# Module 6: Storing Unstructured Data in Azure

# Lab: Storing Event Registration Data in Azure Cosmos DB

### Scenario

Even though event registrations could be stored in SQL, you have a unique need. Each event requires a different registration form that can be changed at any time. Essentially, registrations could be of any schema. A relational database such as SQL requires a well-defined schema. Because of your business requirement, you require a database that can store items with flexible structures (or schemas). To facilitate this you have elected to use Azure Cosmos DB for your event registrations.

### Objectives

After you complete this lab, you will be able to:

* Configure an Azure Cosmos DB Account
* Use Azure Cosmos DB in a .NET Application

### Lab Setup

* *Estimated Time*: 90 minutes

Before starting this lab, you must complete the lab in Module 2. For the lab in this module, you will use the available host machine. Also, you must complete the following steps:

1. On the host computer, click **Start**, type **Remote**, and then click **Remote Desktop Connection**.
2. In Remote Desktop Connection, provide the name of your virtual machine in the **Computer** box by using the following format:
   * **[Your VM IP Address]:[*Your VM RDP Port*]**

* **Note:** The name and port for your virtual machine might be saved in the Computer drop-down list. If this is the case, use this value instead of typing it in manually. If you are unsure about your virtual machine’s RDP port, use either of the Azure portals to find your virtual machine’s endpoints. The endpoint with the name **Remote Desktop** is the correct port for RDP. This port is randomized to protect your virtual machine from unauthorized access.

1. In Remote Desktop Connection, click **Connect**. Wait until the RDP client accesses the virtual machine.
2. If necessary, sign in by using the following credentials:
   * User name: **Student**
   * Password: **AzurePa$$w0rd**
3. Verify that you received the credentials to sign in to the Azure portal from your training provider. You will use these credentials and the Azure account throughout the labs in this course.

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

#### Task 1: Sign in to the Azure Portal

1. Sign in to the Azure Portal (https://portal.azure.com).
2. If this is your first time logging in to the Azure portal, you will see a dialog with a tour of the portal. Click Get Started.

#### Task 2: Create an Azure Storage Instance

1. Create a new **Storage Account** with the following details:
   * Name: star20532[your name in lowercase here]
   * Deployment model: Resource manager
   * Account kind: StorageV2 (general purpose v2)
   * Location: Closest region to your current location
   * Replication: Locally-redundant storage (LRS)
   * Performance: Standard
   * Access tier: Hot
   * Secure transfer required: Enabled
   * New resource group: MOD06NSQL
   * Virtual networks: Disabled

* **Note**: Wait for Azure to finish creating the storage account prior to moving forward with the lab. You will receive a notification when the *Storage Account* is created.

1. Access the **Access Keys** blade of your newly created **Storage account** instance.
2. Record any one of the **Keys** and record the value in the **Connection string** field. You will use this value later in this lab.

#### Task 3: Create an Azure SQL Database Instance

1. Create a new **SQL Database** with the following details:
   * Name: db20532
   * Resource Group: MOD06NSQL
   * Select Source: Blank Database
   * Server Name: sv20532[*Your Name Here*]
   * Admin login: testuser
   * Password/Confirm Password: TestPa$$w0rd
   * Location: Closest to your location
   * Pricing Tier: Basic

* **Note**: Wait for Azure to finish creating the SQL Database instance prior to moving forward with the lab. You will receive a notification when the *SQL Database* is created.

1. Access the **Connection strings** blade of your newly created **SQL database**.
2. Record the value of the **ADO.NET** connection string. Be sure to replace the placeholder values for {your\_username} and {your\_password} with the values **testuser** and **TestPa$$w0rd** respectively.

* **Note**: For example, if your copied connection string is Server=tcp:sv20532microsoft.database.windows.net,1433;Initial Catalog=db20532;Persist Security Info=False;User ID={your\_username};Password={your\_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;, your updated connection string would be Server=tcp:sv20532microsoft.database.windows.net,1433;Initial Catalog=db20532;Persist Security Info=False;User ID=testuser;Password=TestPa$$w0rd;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

1. Access the **Firewalls and virtual networks** blade of your newly created **SQL server**.
2. Add your virtual machine's IP Address to the list of allowed IP Address ranges.

* **Note**: It might take couple of minutes for the firewall changes to get updated on server.

#### Task 4: Create an Azure Cosmos DB Instance

1. Create a new **Azure Cosmos DB** instance with the following details:
   * ID: nosql20532[your name in lowercase here]
   * API: SQL
   * Resource Group: MOD06NSQL
   * Location: Closest to your location
   * Enable geo-redundancy: Unchecked
   * Configure virtual networks: Disabled

* **Note**: Wait for Azure to finish creating the Azure Cosmos DB account prior to moving forward with the lab. You will receive a notification when the *Azure Cosmos DB account* is created.

1. Access the **Keys** blade of your newly created **Azure Cosmos DB** instance.
2. Record the value of the **URI** and **PRIMARY KEY**. You will use these values later in this lab.

