Session: 15

Testing and Deploying

Objectives

- Define and describe how to perform unit tests
- Explain how to prepare an application for deployment
- Define and describe how to deploy an application on IIS

Testing

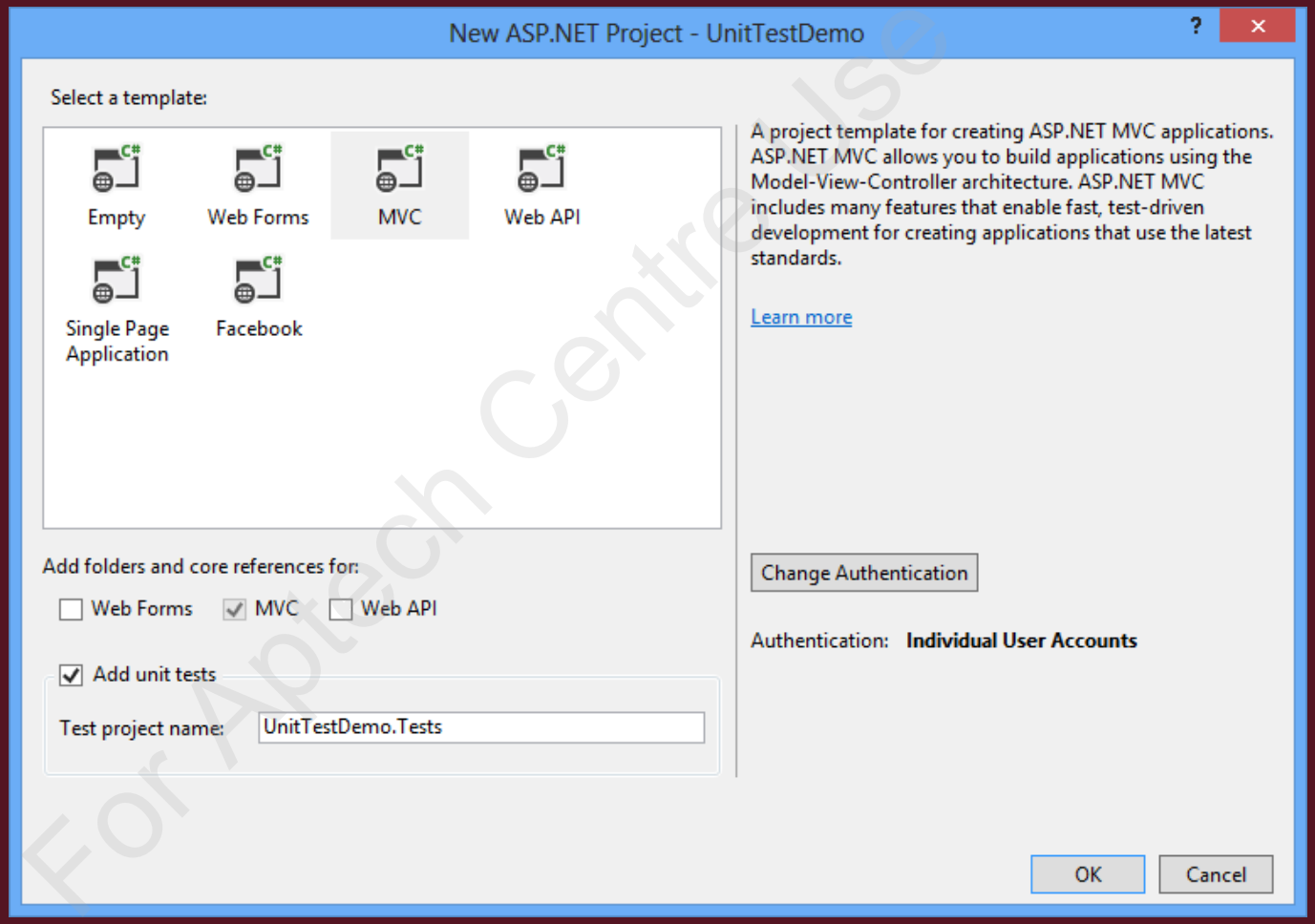
- Once you have created an application, you should ensure that the application works properly by testing it thoroughly to provide the best quality possible.
- Sometimes, an application may work properly at the time of development.
- However, when it is deployed, you cannot be sure of the inputs that the users may provide to the application and the results that the application may generate in such a scenario.
- To avoid such problems, you should test the application before you deploy it.

