Module 15: **Deploying ASP.NET MVC 5 Web Applications**

# ****Contents:****

|  |  |
| --- | --- |
|  | [Module Overview](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C15.html#P1) |
| **Lesson 1:** | [Deploying a Web Application](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C15.html#P2) |
| **Lesson 2:** | [Deploying an ASP.NET MVC 5 Web Application](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C15.html#P3) |
| **Lab:** | [Deploying ASP.NET MVC 5 Web Applications](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C15.html#P4) |
|  | [Module Review and Takeaways](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C15.html#P5) |

# ****Module Overview****

After developing an application, you need to deploy the application to the production environment to make the application available to users. The configuration or deployment method of each application varies based on the setup of the production environment. ASP.NET MVC applications have some special considerations such as the configuration needed for IIS.

### ****Objectives****

After completing this module, you will be able to:

|  |  |
| --- | --- |
| • | Configure and deploy an ASP.NET web application. |
| • | Deploy an ASP.NET MVC 5 web application. |

# Lesson 1: ****Deploying a Web Application****

The considerations for deploying a web application to a web farm are different for single and multi-server configurations. For example, in multi-server web farms, you need to ensure that the state information is available on all web servers. You also need to ensure that the dependencies needed for hosting ASP.NET MVC 5 web applications are present in the web farm. If any of the application dependencies are missing on the production web servers, some or all of your applications may not function and users may encounter errors. You need to apply special considerations while deploying web applications on Microsoft Azure. Deploying a web application on Microsoft Azure provides advantages such as high availability and flexibility.

## ****Lesson Objectives****

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Ensure that dependencies are present for hosting ASP.NET MVC4 applications. |
| • | Deploy an ASP.NET MVC 5 web application to a single-server web farm. |
| • | Deploy an ASP.NET MVC 5 web application to a multi-server web farm. |
| • | Deploy an ASP.NET MVC 5 web application on Microsoft Azure. |

## ****ASP.NET MVC 5 Dependencies****

ASP.NET MVC 5 is built on the ASP.NET Framework. Each web application requires a range of different components to be present on the web server, to function appropriately. Examples of common requirements include:

|  |  |
| --- | --- |
| • | The ASP.NET Framework Common Language Runtime (CLR). The CLR runs any managed code such as C# classes. All ASP.NET web applications require the CLR, and you can install the CLR with Internet Information Services (IIS). |
| • | The ASP.NET MVC 5 runtime. The ASP.NET MVC 5 runtime locates controllers and actions to handle each request that the web application receives, and it returns the compiled webpages to users. You can install the MVC runtime in your web application by installing the MVC 5 NuGet package. All MVC web applications require the MVC runtime. |
| • | A Database server. Most web applications use a database to store information such as product details, customer details, images, and other entities. You need to ensure that the production environment supports the database that you used during development. For example, if you use SQL Compact during development, you need to ensure that Microsoft SQL Compact is installed on the production server. Alternatively, you can migrate the database to Microsoft SQL Server. |
| • | Entity Framework. If you use Entity Framework to model data in your web application, you need to ensure that you deploy the framework with your application. |
| • | Membership Providers. If your application uses membership providers to store user accounts and roles, you need to ensure that these providers are available on the server. The .NET Framework includes the SQL Server Membership Provider. If you use the Universal Membership Providers by installing the NuGet package, you need to ensure that you include the NuGet package with your deployed application. |

**Question:** Which of the common requirements in the above list is required to support forms authentication?

## ****Deploying Web Applications to Web Servers****

Before you deploy a web application to a production IIS web server, you must create and configure the IIS application and the folder that hosts the application.

**Creating an IIS Web Application**

You need to host ASP.NET MVC 5 web applications in an IIS web folder that is marked as a web application starting point. You can use the Internet Service Manager tool to configure this setting on the top-level folder of the web application. When you mark a web folder as a web application, you enable IIS to recognize that the content of the folder is ASP.NET and requires the ASP.NET runtime engine to process each request.

After creating an application, you need to configure the application to use .NET 4.0 or above. This practice enables ASP.NET MVC 5 to run, because the MVC runtime components are part of the .NET framework.

