# Module 3: Hosting Web Applications on the Azure Platform

# Lab: Creating an ASP.NET Web App by Using Azure Web Apps

## Exercise 1: Creating an Azure Web App and Function App

#### 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. In the email address box, type the email address of your Microsoft account.
4. Click **Next**.
5. In the password box, type the password for your Microsoft account.
6. Click **Sign In**.

* **Note**: If this is your first time logging in to the Portal, you may be prompted with a “Welcome” dialog. You can safely close this dialog and continue.

#### Task 2: Create a Web App instance by using the Portal

1. In the navigation pane on the left side of the screen, click **Create a resource**.
2. In the **New** blade that appears, enter the following text into the search box and then press the **Enter** key:

* web app sql

1. In the search results, select the **Web App + SQL** template.
2. In the **Web App + SQL** blade, click the **Create** button.
3. In the form that displays, perform the following steps:
   1. In the **App name** dialog box, create a unique name for your website.
   2. In the **Resource Group** section, select the **Create new** option and then enter the value **MOD03WEBA** into the dialog box.
   3. Select the **App Service plan** option.
   4. In the **App Service plan** blade, click the **Create New** button.
   5. In the next **App Service plan** blade, locate the **App Service plan** dialog box and provide the value **FreePlan**.
   6. In the **Location** list, select the region that is closest to your location.
   7. Select the **Pricing tier** option.
   8. In the new blade that appears, locate and select the **Dev / Test** tab.
   9. Click the **F1** pricing tier.
   10. Click the **Apply** button.
   11. Back in the **App Service plan** blade, click the **OK** button.
   12. Back in the **Web App + SQL** blade, click the **SQL Database** option.
   13. In the **Database** blade, click the **Create a new database** option.
   14. In the **SQL Database** blade, locate the **Name** field and provide the value **EventsContextDB**.
   15. In the **SQL Database** blade, click the **Target server** option.
   16. In the **Server** blade, click the **Create a new server** option.
   17. In the **Server name** dialog box, create a unique name for your server instance.
   18. In the **Server admin login** dialog box, create a user name (**testuser**) for your administrative SQL user.
   19. In the **Password** and **Confirm Password** dialog boxes, create a password (**TestPa$$w0rd**) for your administrative SQL user.
   20. In the **Location** list, select the region that is closest to your location.
   21. Click the **Select** button.
   22. Back in the **SQL Database** blade, click the **Pricing tier** option.
   23. In the **Resource Configuration & Pricing** blade, select the **Free** option and click the **Apply** button.
   24. Back in the **SQL Database** blade, click the **Select** button.
   25. Back in the **Web App + SQL** blade, click the **Create** button to create your Web App and SQL database.
4. In the navigation pane on the left side of the screen, click **Create a resource**.
5. In the **New** blade that appears, enter the following text into the search box and then press the **Enter** key:

* function app

1. In the search results, select the **Function App** template.
2. In the **Function App** blade, click the **Create** button.
3. In the form that displays, perform the following steps:
   1. In the **App name** dialog box, create a unique name for your website.
   2. In the **Resource Group** section, select the **Use existing** option and then select the **MOD03WEBA** Resource Group from the list.
   3. In the **OS** group, select the **Windows** option.
   4. In the **Hosting Plan** list, select the **Consumption Plan** option.
   5. In the **Location** list, select the region that is closest to your location.
   6. In the **Storage** section, select the **Create new** radio button and leave the name set to the recommended value that is displayed.
   7. Click the **Create** button to create your Function App.

#### Task 3: Go to the newly created Web App's placeholder page

**Note**: Creating Apps within App Service is typically very fast, but you have to wait for the App to be created and running before you can go to the App's blade. When the App is running, its status displays as Running in the **App Service** list.

1. In the navigation pane on the left side, click **All services**.
2. Scroll-down, locate and select the **App Services** option.
3. In the **App Services** list, click the **Web App** that you just created.
4. Observe the **Essentials** panel at the top of the blade.
5. Under the **Url** header, click the hyperlink to the Web App.
6. Verify that the Web App is running at the specified address.
7. Close the tab displaying the website.

**Results**: After completing this exercise, you will have used the Azure Portal to create a Web App instance.

## Exercise 2: Deploying an ASP.NET Web Application to an Azure Web App

#### Task 1: Open an existing ASP.NET web application project with Visual Studio 2017

1. On the Start screen, click the **Desktop** tile.
2. On the taskbar, click the **File Explorer** icon.
3. In the Libraries window, go to **Allfiles (F):\Mod03\Labfiles\Starter\**, and then double-click **Contoso.Events.sln**.
4. In the **Solution Explorer** pane, right-click the **Contoso.Events.Management** project, and then click **Set as Startup Project**.
5. On the **Debug** menu, click **Start Debugging**.