Preparing for Unit Tests 1-3

- Unit testing:
  - Is a technique that allows you to create classes and methods in your application and test their intended functionality.
  - Is the smallest part of an application that can be tested.
  - Is concerned whether or not the individual units that make up the application functions as expected.
- When you create an ASP.NET MVC application in Visual Studio 2013, you can specify whether to create a unit test for the application.
- To create an application in Visual Studio 2013 with unit test, you need to perform the following steps:
  - Create a new ASP.NET Web Application project, named UnitTestDemo.
  - Click OK. The New ASP.NET Project – UnitTestDemp dialog box appears.
  - Select MVC under the Select a template section and select the Add unit tests check box.

Preparing for Unit Tests 2-3

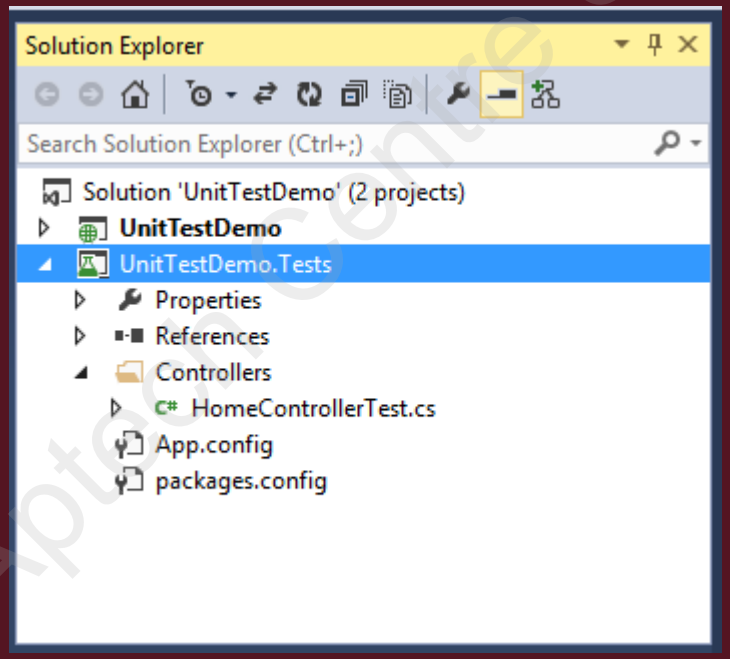
- Following figure shows the New ASP.NET Project – UnitTestDemo dialog box:



Preparing for Unit Tests 3-3

- Click OK. Visual Studio 2013 creates the project with support for unit testing. The Solution Explorer window displays a UnitTestDemo.Tests node that contains the resources to perform unit test on the application.

- Following figure shows Solution Explorer that displays the UnitTestDemo.Tests node:



- Click the HomeControllerTest.cs under the Controllers folder to open it.

Unit Test Classes and Methods 1-4

- A unit test class uses the [TestClass] attribute in the class declaration to indicate that it is a unit test class.
- Next, for each action method present in the target controller that needs to be tested, the unit test class has a corresponding method with the [TestMethod] attribute applied to it.
- To test an action method that passes data to the view using a ViewBag object, the test method uses the Assert.AreEqual() method.
- This method accepts the message being passed to the view as the first parameter and ViewResult.ViewBag.Message property as the second parameter.