**IIS Application Pools**

In IIS, web applications are run within a context known as an application pool. You can run many applications in the same application pool or install an application in its own isolated pool. Applications that run in the same application pool share the following resources:

|  |  |
| --- | --- |
| • | Worker Process. The worker process uses a single execution process to run all the applications in a pool. |
| • | Memory. The applications share a single memory range, because the applications in a pool share a single worker process. |
| • | Identity. The applications within a single pool share a user account. |

You can place two or more applications in an application pool, and thereby reduce the total memory usage, because the applications share memory. However, this arrangement can reduce the reliability of each application, because a malfunctioning application can affect all the applications in the pool. To maximize the reliability of an application, you should install the application in an isolated pool that runs only one specific application.

The shared identity within an application pool affects the way the applications authenticate resources such as database servers when using integrated authentication. Each application within a pool shares a single user account. If each application has a separate database, you need to set permissions, to enable the user accounts to access all the databases. If you separate applications into isolated pools, you can use a different user account for each application. This practice can enhance the security of the application, because each account can access only one database.

**Deploying the Application**

When the IIS web application is ready, you need to copy the web application files to IIS. These files include all the model classes, controllers, views, script files, style sheets, images, and any other content in the Microsoft Visual Studio ASP.NET MVC 5 project. You can use the deployment tools available in Microsoft Visual Studio, or other technologies such as FTP, to copy these files. When the copy operation is complete, the web application is ready for deployment.

**Question:** What is the purpose of configuring additional application pools?

## ****Deploying Web Applications to Multi-Server Farms****

You can deploy a web application to a multi-server web farm to increase performance, resilience, and reliability. In a multi-server web farm, a group of two or more web servers host a single web application. All the servers in the farm share a single server name, such as www.contoso.com. Windows Network Load Balancing or dedicated hardware load balancers distribute browser requests to the servers.

A multi-server farm usually has greater capacity than a single web server, because the multi-server farm almost equally shares the load amongst several servers. Multi-server farms also increase reliability. When a server fails for any reason, the load balancing mechanism automatically directs requests to another server that remains online.

If you decide to deploy your web application to a multi-server farm, you need to perform the following steps. To complete these steps, you may need to work with the web server administrator who is responsible for the server farm:

|  |  |
| --- | --- |
| 1. | Create IIS applications and application pools on each server. This step is the same as configuring the applications and application pools in a single-server scenario. However, you need to perform this step on every server in the farm. You also need to ensure that applications and application pools have the same configuration, throughout the farm. |
| 2. | Create a matching IIS configuration on each server. IIS configuration options, such as any encryption certificates, file extensions, and optional components, should be identical on every server in the farm. |
| 3. | Use external hosted session state or session affinity. You can configure ASP.NET web applications to store session state in the Windows State Service or a database. You should use this technique in a multi-server farm, because each request from a single user session might be directed to a different web server. Alternatively, by configuring session affinity, you can ensure that all requests from a user session are always sent to the same web server. If you do not use external hosted session state or session affinity, the user preference on one web server may get lost when a request is sent to another web server. Session affinity is supported by some but not all load balancing solutions. Work with your web server administrator to determine if affinity is available on your farm. |
| 4. | Configure the **machineKey** element in the Web.config file. If you are using external hosted session state, you can encrypt the connection between the web servers and the State Service server or database server. This technique improves security by protecting session state during network transmissions. You can configure this encryption by setting up the **machineKey** element in the Windows registry. |

**Additional Reading:**For more information about configuring the **machineKey** element, go to: <http://go.microsoft.com/fwlink/?LinkID=288995&clcid=0x424>

**Question:** What is the purpose of configuring the **machineKey** element in the Web.config file?

## ****Deploying Web Applications on Microsoft Azure****

You can deploy any ASP.NET web application, including ASP.NET MVC 5 web applications, on Microsoft Azure. By choosing Microsoft Azure as a platform, you can use a highly-available and flexible infrastructure that you can optimize to cope with intense load. You have a choice of subscriptions, which can help you to pay only for the traffic that your web application uses.