#### Task 5: Open the ASP.NET web application

1. Open the **Contoso.Events.sln** solution found in **Allfiles (F):\Mod06\Labfiles\Starter**.
2. Build the Solution.

#### Task 6: Update Various Application Settings

1. Open **appsettings.json** located in the **Contoso.Events.Web** project.
2. Locate and update the value of the following properties:
   * EndpointUrl: [Azure Cosmos DB URI]
   * AuthorizationKey: [Azure Cosmos DB Key]
   * EventsContextConnectionString: [SQL Database Connection String]
3. Save the **appsettings.json** file.
4. Open **local.settings.json** located in the **Contoso.Events.Worker** project.
5. Locate and update the value of the following properties:
   * AzureWebJobsStorage: [Storage Account Connection String]
   * AzureWebJobsDashboard: [Storage Account Connection String]
   * EventsContextConnectionString: [SQL Database Connection String]
   * CosmosEndpointUrl: [Azure Cosmos DB URI]
   * CosmosAuthorizationKey: [Azure Cosmos DB Key]
6. Save the **local.settings.json** file.

#### Task 7: Retrieve Registrant Names in the Azure Functions Project

1. Open the **ProcessDocuments.cs** file.
2. Locate and replace the **ProcessDocuments** class with the following class implementation:

* public static class ProcessDocuments  
  {  
   private static ConnectionManager \_connection = new ConnectionManager();  
   private static RegistrationContext \_registrationsContext = \_connection.GetCosmosContext();  
    
   [FunctionName("ProcessDocuments")]  
   public static async Task Run([HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = "")]HttpRequest request, TraceWriter log)  
   {  
   string message = request.Query["eventkey"];  
    
   log.Info($"Request received to generate sign-in sheet for event: {message}");  
    
   var registrants = await ProcessHttpRequestMessage(message);  
    
   log.Info($"Registrants: {String.Join(", ", registrants)}");  
    
   log.Info($"Request completed for event: {message}");  
   }  
    
   private static async Task<List<string>> ProcessHttpRequestMessage(string eventKey)  
   {  
   using (EventsContext eventsContext = \_connection.GetSqlContext())  
   {  
   await eventsContext.Database.EnsureCreatedAsync();  
   await \_registrationsContext.ConfigureConnectionAsync();  
    
   Event eventEntry = await eventsContext.Events.SingleOrDefaultAsync(e => e.EventKey == eventKey);  
    
   List<string> registrants = await \_registrationsContext.GetRegistrantsForEvent(eventKey);  
    
   return registrants;  
   }  
   }  
  }

1. Save the **ProcessDocuments.cs** file.

## Exercise 2: Updating the Events Website to use Azure Cosmos DB

#### Task 1: Implement RegistrationContext class

1. Open **RegistrationContext.cs** located in the **Contoso.Events.Data** project.
2. Locate and replace the **RegistrationContext** class with the following class implementation:
3. In the code editor tab for the **RegistrationContext.cs** file, locate the **RegistrationContext** class:

* public class RegistrationContext  
  {  
   protected Database Database { get; set; }  
    
   protected DocumentCollection Collection { get; set; }  
    
   protected DocumentClient Client { get; set; }  
    
   protected CosmosSettings CosmosSettings { get; set; }  
    
   public RegistrationContext(IOptions<CosmosSettings> cosmosSettings)  
   {  
   CosmosSettings = cosmosSettings.Value;  
   Client = new DocumentClient(new Uri(CosmosSettings.EndpointUrl), CosmosSettings.AuthorizationKey);  
   }  
    
   public async Task ConfigureConnectionAsync()  
   {  
   Database = await Client.CreateDatabaseIfNotExistsAsync(new Database { Id = CosmosSettings.DatabaseId });  
   Collection = await Client.CreateDocumentCollectionIfNotExistsAsync(Database.SelfLink, new DocumentCollection { Id = CosmosSettings.CollectionId });  
   }  
    
   public async Task<int> GetRegistrantCountAsync()  
   {  
   FeedOptions options = new FeedOptions { EnableCrossPartitionQuery = true };  
   IDocumentQuery<int> query = Client.CreateDocumentQuery<int>(Collection.SelfLink, "SELECT VALUE COUNT(1) FROM registrants", options).AsDocumentQuery();  
    
   int count = 0;  
   while (query.HasMoreResults)  
   {  
   FeedResponse<int> results = await query.ExecuteNextAsync<int>();  
   count += results.Sum();  
   }  
   return count;  
   }  
    
   public async Task<List<string>> GetRegistrantsForEvent(string eventKey)  
   {  
   IDocumentQuery<GeneralRegistration> query = Client.CreateDocumentQuery<GeneralRegistration>(Collection.SelfLink).Where(r => r.EventKey == eventKey).AsDocumentQuery();  
    