Unit Test Classes and Methods 2-4

- Following code shows the unit test class, named HomeControllerTest:

Snippet

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web.Mvc;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using UnitTestDemo;
using UnitTestDemo.Controllers;
namespace UnitTestDemo.Tests.Controllers{
  [TestClass]
  public class HomeControllerTest
  {
     [TestMethod]
     public void Index()
      {
         // Arrange
         HomeController controller = new HomeController();
         // Act
         ViewResult result = controller.Index() as ViewResult;
```

Unit Test Classes and Methods 3-4

Snippet

```
// Assert
        Assert.IsNotNull(result);
    }
     [TestMethod]
     public void About()
    {
       // Arrange
        HomeController controller = new HomeController();
       // Act
       ViewResult result = controller.About() as ViewResult;
      // Assert
      Assert.AreEqual("Your application description page.",
      result.ViewBag.Message);
    }
```

Unit Test Classes and Methods 4-4

Snippet

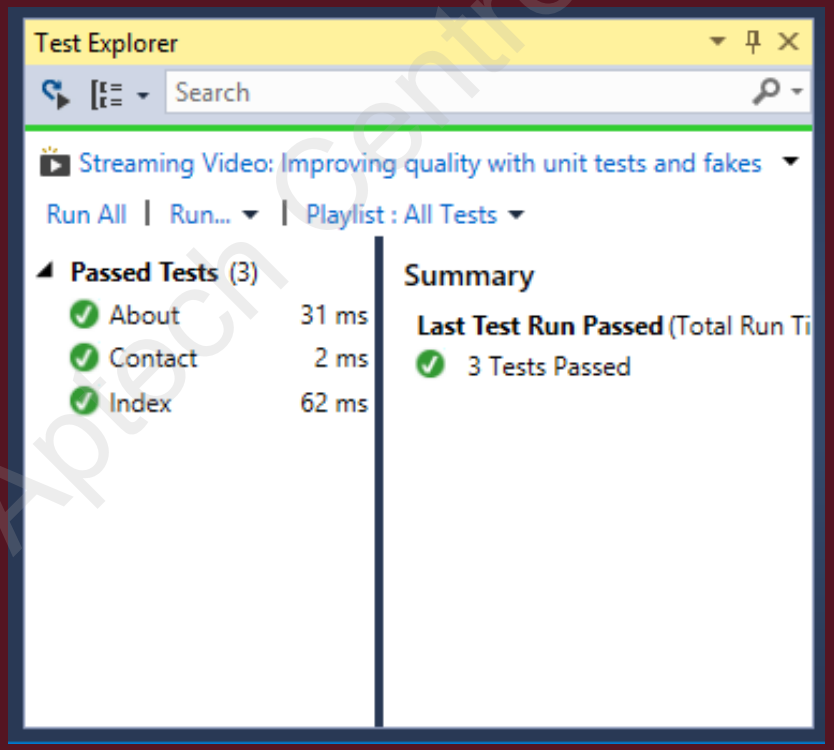
```
[TestMethod]
      public void Contact()
      {
        // Arrange
        HomeController controller = new HomeController();

        // Act
          ViewResult result = controller.Contact() as ViewResult;
        // Assert
          Assert.IsNotNull(result);
      }
   }
}
```

- This code uses the [TestClass] attribute in the class declaration and the [TestMethod] attributes for each action method that needs to be tested.

Performing Unit Tests

- To unit test the Home controller class using the HomeControllerTest unit test, you need to select Test→Run→All Tests in Visual Studio 2013. The Test Explorer window displays the test results.
- Following figure shows the Test Explorer window:



Preparing the Application for Deployment

- While developing and executing an ASP.NET MVC application using Visual Studio 2013, the application automatically deployed on Internet Information Server (IIS) Express.
- IIS Express makes deployment simple because it runs with your identity, and allows you to start and stop the Web server whenever required.
- To make your application accessible over the Internet, you need to host it on an IIS or any other Web server.
- Before deploying the application on a Web server, you first need to prepare the application for deployment.
- This process of preparing an application contains activities, such as identifying the files and folders to be copied on the Web server, configuring Web.config file, and precompiling the application.

Identifying the Files and Folders

- When you create an application using Visual Studio 2013, the application is saved on the path of the local computer that you can explicitly specify.
- Once you have created the application then, you need to deploy it on a Web server, such as IIS.
- To deploy the application on IIS, first you need to identify the files and folders that are created in the specified path and need to be copied to the destination server.
- While using Visual Studio 2013, by default it deploys only those files and folders that are required to run the application.
- However, sometimes there might be a requirement where you need to copy several other files and folders on the destination server.

Configuring the Web.config File 1-2

- Once you have deployed an ASP.NET MVC application on a Web server, some of the settings of the Web.config file vary in the deployed application.
- A transform file is associated with a build configuration.
- When you compile and execute an application in Visual Studio 2013, by default it creates the Debug and Release build configurations files named Web.Debug.config and Web.Release.config respectively.
- When you compile the application in Release mode, the Web.release.config file contains the changes that Visual Studio 2013 made in the Web.config file.
- On the other hand, when you compile the application in the Debug mode, the Web.debug.config file contains the changes that Visual Studio 2013 made in the Web.config file.

Configuring the Web.config File 2-2

- To publish the application using release configuration, you need to remove the debug attribute from the <compilation> element in the Web.config file.
- Following code snippet shows how to remove the debug attribute:

Snippet