Deploying a web application on Microsoft Azure is different from deploying the application on on-premise versions of IIS. To deploy an application on Microsoft Azure, you perform the following steps:

**Publishing To A New Web Application:**

|  |  |
| --- | --- |
| 1. | In Microsoft Visual Studio, open your web application solution. Then, start the Publish wizard and choose Azure Web App. |
| 2. | In the publishing wizard, give the new Web App a unique name. |
| 3. | Complete the Publish Wizard. |
| 4. | Observe that Microsoft Visual Studio publishes the web application to Microsoft Azure. |

**Publishing To An Existing Web Application**

|  |  |
| --- | --- |
| 1. | Create a new web application in the Microsoft Azure management portal. |
| 2. | In Microsoft Visual Studio, open your web application solution. Then, start the Publish wizard. |
| 3. | Select Azure Web App and make sure you select “Use Existing” |
| 4. | Ensure that you select the correct connection strings for Entity Framework connections, service database connections, and any other database connections. |
| 5. | Complete the Publish wizard. |
| 6. | Observe that Microsoft Visual Studio publishes the web application to Microsoft Azure. |

When the process is complete, you can access the Microsoft Azure-hosted web application from any Internet-connected browser.

**Note:** At the time of writing, Microsoft Azure web applications are still in preview, within the Microsoft Azure platform.

## ****Demonstration: How to Create a Microsoft Azure Web App****

In this demonstration, you will see how to create a new database and a new web app in Microsoft Azure.

### ****Demonstration Steps****

|  |  |
| --- | --- |
| • | You will find the steps in the “Lesson 1: Deploying a Web Application“ section on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD15_DEMO.md>. |

# Lesson 2: ****Deploying an ASP.NET MVC 5 Web Application****

Deploying ASP.NET MVC applications is similar to deploying other ASP.NET applications. This process is simple and straightforward. However, you need to update the configuration of the application for use in the production environment. Microsoft Visual Studio includes features that simplify updating the configuration of the application. You can also automate the deployment process by using deployment tools available in Microsoft Visual Studio.

## ****Lesson Objectives****

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Reconfigure an ASP.NET MVC application to deploy it on a production web server. |
| • | Use the Bin Deploy method to deploy the .NET Framework libraries along with your web application. |
| • | Use the deployment tools in Visual Studio. |

## ****Reviewing Configuration for Production****

Each ASP.NET MVC web application includes a Web.config file. In addition to this file, Microsoft Visual Studio creates two variance files to specify deployment transformations to the Web.config file. You can use these variance files for different build configurations. The variance files include:

|  |  |
| --- | --- |
| • | The Web .release.config file. This file stores the changes that Microsoft Visual Studio applies to the Web.config file, when you compile the application in the Release mode. |
| • | The Web .debug.config file. This file stores the changes that Microsoft Visual Studio applies to the Web.config file, when you compile the application in the Debug mode. |

Microsoft Visual Studio can automatically transform the Web.config file for use in different environments. Before you publish the application, you can remove the **debug** attribute from the Web.config file.

**Removing the Debug Attribute for Deployment**

<system.web>

<compilation xdt:Transform="RemoveAttributes(debug)" />

</system.web>

**Removing the Debug Attribute by Using a Deployment Transformation**

<system.web>

<compilation xdt:Transform="RemoveAttributes(debug)" />

</system.web>

The preceding code sample uses the **xdt:Transform** attribute to modify the existing Web.config file and produce the published Web.config file. You can also use the **Insert** value to add additional elements to the resultant Web.config file.

**Inserting a Connection String**

<connectionStrings>

<add name="DemoConnStr" connectionString="Data Source=|DataDirectory|demo.sdf"

providerName="System.Data.SqlServerCe.4.0" xdt:Transform="Insert"/>

</connectionStrings>

**Question:** How can you configure the Web.config file for publishing to a production environment without using the Web.release.config configuration file?

## ****Using Bin Deploy****

ASP.NET MVC applications depend on a range of .NET assemblies; these assemblies ensure that the application functions as required. These assemblies are dynamic-link library (DLL) files available in your project references. The assembly files are usually already present on the web server because they are included as part of the MVC library installation. However, sometimes the web server administrators do not install all .NET Framework assemblies. Microsoft Visual Studio provides the Bin Deploy feature to copy all depending assembly files into a folder within the deployed web application. This feature helps deploy dependencies on the server.

You can use the **Add Deployable Dependencies** feature to generate the \_bin\_deployableAssemblies folder. This folder includes the following libraries:

|  |  |
| --- | --- |
| • | Microsoft.Web.Infrastructure |
| • | System.Web.Helpers |
| • | System.Web.Mvc |
| • | System.Web.Razor |
| • | System.Web.WebPages |
| • | System.Web.WebPages.Deployment |
| • | System.Web.WebPages.Razor |

You can choose a specific set of libraries to include in Microsoft Visual Studio. This practice enables you to ensure that Microsoft Visual Studio deploys the:

|  |  |
| --- | --- |
| • | Right assemblies. |
| • | Right versions of the assemblies. |

You can then copy the library to the bin folder of the production server, after adding it to the library of your project.

