# Module 4: Storing SQL Data in Azure

# Lab: Storing Event Data in Azure SQL Databases

### Scenario

Now that the web application is ready for publishing, you can begin migrating the web application to Azure by creating a database in Azure. You decided to use a database initializer and Entity Framework Code First to automate the creation of your database in SQL Database.

### Objectives

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

* Create an Azure SQL Database Instance
* Use Entity Framework with Local SQL Server
* Use Entity Framework with Azure SQL Database

### 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: Creating an Azure SQL Database Instance

#### 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 SQL database by using the Azure Portal

1. Create a **SQL Database** with the following details.
   * Name: db20532
   * New Resource Group: MOD04SQLD
   * Select source: Blank database
   * Server Name: sv20532[Your Name Here]
   * Server admin login: testuser
   * Server admin password: TestPa$$w0rd
   * Server Location: Region that is closes to your location.
   * Pricing Tier: Basic
2. Record the server and database names for your new SQL Database instance.

### Task 3: Configure SQL Server Firewall

1. In the navigation pane on the left side, scroll down, and then click **All services**.
2. Scroll-down, locate and select the **SQL databases** option.
3. In the list of **SQL Databases**, select the SQL database named **db20532**.
4. In the **db20532 - SQL database** blade, locate the **Essentials** panel.
5. Locate the **Server name** section and click on the associated hyperlink to navigate to the server blade.
6. In the **SQL server** blade, locate the **Essentials** panel.
7. Locate the **Firewalls and virtual networks** section and click on the associated hyperlink to navigate to manage firewall settings.
8. In the **Firewalls and virtual networks** blade, click the **Add client IP** button to add your virtual machine's IP Address to the list of allowed IP Address ranges.
9. Click on **Save** at the top of the blade. Once saved, close the confirmation dialog by clicking the **Ok** button.

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

**Results**: After completing this exercise, you will have created both servers and databases in the SQL Database service.

## Exercise 2: Using Entity Framework with Local SQL Server

#### Task 1: Run the ASP.NET web application

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):\Mod04\Labfiles\Starter**, and then double-click **Contoso.Events.sln**.
4. In the **Solution Explorer** pane, right-click the **Contoso.Events.Web** project, and then click **Set as Startup Project**.
5. On the **Debug** menu, click **Start Debugging**.
6. On the home page of the web application, verify that only 'Upcoming Events' show on the home page.
7. Close the browser window that is displaying the website.

#### Task 2: Configure Database Initialization Logic

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, expand the **Contoso.Events.Data** project.
2. In the **Solution Explorer** pane, double-click **ContextInitializer.cs**.
3. Locate the **InitializeAsync** method:

* public async Task InitializeAsync(EventsContext eventsContext)

1. Within the **InitializeAsync** method, add the following line of code to ensure the database is created:

* await eventsContext.Database.EnsureCreatedAsync();

1. Add the following block of code to create a conditional *if* block that only executes the code within the block if there are no events in the database:

* if (!await eventsContext.Events.AnyAsync())  
  {  
  }

1. Within the newly created *if* block, add the following line of code to create a new instance of the **Event** class:

* Event eventItem = new Event();

1. Still within the *if* block, add the following block of code to set various properties of the new **Event** class instance:

* eventItem.EventKey = "FY17SepGeneralConference";  
  eventItem.StartTime = DateTime.Today;  
  eventItem.EndTime = DateTime.Today.AddDays(3d);  
  eventItem.Title = "FY17 September Technical Conference";  
  eventItem.Description = "Sed in euismod mi.";  
  eventItem.RegistrationCount = 1;

1. Still within the *if* block, add the following line of code to add the new **Event** class instance to the **Events** property of type **DbSet<Event>**:

* eventsContext.Events.Add(eventItem);

1. Outside of and after the *if* block, add the following line of code to save the changes to the database context:

* await eventsContext.SaveChangesAsync();