```
<system.web>
<compilation xdt:Transform="RemoveAttributes(debug)" />
</system.web>
```

- In this code, the xdt:Transform attribute is used to remove the debug attribute from the Web.config file.

Precompilation 1-2

- To deploy an application, you need to copy the files and folders of the application to the hard drive of a Web server.
- In this process, most of the files are deployed to the Web server without compilation.
- Deploying an application in this way typically contains the following issues:
  - The application might get deployed with compilation errors
  - The source code of the application is exposed
  - The application loads slowly because files are required to be compiled when it is accessed for the first time

Precompilation 2-2

- Precompiling a Web application is a process that involves compilation of the source code into DLL assemblies before deployment.
- The process of precompiling provides the following advantages:
  - It provides faster response because the files of the application do not need to be compiled the first time it is accessed.
  - It helps in identifying the errors that can occur when a page is requested, because errors are rectified at the time of compiling the application.
  - It secures the source code of the application from malicious users.

Deploying on IIS

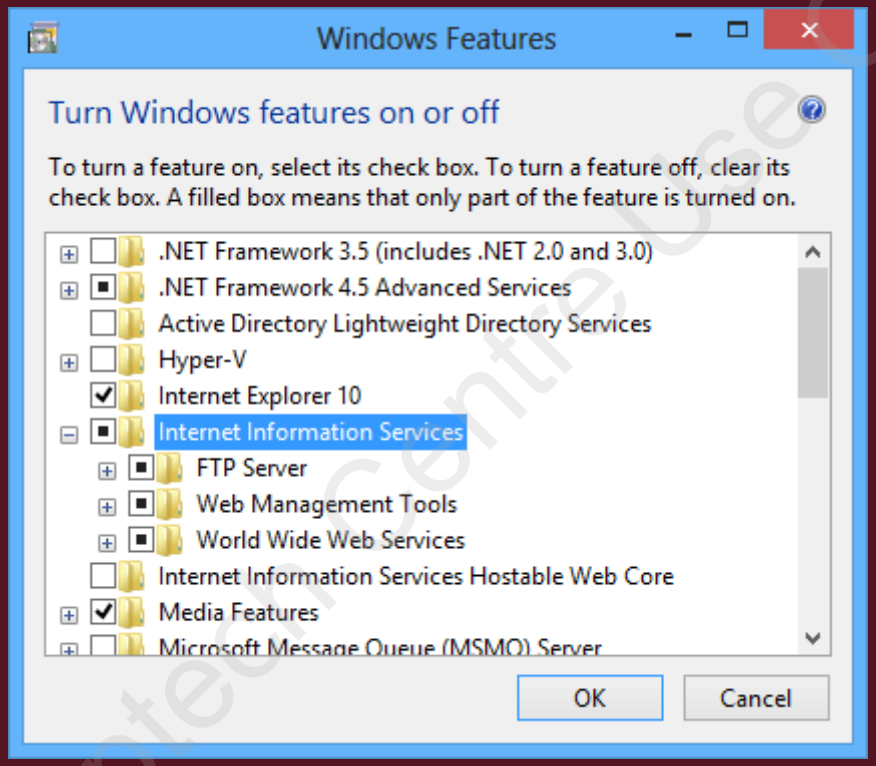
- IIS is a Web server that allows you to develop, host, and manage your application.
- Before deploying an application on IIS, you first need to install it on your computer.
- To deploy an ASP.NET MVC application, you need to perform the following tasks:
  - Install IIS
  - Create an application in IIS
  - Create a publish profile for the application
  - Publish the project

Installing IIS 1-2

- The first step to deploy an application is to install IIS.
- To install IIS, you need to perform the following tasks:
  - Open Control Panel.
  - Click Uninstall a program under the Programs icon. The Program and Features window displays the Uninstall or change a program screen.
  - Click the Turn Windows features On or Off link in the left pane. The Windows Features window is displayed.
