(signInQueueName);

7. Invoke the **CreateIfNotExists** method of the **CloudQueue** class to ensure that the queue exists:

   queue.CreateIfNotExists();

8. After the above mentioned code, create a new instance of the **CloudQueueMessage** class by passing in the string **message** into the constructor:

   CloudQueueMessage queueMessage = new CloudQueueMessage(message);

9. Invoke the **AddMessage** method of the *CloudQueue* variable by using the **CloudQueueMessage** as the parameter:

   queue.AddMessage(queueMessage);

Task 3: Create a Storage Account Instance

1. On the Start screen, click the **Internet Explorer** tile.
2. Go to 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 (LRS)** option.

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

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

j. 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. Click the **Settings** button at the top of the blade.
14. In the **Settings** section, select the **Access keys** option.
15. 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.

16. 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.
17. 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.

18. Close the **View connection string** dialog.

Task 4: 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 5: Debug and verify the application

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**.

   d. Ensure that all the remaining projects have their **Action** set to **None**.

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. Within the **app.config** file, locate the following configuration setting:

    <add name="AzureWebJobsServiceBus" connectionString="Endpoint=sb://[yourServiceNamespace].servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=[yourKey]"/>

25. Update the setting by replacing the value of the **connectionString** attribute (currently *Endpoint=sb://[yourServiceNamespace].servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=[yourKey]*) with your *Service Bus*'s connection string.
26. On the **Debug** menu, click **Start Debugging**.
27. On the desktop, click the **Contoso.Events – Microsoft Visual Studio** window.
28. Click the **View** menu and select the **Solution Explorer** option.
29. In the **Solution Explorer** pane, expand the **Roles** folder.
30. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
31. Double-click the **Functions.cs** file.
32. Locate the **ProcessQueueMessage** method.
33. Locate the target line of code within the try-catch block:

    HandleMessage(message);

34. Right-click the target line of code, point to **Breakpoint**, and click **Insert Breakpoint**.
35. On the desktop, click the **Home - Contoso.Events.Administration** browser window.
36. On the home page of the **Contoso Events Administration** web application, click the **Events** button to go to the list of events.
37. Click **Sign-In Sheet** for any event in the list.
38. View the sign-in page which notifies you that the sign-in sheet is being generated with the following message: **Sign-In Document Generation in Progress**.
39. Wait for one minute for the worker role to receive the queue message.
40. Verify that the application temporarily pauses execution at the breakpoint.
41. Press *F5* to resume execution of the application.
42. Wait for one minute, and then refresh the sign-in sheet page.
43. Click **Sign-In Sheet** to download the sign-in sheet from the server.
44. Close the **Internet Explorer** application.

Results: After completing this exercise, you will have created and consumed messages from Storage queues.

Exercise 3: Using Service Bus Queues for Document Generation

Task 1: Update worker role to consume requests from the queue

1. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
2. Double-click the **app.config** file.
3. Locate the **Setting** element with the name **Microsoft.ServiceBus.ConnectionString**.
4. Replace the value with your previously recorded connection string.
5. In the **Solution Explorer** pane, expand the **Roles** folder.
6. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
7. Double-click the **ServiceBusQueueHelper.cs** file.
8. Add a using statement for the **System.Configuration** namespace to the top of the file:

   ```
   using System.Configuration;
   ```

9. At the end of the **ServiceBusQueueHelper** constructor and before the closing curly bracket, store the **Microsoft**.**ServiceBus.ConnectionString** setting value from your configuration in a string variable, as shown in the following code:

   ```
   string serviceBusConnectionString = ConfigurationManager.AppSettings["Microsoft.ServiceBus.ConnectionString"];
   ```

10. Store the queue name in a string variable.

    ```
    string signInQueueName = ConfigurationManager.AppSettings["SignInQueueName"];
    ```

11. Invoke the static **QueueClient.CreateFromConnectionString** method using the queue name and connection string as parameters, and assign the result to the *_client* variable, as shown in the following code:

    ```
    _client = QueueClient.CreateFromConnectionString(serviceBusConnectionString, signInQueueName);
    ```

12. In the **ServiceBusQueueHelper** class, find the method with the following signature:

    ```
    IQueueMessage<BrokeredMessage> Receive()
    ```

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

    ```
    return new ServiceBusQueueMessage(null);
    ```

14. At the end of the **Receive** method and before the closing curly bracket, create a new instance of the **CloudQueue** class by calling the **GetQueueReference** method of the *CloudQueueClient* variable using the **string** name of the queue, as shown in the following code:

    ```
    BrokeredMessage message = _client.Receive();
    ```

15. Invoke the **CreateIfNotExists** method to ensure that the queue exists

    ```
    return new ServiceBusQueueMessage(message);
    ```

16. At the end of the **CompleteMessage** method and before the closing curly bracket, invoke the **Complete** method on the **message** parameter, as shown in the following code:

    ```
    message.Complete();
    ```

17. At the end of the **AbandonMessage** method and before the closing curly bracket, invoke the **Abandon** method on the **message** parameter, as shown below:

message.Abandon();

18. On the **View** menu, point to **Other Windows**, and then click **Package Manager Console**.

    a. In the **Package Manager Console** pane, in the *Default Project* list, select **Contoso.Events.Worker**.

    b. In the **Package Manager Console** text area, place the cursor after the text **PM**, and then type the following command:

    ```
    Install-Package Microsoft.Azure.WebJobs.ServiceBus -Version 1.1.2
    ```

    c. Press Enter.

19. In the **Solution Explorer** pane, expand the **Roles** folder.
20. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
21. Double-click the **Functions.cs** file.
22. Locate the **ProcessQueueMessage** method.

    public static void ProcessQueueMessage([QueueTrigger("signin")] QueueMessage message, TextWriter log)

23. Update the **ProcessQueueMessage** method by changing the parameter attribute of type **QueueTrigger** to type **ServiceBusTrigger**.

    public static void ProcessQueueMessage([ServiceBusTrigger("signin")] QueueMessage message, Text Writer log)

24. In the **Solution Explorer** pane, expand the **Roles** folder.
25. In the **Solution Explorer** pane, expand the **Contoso.Events.Worker** project.
26. Double-click the **Program.cs** file.
27. At the beginning of the **Main** method and after the opening curly bracket, create a new instance of the **JobHostConfiguration** class as shown below:

    JobHostConfiguration config = new JobHostConfiguration();

28. At the beginning of the **Main** method and after the opening curly bracket, enable the **Service Bus** extension as shown below:

    config.UseServiceBus();

29. At the beginning of the **Main** method and after the opening curly bracket, locate the initialization of the **JobHost** instance as shown below:

    var host = new JobHost();

30. Replace the line of code with the following line of code that updates the initialization of the **JobHost** instance by passing in the **JobHostConfiguration** as a constructor parameter:

    var host = new JobHost(config);

Task 2: Update administration application to add requests to the queue

1. In the **Solution Explorer** pane, expand the **Administration** folder and then expand the **Contoso.Events.Management** project.

2. Double-click the **Web.config** file.
3. Locate the **appSettings** element.
4. Locate the **add** element with the key **Microsoft.ServiceBus.ConnectionString**.
5. Replace the value with your previously recorded connection string.
6. In the **Solution Explorer** pane, expand the **Shared** folder.
7. In the **Solution Explorer** pane, expand the **Contoso.Events.ViewModels** project.
8. Double-click the **SignInSheetViewModel.cs** file.
9. In the constructor, locate the following line of code:

   GenerateSignInSheetTableStorage(context, eventItem, messageString);

10. Replace the above line of code with the following line of code:

    GenerateSignInSheetServiceBus(context, eventItem, message);

11. At the beginning of the **GenerateSignInSheetServiceBus** method and after the opening curly bracket, create a **QueueClient** instance by using the connection string, as shown in the following code:

    QueueClient client = QueueClient.CreateFromConnectionString(serviceBusConnectionString, signInQueueName);

12. After the above code, create a new instance of the **BrokeredMessage** class by passing in the **QueueMessage** message into the constructor, as shown in the following code:

    BrokeredMessage queueMessage = new BrokeredMessage(message);

13. Invoke the **Send** method of the *QueueClient* variable by using the **BrokeredMessage** as the parameter:

    client.Send(queueMessage);

Task 3: Debug and verify the application

1. On the **Debug** menu, click **Start Debugging**.
2. On the home page for the **Contoso Events Administration** web application, click the **Events** 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 the sign-in sheet is being generated with the message:

Sign-In Document Generation in Progress.

1. Wait for one minute for the worker role to receive the queue message.
2. Verify that the application temporarily pauses execution at the breakpoint.
3. Press F5 to resume execution of the application
4. Wait for one minute, and then refresh the sign-in sheet page.
5. Click **Sign-In Sheet** to download the sign-in sheet from the server.
6. Close the **Internet Explorer** window.
7. Close the **Contoso.Events – Microsoft Visual Studio** window.

Results: After completing this exercise, you will have created and consumed messages from Service Bus Queues.