1. Your **InitializeAsync** method should now look like this:

* public async Task InitializeAsync(EventsContext eventsContext)  
  {  
   await eventsContext.Database.EnsureCreatedAsync();  
    
   a. (!await eventsContext.Events.AnyAsync())  
   {  
   Event eventItem = new Event();  
   eventItem.EventKey = "FY17SepGeneralConference";  
   eventItem.StartTime = DateTime.Today;  
   eventItem.EndTime = DateTime.Today.AddDays(3d);  
   eventItem.Title = "FY17 September Technical Conference";  
   eventItem.Description = "Sed in euismod mi.";  
   eventItem.RegistrationCount = 1;  
   eventsContext.Events.Add(eventItem);  
   }  
    
   await eventsContext.SaveChangesAsync();  
  }

#### Task 3: Validate Database Initialization

1. In the **Solution Explorer** pane, right-click the **Contoso.Events.Web** project, and then click **Set as Startup Project**.
2. On the **Debug** menu, click **Start Debugging**.
3. On the home page of the web application, you will see that no events are shown.

* **Note**: We have not written the code to display events yet. We simply are running the web application to initialize and populate the database.

1. Close the browser window that is displaying the website.
2. Return to the open instance of **Visual Studio**.
3. In Visual Studio, open the **View** menu and then select the **Server Explorer** option.
4. Right click on **Data Connections** and click on **Add Connection**.
5. Choose **Microsoft SQL Server** for data source and click **Continue**.
6. In the **Add Connection** wizard, provide following values and click **OK**.
   1. In the **Server name** field, type **(localdb)\mssqllocaldb**.
   2. In the **Authentication** list, select **Use Windows Authentication**.
   3. In the **Select or enter a database name** dropdown, select **Contoso.Events.Mod04**.
   4. Click the **OK** button.

* **Note**: If firewall rules are not updated on the server, you may have to wait a few more minutes before proceeding.

1. On Visual Studio **Server Explorer**, expand **Data Connections** then the **localdb** node.
2. Right click the **localdb** node and select the **New Query** option.
3. In the *Query Editor* tab, enter the following SQL query:

* SELECT \* FROM Events

1. In Visual Studio, open the **SQL** menu and then select the **Execute** option.
2. Observe the results of the query.
3. In the *Query Editor* tab, replace the existing SQL query with the following SQL query:

* DROP TABLE Events

1. In Visual Studio, open the **SQL** menu and then select the **Execute** option.
2. Wait for this query to finish executing.
3. Close the *Query Editor* tab.

#### Task 4: Update Database Initialization Logic

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, expand the **Contoso.Events.Data** project.
2. In the **Solution Explorer** pane, double-click **ContextInitializer.cs**.
3. Locate the **InitializeAsync** method:

* public async Task InitializeAsync(EventsContext eventsContext)

1. Replace the method with the following method implementation:

* public async Task InitializeAsync(EventsContext eventsContext)  
  {  
   await eventsContext.Database.EnsureCreatedAsync();  
    
   if (!await eventsContext.Events.AnyAsync())  
   {  
   await eventsContext.Events.AddRangeAsync(  
   new List<Event>()   
   {  
   new Event { EventKey = "GeneralConferenceAlpha", StartTime = DateTime.Today, EndTime = DateTime.Today.AddDays(5d), Title = "First General Conference", Description = "Sed in euismod mi.", RegistrationCount = 15 },  
   new Event { EventKey = "GeneralConferenceBravo", StartTime = DateTime.Today.AddDays(10d), EndTime = DateTime.Today.AddDays(15d), Title = "Second General Conference", Description = "Sed in euismod mi.", RegistrationCount = 20 },  
   new Event { EventKey = "GeneralConferenceCharlie", StartTime = DateTime.Today.AddDays(20d), EndTime = DateTime.Today.AddDays(25d), Title = "Third General Conference", Description = "Sed in euismod mi.", RegistrationCount = 5 },  
   new Event { EventKey = "GeneralConferenceDelta", StartTime = DateTime.Today.AddDays(30d), EndTime = DateTime.Today.AddDays(35d), Title = "Fourth General Conference", Description = "Sed in euismod mi.", RegistrationCount = 25 },  
   new Event { EventKey = "GeneralConferenceEcho", StartTime = DateTime.Today.AddDays(40d), EndTime = DateTime.Today.AddDays(45d), Title = "Fifth General Conference", Description = "Sed in euismod mi.", RegistrationCount = 10 },  
   new Event { EventKey = "GeneralConferenceFoxtrot", StartTime = DateTime.Today.AddDays(50d), EndTime = DateTime.Today.AddDays(55d), Title = "Sixth General Conference", Description = "Sed in euismod mi.", RegistrationCount = 0 }  
   }  
   );  
    