  - Ensure that all the check boxes are selected under the Internet Information Services node.

Installing IIS 2-2

- Following figure shows the Windows Features window:



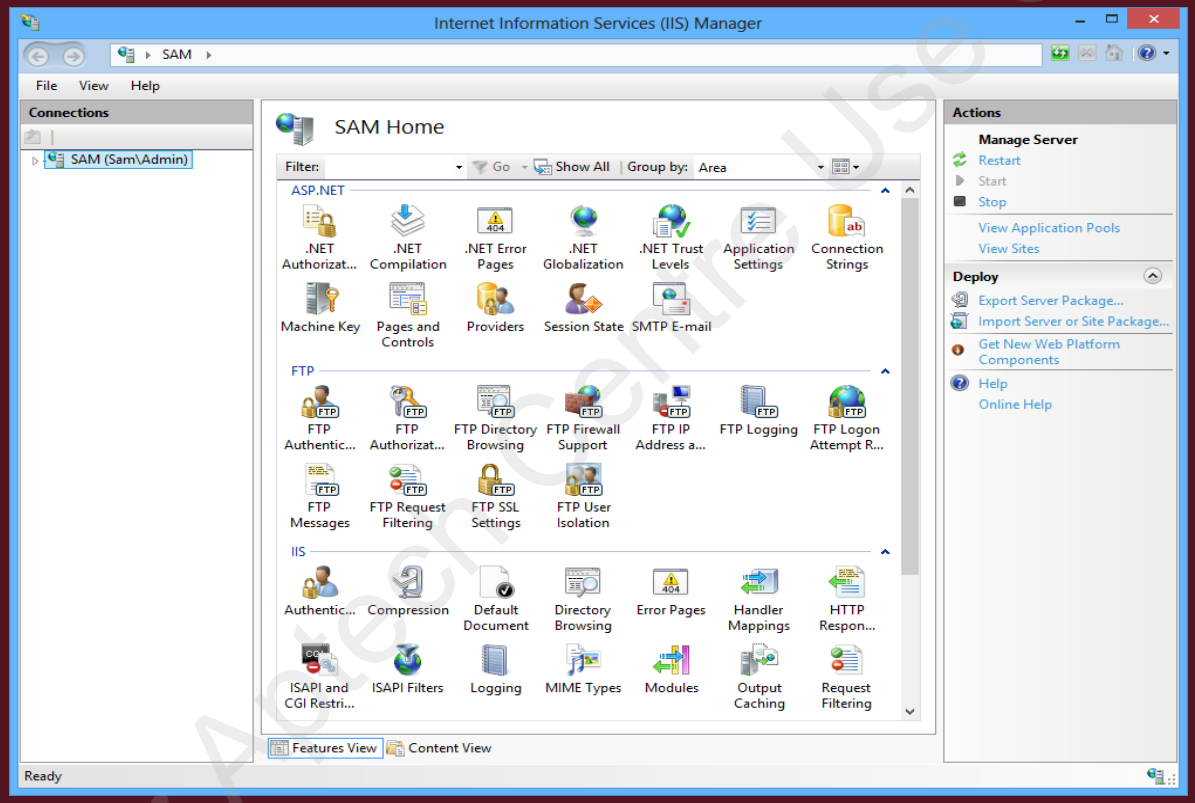
  - Click OK. The Windows Features message box is displayed.
  - Wait until the changes are applied to the system.
  - Click the Close button.
  - Close the Programs and Features window.

Creating an Application on IIS 1-4

- Once you have installed IIS, the next step that you need to perform is creating a publish profile.
- To create an application on IIS, you need to perform the following steps:
  - Create a folder with the name of your project, for example MVCDemo in the C:\inetpub\wwwroot folder.
  - Press the Windows+R keys. The Run dialog box is displayed.
  - Type inetmgr in the Run dialog box.
  - Click OK. The Internet Information Services (IIS) Manager window is displayed.

Creating an Application on IIS 2-4

- Following figure shows the Internet Information Services (IIS) Manager window:

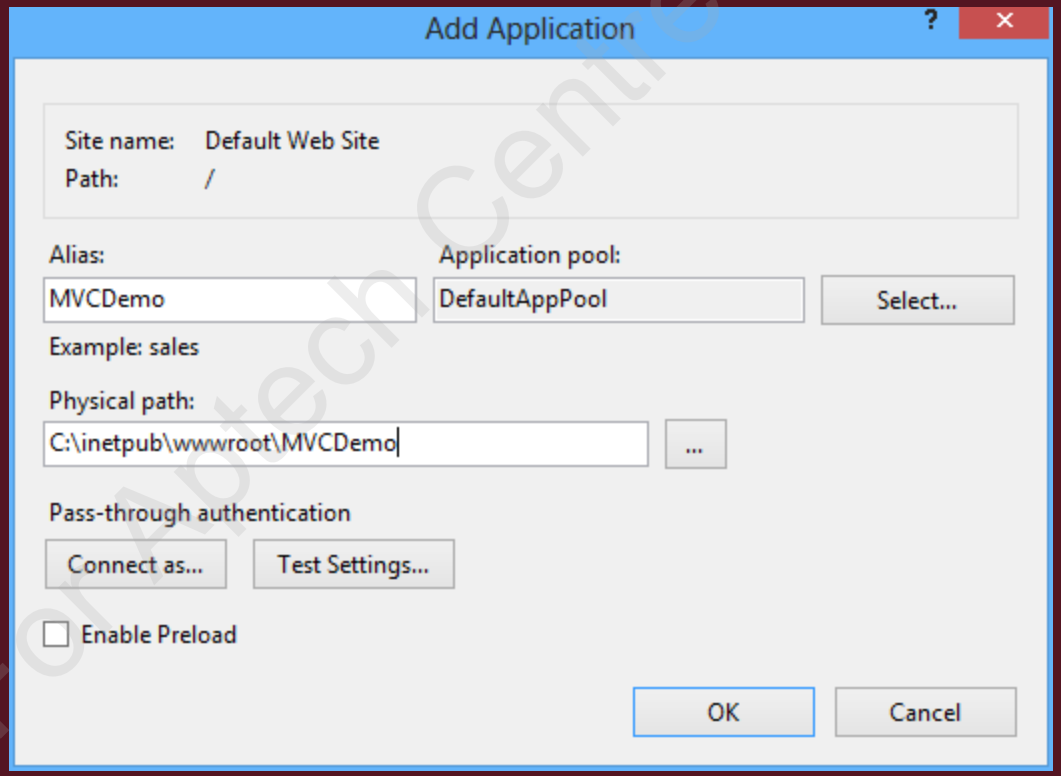


  - Expand the node under the Connections pane.