* **Note**: If this is the first time you are creating a build for this solution, NuGet will implicitly restore any missing packages. You do not have to manually restore the missing packages.

1. On the home page of the web application, verify that a list of two events displays under the **Events Administration** header.
2. Close the tab displaying the Web App.

#### Task 2: Download the publish profile for the Azure Web App

1. Return to the **Azure Portal**.
2. In the navigation pane on the left side, click **All services**.
3. Scroll-down, locate and select the **App Services** option.
4. In the **App Services** list, click the Web App that you created earlier in this lab.
5. Click the **Get publish profile** button at the top of the blade.
6. In the download dialog box, click the arrow to the right of the **Save** button, and then click **Save As**.
7. In the **Save As** dialog box, go to **Allfiles (F):\Mod03\Labfiles\Starter**, and then click **Save**.

#### Task 3: Publish the ASP.NET web application to the Azure Web App

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, right-click **Contoso.Events.Management**, and then click **Publish**.
2. In the **Pick a publish target** window, click the **Import Profile...** button.
3. In the **Import Publish Settings File** window, go to **Allfiles (F):\Mod03\Labfiles\Starter**, and then double-click your previously saved publish profile.
4. The publish process will start automatically. Wait for the publish process to complete.
5. A browser tab will open automatically displaying the deployed web application.

#### Task 4: Verify that the web application is successfully published

**Note**: If you see the placeholder page that displayed in Exercise 1 - Task 2, it might mean that client-side caching is occurring. You can always force refresh your browser and cache by pressing Ctrl+F5.

1. On the home page of the web application, verify that a list of two events displays under the **Events Administration** header.
2. Close the tab displaying the website.

**Results**: After completing this exercise, you will have used a publish profile to publish web applications directly to a Web App.

## Exercise 3: Configuring an Azure Web App

#### Task 1: Implement logic to read a configuration setting from app settings

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, right-click the **Contoso.Events.Models** project, hover over the **Add** option and then select the **Class...** option.
2. In the **Add New Item** dialog, perform the following actions:
   1. Select the **Visual C# Items** option from the list of template categories.
   2. Select the **Class** option from the list of templates.
   3. In the **Name** dialog box, enter the value **ApplicationSettings.cs**.
   4. Click the **Add** button.
3. In the code editor tab for the **ApplicationSettings.cs** file, replace the entire content of the file with the following code:

* namespace Contoso.Events.Models  
  {  
   public class ApplicationSettings  
   {  
   public int LatestEventCount { get; set; }  
   }  
  }

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, expand the **Contoso.Events.Management** project.
2. In the **Solution Explorer** pane, expand the **Controllers** folder and then double-click **HomeController.cs**.
3. **Locate** the following code in line 16 of HomeController.cs:

* public async Task<IActionResult> Index([FromServices] EventsContext eventsContext)

1. **Replace** the line of code with the following code:

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

1. **Locate** the following code in line 18 of HomeController.cs:

* var events = await eventsContext.Events.OrderBy(e => e.StartTime).Take(2).ToListAsync<Event>();

1. **Replace** the line of code with the following line of code:

* var events = await eventsContext.Events.OrderBy(e => e.StartTime).Take(settings.Value.LatestEventCount).ToListAsync<Event>();

1. **Locate** the following code in line 23 of HomeController.cs:

* LatestEventCount = 2

1. **Replace** the line of code with the following line of code:

* LatestEventCount = settings.Value.LatestEventCount

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, expand the **Contoso.Events.Management** project.
2. In the **Solution Explorer** pane, double-click **appsettings.json**.
3. In the JSON object, in line 1, locate the following block of code:

* {  
   "ConnectionStrings": {  
   "defaultConnection": "Server=(localdb)\\mssqllocaldb;Database=Contoso.Events.Mod03;Trusted\_Connection=True;MultipleActiveResultSets=true"  
   },

1. Replace that line of code with the following block of code:

* {  
   "ApplicationSettings": {  
   "LatestEventCount": 5  
   },  
   "ConnectionStrings": {  
   "defaultConnection": "Server=(localdb)\\mssqllocaldb;Database=Contoso.Events.Mod03;Trusted\_Connection=True;MultipleActiveResultSets=true"  
   },

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, expand the **Contoso.Events.Management** project.
2. In the **Solution Explorer** pane, double-click **Startup.cs**.
3. In the **Startup** class at line 26, enter the following code:

* services.Configure<ApplicationSettings>(Configuration.GetSection(nameof(ApplicationSettings)));

1. **Save** all open editor files.
2. In the **Solution Explorer** pane, right-click the **Contoso.Events.Management** project, and then click **Set as Startup Project**.
3. On the **Debug** menu, click **Start Debugging**.
4. On the home page of the web application, verify that a list of five events displays under the **Events Administration** header.
5. Close the tab displaying the Web App.