   await eventsContext.SaveChangesAsync();  
   }  
  }

#### Task 5: Write Entity Framework Queries in the ASP.NET MVC Controllers

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, expand the **Contoso.Events.Web** project.
2. In the **Solution Explorer** pane, expand the **Controllers** folder.
3. In the **Solution Explorer** pane, double-click **HomeController.cs**.
4. Locate the **Index** method:

* public IActionResult Index([FromServices] EventsContext eventsContext, [FromServices] IOptions<ApplicationSettings> appSettings)

1. Within the **Index** method, locate the following line of code:

* var upcomingEvents = Enumerable.Empty<Event>();

1. Replace that line of code with the following block of code to query the **Events** table, order the results by the **StartTime** property and then take a subset of results based on an application setting:

* var upcomingEvents = eventsContext.Events  
   .Where(e => e.StartTime >= DateTime.Today)  
   .OrderBy(e => e.StartTime)  
   .Take(appSettings.Value.LatestEventCount);

1. In the **Solution Explorer** pane, double-click **EventsController.cs**.
2. Locate the **Index** method:

* public IActionResult Index([FromServices] EventsContext eventsContext, [FromServices] IOptions<ApplicationSettings> appSettings, int? page)

1. Within the **Index** method, locate the following line of code:

* var pagedEvents = Enumerable.Empty<Event>();

1. Replace that line of code with the following block of code to query the **Events** table, and use the **Skip** and **Take** methods to create a page of results based on the current page number:

* int currentPage = page ?? 1;  
  int totalRows = eventsContext.Events.Count();  
  int pageSize = appSettings.Value.GridPageSize;  
  var pagedEvents = eventsContext.Events  
   .OrderByDescending(e => e.StartTime)  
   .Skip(pageSize \* (currentPage - 1))  
   .Take(pageSize);

1. Within the **Index** method, locate the following block of code:

* EventsGridViewModel viewModel = new EventsGridViewModel  
  {  
   CurrentPage = 0,  
   PageSize = 0,  
   TotalRows = 0,  
   Events = pagedEvents  
  };

1. Replace that block of code with the following block of code to set various properties of the **EventsGridViewModel** class instance:

* EventsGridViewModel viewModel = new EventsGridViewModel  
  {  
   CurrentPage = currentPage,  
   PageSize = pageSize,  
   TotalRows = totalRows,  
   Events = pagedEvents  
  };

1. Locate the **Detail** method:

* public IActionResult Detail([FromServices] EventsContext eventsContext, string key)

1. Within the **Detail** method, locate the following line of code:

* var matchedEvent = default(Event);

1. Replace that line of code with the following block of code to query the **Events** table for a single record that matches the **EventKey** property:

* var matchedEvent = eventsContext.Events  
   .SingleOrDefault(e => e.EventKey == key);

#### Task 6: Run the ASP.NET Web Application to View Events from the Local SQL Database

1. In the **Solution Explorer** pane, right-click the **Contoso.Events.Web** project, and then click **Set as Startup Project**.
2. On the **Debug** menu, click **Start Debugging**.
3. On the home page of the web application, verify that it shows a list of multiple events.
4. Click the **Events** hyperlink at the top of the page.
5. Observe the grid of events.
6. Click the *"arrows pointing right"* at the bottom of the page to go to the next page of results.
7. Observe the results in the grid on the next page of results.
8. Click the **Go to Event** button for one of the event results.
9. Observe the event details displayed on the current page.
10. Close the browser window that is displaying the website.