  - Expand the Sites node.

Creating an Application on IIS 3-4

- Right-click the Default Website node under the Sites node, and then, select Add Application. The Add Application dialog box is displayed.
- In the Add Application dialog box, type MVCDemo in the Alias text field and type C:\inetpub\wwwroot\MVCDemo in the Physical path text field.

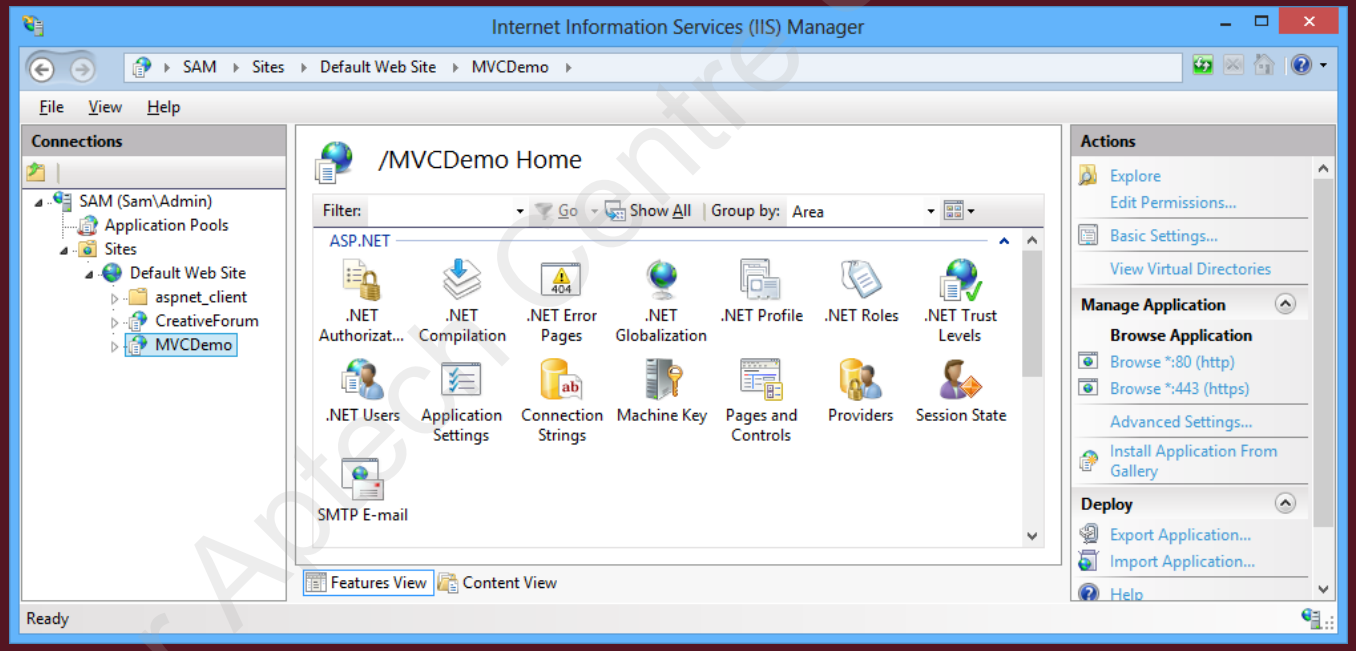
- Following figure shows the Add Application dialog box:



Creating an Application on IIS 4-4

- Click the OK button in the Add Application dialog box. The MVC node is displayed under the Connection node of the Internet Information Services (IIS) Manager window.

- Following figure shows the MVC node under the Connection node:



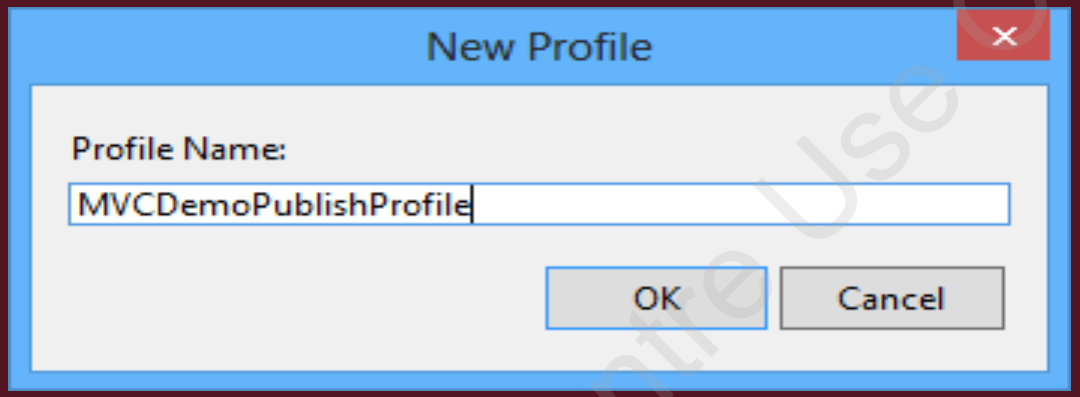
- Close the Internet Information Services (IIS) Manager window.

Creating a Publish Profile 1-5

- After creating the application on IIS, you need to create a publish profile.
- A publish profile represents various deployment options, such as the target server to be used for deployment, the credentials needed to log on to the server to deploy.
