# Module 3: Hosting Web Applications on the Azure Platform

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

### Scenario

You have an events administration application that is currently used by a static set of users. The application must be upgraded to handle all the users in your organization in the future. You need a hosting option that provides the least amount of friction so that you can immediately deploy the web application for immediate use. You also need the hosting option to be flexible enough so that it allows you to configure and scale the web application, thereby ensuring that it can handle an increase in the number of administrative users. For these reasons, you have chosen to deploy the application to Web Apps. Web Apps will also give you the flexibility to integrate your application with Azure Active Directory in the future so that all of your organization’s users can access the application.

In this lab, you will create a Web App, deploy your existing application, and then configure the Web App after deployment.

### Objectives

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

* Deploy an ASP.NET Web Application to an Azure Web App
* Configure an Azure Web App
* Deploy a Console Application to an Azure Function App

### 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 Web App and Function App

#### 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.

**Results**: After completing this exercise, you will have signed in to the Azure Portal.

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

1. Open the **new** panel, and search for **web app sql**.
2. In the **Marketplace** blade, create a new instance using **Web App + SQL** with the following details:
   * Name: Enter a globally unique name
   * New Resource Group: MOD03WEBA
   * New App Service Plan Name: FreePlan
   * New App Service Plan Location: Select the region that is closest to your location
   * New App Service Plan Pricing Tier: F1
   * SQL Database Name: EventsContextDB
   * SQL Database Target Server Name: Give it a globally unique name
   * SQL Database Target Server admin login: testuser
   * SQL Database Target Server admin Password: TestPa$$w0rd
   * SQL Database Target Server Location: Select the region that is closest to your location
   * SQL Database Target Server Pricing Tier: Free
3. Open the **new** panel, and search for **Function App**.
4. In the **Marketplace** blade, create a new instance using **Function App** with the following details:
   * App Name: Enter a globally unique name
   * Existing Resource Group: MOD03WEBA
   * OS: Windows
   * Hosting Plan: Consumption Plan
   * Location: Select the region that is closest to your location
   * Storage: Select Create new and name at its default value

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

1. View the web app you just created on your subscription.
2. Verify the Web App is running at the **Url** listed.

## 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. Go to **Allfiles (F):03\* and run** Contoso.Events.sln\*\*.
2. Set **Contoso.Events.Management** project as the Startup Project.
3. Debug the solution.

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

1. Return to the **Azure Portal**
2. Access the Web App you created earlier in the lab.
3. Download and save the **publish profile** to \*\*Allfiles (F):03\*

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

1. Back in Studio Visual 2017, publish **Contoso.Events.Management** by importing the publish profile you previously downloaded.

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

1. On the home page of the web application, verify that a list of two events displays under the Events Administration header.

## Exercise 3: Configuring an Azure Web App

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

1. Create a new class in the **Contoso.Event.Models** project named **ApplicationSettings.cs**.
2. Replace the entire code in **ApplicationSettings.cs** with the following code:

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

1. Access **HomeController.cs** located in the **Controllers** folder of the **Contoso.Events.Management** project.
2. **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. Access **appsettings.json** within the same project.
2. 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. Access **Startup.cs** within the same project.
2. 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. Debug the solution.
3. On the home page of the web application, verify that a list of five events displays under the Events Administration header.

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

1. Publish **Contoso.Events.Management**.
2. On the home page of the web application, verify that a list of five events displays under the Events Administration header.

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

1. Return to the **Azure Portal**.
2. Access the **Web App** you created earlier in the lab.
3. Locate **Application Settings** and create one with the following details:
   * Name: ApplicationSettings:LatestEventCount
   * Value: 3
4. Save your settings

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

1. Access the home page of your Web App.
2. On the home page of the web application, verify that a list of three events displays under the Events Administration header.

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

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

1. Create a new **project** for the **Contoso.Events** solution with the following details:
   * Project Template: Azure Functions
   * Name: Contoso.Events.API
   * Location: F:03
   * Runtime: Azure Functions v2 (.NET Core)
   * Template: Http trigger
   * Storage Account (AzureWebJobsStorage): Storage Emulator
   * Access rights: Anonymous

#### Task 2: Deploy Azure Function Project

1. Publish **Contoso.Events.API**.
2. Select the **Function App** you created earlier in this lab as a publish target.

#### Task 3: Test Azure Function

1. Return to the Azure Portal.
2. Access the **Function App** you created earlier in this lab.
3. View the **Url** listed to verify the Function App is running.
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.

## 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.