## Exercise 3: Using Entity Framework with Azure SQL Database

#### Task 1: Create a Production Settings File

1. In the **Solution Explorer** pane, expand the **Contoso.Events.Web** project.
2. Right-click the **Contoso.Events.Web** project, hover over the **Add** option and then select the **New Item...** option.
3. In the **Add New Item** dialog, perform the following actions:
   1. Expand the **ASP.NET Core** category.
   2. Select the **Data** option from the list of template categories.
   3. Select the **JSON File** option from the list of templates.
   4. In the **Name** dialog box, enter the value **appsettings.Production.json**.
   5. Click the **Add** button.
4. In the code editor tab for the **appsettings.Production.json** file, replace the contents of the file with the following JSON object:

* {  
   "ConnectionStrings": {  
   "EventsContextConnectionString": ""  
   }  
  }

1. Update the value of the **EventsContextConnectionString** property by adding a connection string to your database in Azure. Remove the placeholder string [SQL Database Instance Name] and replace it with the name of your **SQL Server** instance in Azure.

* Server=tcp:[SQL Database Instance Name].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;
* **Note**: The name of your server instance is typically in the following format: sv20532[Your Name Here]. Ensure that you remove the square brackets as you replace the placeholder.

1. Save the **appsettings.Production.json** file.

#### Task 2: Publish the Web Application with Updated DbContext to Azure

1. In the **Solution Explorer** pane, right-click the **Contoso.Events.Web** project, and then click **Publish**.
2. In the **Pick a publish target** window, perform the following actions:
   1. Click the **App Service** option.
   2. Select the **Create New** radio button.
   3. Click the **Publish** button.
3. In the **Create App Service** dialog, perform the following actions:
   1. In the **App Name** dialog box, keep the autogenerated value created by the tool.
   2. In the **Resource Group** list, select the **MOD04SQLD** option.
   3. In the **Hosting Plan** list, keep the autogenerated value created by the tool.
   4. Click the **Create** button.
4. The publish process will start automatically. Wait for the publish process to complete.

* **Note**: It typically takes five to ten minutes for the publish process to complete. You can track the progress of your publish in the Microsoft Azure Activity Log (**View** > **Other Windows** > **Web Publish Activity**) pane that displays when you publish your Web App project.

#### Task 3: Verify that the Azure Web App website is using the new data

1. Wait for the publish process to complete and the console window to display the message **Publish Succeeded**.

* **Note**: The publish process is complete when the message “**Publish Succeeded”** displays in the **Web Publish Activity** history console. The green circular indicator in the Activity Log does not indicate that the publish process is complete, but it indicates that the package is uploaded successfully.

1. A browser tab will open automatically displaying the deployed web application.
2. Verify that the website displays the events that you created in your Entity Framework context initializer.

#### Task 4: View the data in the Azure SQL Database

1. Return to the open instance of **Visual Studio**.
2. In Visual Studio, open the **View** menu and then select the **Server Explorer** option.
3. **Expand** the **Data Connections** node.
4. Right click on **Data Connections** and click on **Add Connection**.
5. In the **Add Connection** wizard, provide following values and click **OK**.
   1. In the **server name** field, type **sv20532[your name].database.windows.net**.
   2. Select **Use SQL Server Authentication**.
   3. In the **Username** field, type **testuser**.
   4. In the **Password** field, type **TestPa$$w0rd**.
   5. In the **Select or enter a database name** dropdown, select **db20532**.
   6. Click the **OK** button.

* **Note**: If firewall rules are not updated on the server, you may have to wait a few more minutes before proceeding.

1. On Visual Studio **Server Explorer**, expand **Data Connections** then the **sv20532[your name].db20532.dbo** node and finally the **Tables** node.
2. Right click **Tables** and click **Refresh**.
3. Right click **Events** table, and then select **Show Table Data**.
4. In the **dbo.Events** table, view the automatically generated records.
5. Close the **Events** table window.
6. Close the **Internet Explorer** application.
7. Close the **Contoso.Events – Microsoft Visual Studio** window.

**Results**: After completing this exercise, you will have configured Entity Framework to initialize a new database with seed data.

## 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 **MOD04SQLD** *Resource Group*:

* az group delete --name MOD04SQLD --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.