   List<string> registrants = new List<string>();  
   while (query.HasMoreResults)  
   {  
   FeedResponse<GeneralRegistration> results = await query.ExecuteNextAsync<GeneralRegistration>();  
   IEnumerable<string> resultNames = results.Select(r => $"{r.FirstName} {r.LastName}");  
   registrants.AddRange(resultNames);  
   }  
   return registrants;  
   }  
    
   public async Task<string> UploadEventRegistrationAsync(dynamic registration)  
   {  
   ResourceResponse<Document> response = await Client.CreateDocumentAsync(Collection.SelfLink, registration);  
   return response.Resource.Id;  
   }  
  }

1. Save the **RegistrationContext.cs** file.

## Exercise 3: Verify that the Events Web Site is using Azure Cosmos DB for Registrations

#### Task 1: Run the ASP.NET Web Application to View Events from the Local SQL Database

1. Set the **Contoso.Events.Web** project as the Startup Project and Debug the solution.
2. Debug the solution.
3. On the home page of the web application, verify that it displays a list of events.
4. Click any of the events in the list.
5. Record the value of the unique **Event Key** of the event you selected. You will use this value later in the lab.

* **Note**: You can see this unique event key by looking at the URL for the event. For example, if the URL is http://localhost:52127/events/detail/fy18septembergeneralconference09, the unique event key would be fy18septembergeneralconference09.

1. On the event web page, click the **Register Now** button.
2. Fill out all of the fields in the registration form and click Submit.
3. Record the value of the **Registration Id** displayed after submitting an event registration. You will use this value later in the lab.
4. Close the browser window that is displaying the website.
5. Set the **Contoso.Events.Worker** project as the Startup Project and Debug the solution.
6. Debug the solution.
7. In the command prompt, you will see a single **Function URL** displayed in *green text* for your **ProcessDocuments** Function endpoint. Record this value as you will use this value later in this lab.
8. Open **Internet Explorer** attempt to go to the **Function URL** you copied earlier in this lab. Use the **Enter** key to issue a **HTTP GET** request to the URL.
9. Return to the console application. You will observe various logging error messages. Most importantly, you will see see a message indicating that there were no registrants for the event key you specified (**null**):

* [00/00/0000 0:00:00 PM] Registrants:

1. Return to your open browser window and update your URL by adding a new *query string* parameter named **eventKey** and using the unique **Event Key** you copied earlier in this lab. Use the **Enter** key to issue a **HTTP GET** request to the URL.

* **Note**: For example, if your *Endpoint URL* is http://localhost:7071/api/ProcessDocuments and your **Event Key** is **fy18septembergeneralconference09**, your new URL would be http://localhost:7071/api/ProcessDocuments?eventKey=fy18septembergeneralconference09.

1. Return to the console application again. Observe that the registration you created earlier is included in the list of registrations. Below is an example list of event registrations:

* [00/00/0000 0:00:00 PM] Registrants: Example Person, Demo User, Unique Individual

1. Close the browser window that you are using to send requests.
2. Close the console window for the Azure Functions instance.

#### Task 2: Validate Azure Cosmos DB Data

1. Return to the Azure Portal.
2. Access the **Azure Cosmos DB** instance you created earlier in the lab.
3. Using the **Data Explorer**, start a new SQL query with the **RegistrationCollection** collection node.
4. In the Query tab, enter and execute the following SQL query:

* SELECT \* FROM registrations ORDER BY registrations.\_ts DESC

1. Observe the results of the Query.
2. In the query tab, replace the contents of the *query editor* with the following SQL query replacing the {registration\_id} placeholder with the value of the **Registration Id** you recorded earlier in this lab:

* SELECT VALUE { first: r.FirstName, last: r.LastName } FROM registrations r WHERE r.id = '{registration\_id}'

1. Execute and observe the results of the Query.
2. In the query tab, replace the contents of the *query editor* with the following SQL query replacing the {event\_key} placeholder with the value of the **Event Key** you recorded earlier in this lab:

* SELECT VALUE CONCAT(r.FirstName, ' ', r.LastName) FROM registrations r WHERE r.EventKey = '{event\_key}'

1. Execute and observe the results of the Query.

## Exercise 4: Cleanup Subscription

#### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. In the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to list all resource groups in the subscription:

* az group list

1. Type in the following command and press **Enter** to view a list of possible CLI commands to *delete a Resource Group*:

* az group delete --help

#### Task 2: Delete Resource Group

1. Type in the following command and press **Enter** to delete the **MOD06NSQL** *Resource Group*:

* az group delete --name MOD06NSQL --no-wait --yes

1. Close the **Cloud Shell** prompt at the bottom of the portal.

#### Task 3: Close Active Applications

1. Close the currently running web browser application.
2. Close the currently running **Visual Studio** application.

**Review**: In this exercise, you "cleaned up your subscription" by removing the **Resource Groups** used in this lab.

©2016 Microsoft Corporation. All rights reserved. The text in this document is available under the [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/legalcode), additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are ***not*** included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.