#### Task 2: Publish web application to the Azure Web App

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, right-click **Contoso.Events.Management**, and then click **Publish**.
2. Click the **Publish** button.
3. On the home page of the web application, verify that a list of five events displays under the **Events Administration** header.
4. Close the tab displaying the Web App.

#### Task 3: Modify the app settings in the Portal

1. Return to the **Azure Portal**.
2. In the navigation pane on the left side, click **All services**.
3. Scroll-down, locate and select the **App Services** option.
4. In the **App Services** list, click the Web App that you created earlier in this lab.
5. In your *Web App*'s blade, locate the **Settings** section.
6. In the **Settings** section, select the **Application settings** option.
7. In the **Application settings** blade, scroll-down to the **Application settings** section.
8. Click the **+ Add new setting** button.
9. In the *Enter a name* dialog box, type **ApplicationSettings:LatestEventCount**.
10. In the *Enter a value* dialog box, type **3**.
11. Click the **Save** button at the top of the blade.

* **Note**: After you update the configuration settings, you must wait for a few seconds to see the configuration changes. You will know that the configuration changes are applied when the **Configure** tab is editable again.

#### Task 4: Verify that the app settings are successfully updated

1. On the left side of the *Web App*'s blade, click the **Overview** option.
2. Click the **Browse** button at the top of the blade.
3. On the home page of the web application, verify that a list of three events displays under the **Events Administration** header.
4. Close the tab displaying the Web App.

**Results**: After completing this exercise, you will have configured custom app settings by using a Web.config file and the Azure Portal.

## Exercise 4: Deploy a Console Application to an Azure Function App

#### Task 1: Create Azure Function Project in Visual Studio

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window; locate the **Contoso.Events** solution, right-click the solution, hover over the **Add** option and then select the **New Project...** option.
2. In the **Add New Project** dialog, perform the following actions:
   1. Expand the **Visual C#** option from the list of template categories.
   2. Select the **Cloud** option from the list of template categories.
   3. Select the **Azure Functions** option from the list of templates.
   4. In the **Name** dialog box, enter the value **Contoso.Events.API**.
   5. In the **Location** dialog box, enter the value **F:\Mod03\Labfiles\Starter**.
   6. Click the **OK** button.
3. In the **New Project - Contoso.Events.API** dialog, perform the following actions:
   1. In the runtime list at the top of the dialog, select the **Azure Functions v2 (.NET Core)** runtime.
   2. Select the **Http trigger** template.
   3. In the **Storage Account (AzureWebJobsStorage)** list, select the **Storage Emulator** option.
   4. In the **Access rights** list, select the **Anonymous** option.
   5. Click the **OK** button.

#### Task 2: Deploy Azure Function Project

1. In the **Solution Explorer** pane of the Contoso.Events - Microsoft Visual Studio window, right-click **Contoso.Events.API**, and then click **Publish**.
2. In the **Pick a publish target** window, perform the following actions:
   1. Click the **Azure Function App** option.
   2. Select the **Select Existing** radio button.
   3. Click the **Publish** button.
3. In the **App Service** dialog, expand the **MOD03WEBA** resource group and select the **Function App** you created earlier in this lab.
4. Click the **OK** button.
5. The publish process will start automatically. Wait for the publish process to complete.

* **Note**: You may see a warning if the version of Azure Functions running in Azure doesn't match the version specified in Visual Studio. If prompted to *"Update Azure application settings to match local Functions version?"*, select the **Yes** button.

#### Task 3: Test Azure Function

1. Return to the **Azure Portal**.
2. In the navigation pane on the left side, click **All services**.
3. Scroll-down, locate and select the **App Services** option.
4. In the **App Services** list, click the **Function App** that you created earlier in this lab.
5. In the **Function Apps** blade, observe the name of your newly created function:

* **Note**: If you did not modify your Visual Studio project, your default Function should be named **Function1**.

1. Observe the **Essentials** panel at the top of the blade.
2. Under the **Url** header, click the hyperlink to the Function App.
3. Verify that the Function App is running at the specified address.
4. Add the following text to the end of your URL to invoke the **Function1** api using the text 20532 Student\*\* as a query string parameter:api/Function1?name=20532+Student``

* **Note**: For example, if your function URL is https://20532dmod03func.azurewebsites.net/, the modifed URL would be https://20532dmod03func.azurewebsites.net/api/Function1?name=20532+Student.

1. Observe that the Function App creates an echo response with the text Hello, 20532 Student.
2. Close the **Internet Explorer** application.
3. Close the **Contoso.Events - Microsoft Visual Studio** application.

**Results**: After completing this exercise, you will have created and tested a custom Azure Function.

## Exercise 5: 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 **MOD03WEBA** *Resource Group*:

* az group delete --name MOD03WEBA --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 **Microsoft Edge** 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.