**Question:** Why should you use the deployable assembly as part of the deployment plan?

## ****Using Visual Studio 2017 Deployment Tools****

The Publish feature helps generate a copy of the web application when the application is ready for deployment in the production environment. The feature includes three methods to deploy the application:

|  |  |
| --- | --- |
| • | File Share |
| • | FTP |
| • | Web Deploy |

On the Publish web application page, you can publish the application by using File Share or FTP. You can use File Share for servers to which you have direct network access. Alternatively, you can use File Share to publish a web application first to a local folder. Then, you can manually copy the files to the web server. File Share also allows you to specify a profile to store all deployment options. This practice eliminates the need to specify the deployment options every time you publish the application.

If you select **Web Deploy**, you can specify database settings, such as connection strings, in the publishing wizard. These settings will override the settings in the Web.config file in the target environment. Web Deploy also includes tools to automatically update the schema of the database. These tools eliminate the need to manually update the schema. The publishing tool detects all schema changes and generates scripts to apply those changes to the published database. If your development and production web servers are isolated from each other, you can run these generated scripts on the production environment. This practice replicates schema changes on a server where Microsoft Visual Studio is not installed.

The following image shows the Publish web application page.

**FIGURE 15.1: THE PUBLISH WEB APPLICATION PAGE**

**Question:** What is the benefit of using the Web Deploy publish tool?

## ****Demonstration: How to Deploy a Website to Microsoft Azure****

In this demonstration, you will see how to obtain a publish profile from Microsoft Azure and use it to publish a website from Visual Studio.

### ****Demonstration Steps****

|  |  |
| --- | --- |
| • | You will find the steps in the “Lesson 2: Deploying an ASP.NET MVC 5 Web Application “ section on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD15_DEMO.md>. |

# ****Lab: Deploying ASP.NET MVC 5 Web Applications****

### ****Scenario****

You have completed the development and testing of the photo sharing application. Your managers and senior developers have signed off the project, and have requested you to deploy the application to the Adventure Works Microsoft Azure account.

### ****Objectives****

After completing this lab, you will be able to:

|  |  |
| --- | --- |
| • | Prepare an MVC web application for deployment. |
| • | Deploy an MVC web application to Microsoft Azure. |

##### ****Lab Setup****

Estimated Time: 45 minutes

You will find the high-level steps on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD15_LAB_MANUAL.md>.

You will find the detailed steps on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD15_LAK.md>.

### ****Exercise 1: Deploying a Web Application to Microsoft Azure****

##### ****Scenario****

In this exercise, you will:

|  |  |
| --- | --- |
| • | Reconfigure the Photo Sharing application for release deployment. |
| • | Configure the **Entity Framework initializer** class, which fills the database with initial data, and ensure that the build configuration and connection strings are correct. |
| • | Create a new web application in Microsoft Azure and deploy the Photo Sharing application to the new site. |

### ****Exercise 2: Testing the Completed Application****

##### ****Scenario****

You have completed and fully deployed the Photo Sharing web application in Microsoft Azure. Now, you want to perform some final functionality tests before you confirm the completion of the application to your manager.

**Check Your Knowledge**

**Discovery**

**Why is it unnecessary to use bin deployment in this lab?**

Show solution Reset

**Check Your Knowledge**

**Discovery**

**In the labs for this course, you used the same Microsoft Azure SQL Database for both development and production. If you wanted to use separate databases for development and production, but did not want to reconfigure the web application every time you deployed to the development and production web servers, how would you configure the web application?**

Show solution Reset

# ****Module Review and Takeaways****

Deployment is usually the last task that developers perform; however, this is the task that developers do not spend much time on. The target environment usually impacts the deployment procedure. Microsoft Web Apps provide a developer friendly environment where the developer can leave the platform management to Microsoft Azure and focus on application development.

Visual Studio 2017 has a simple, yet powerful deployment tool that lets the developer to quickly and safely deploy applications to Microsoft Azure Web Apps.

### ****Review Question(s)****