- To create a publish profile in Visual Studio 2013, you need to perform the following tasks:
  - Start Visual Studio 2013 with Administrator privilege.
  - Open the MVCDemo project to publish.
  - Right-click the project in the Solution Explorer window and select Publish. The Publish Web dialog box is displayed.
  - From the Select or import a publish profile drop-down list, select the <New…> option.
  - The New Profile dialog box is displayed.
  - Type MVCDemoPublishProfile in the Profile Name text field.

Creating a Publish Profile 2-5

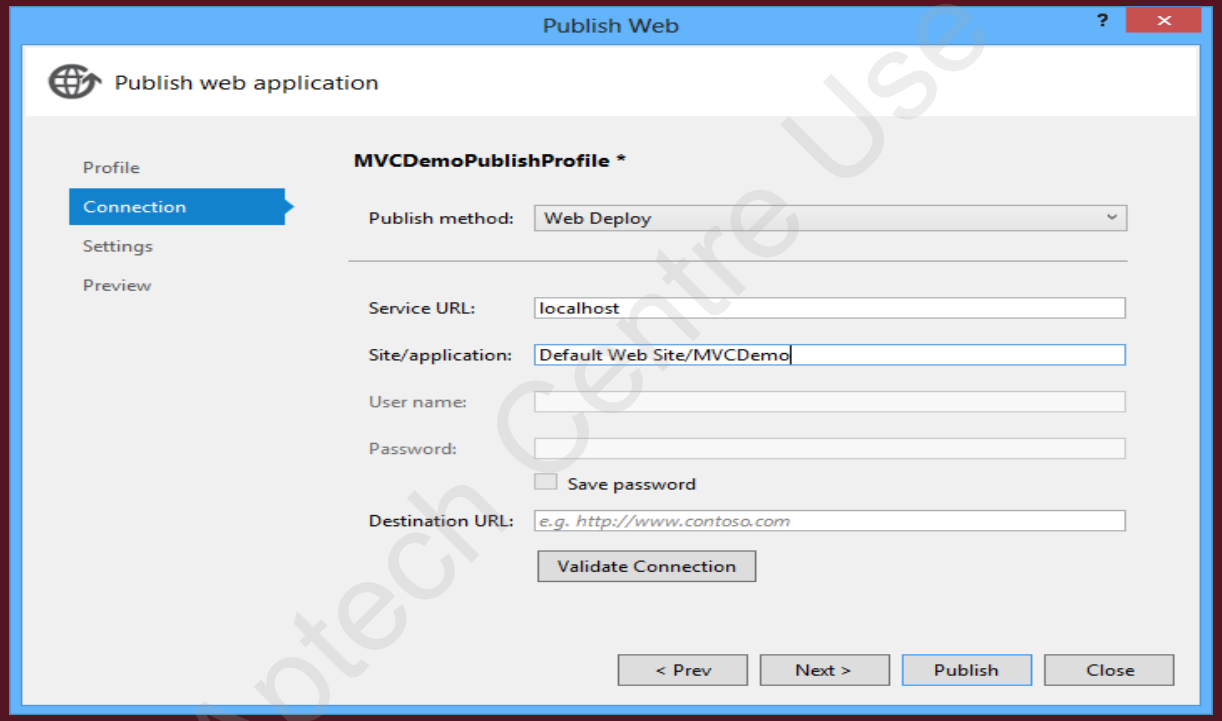
- Following figure shows the New Profile dialog box:



  - Click OK. The Publish Web dialog box is displayed with the specified profile name.
  - Ensure that Web Deploy is selected in the Publish method drop-down list.
  - In the Service URL text field, type localhost and in the Site/application text field, type Default Web Site/MVCDemo.

Creating a Publish Profile 3-5

- Following figure shows the specifying the Service URL and Site/application text fields:

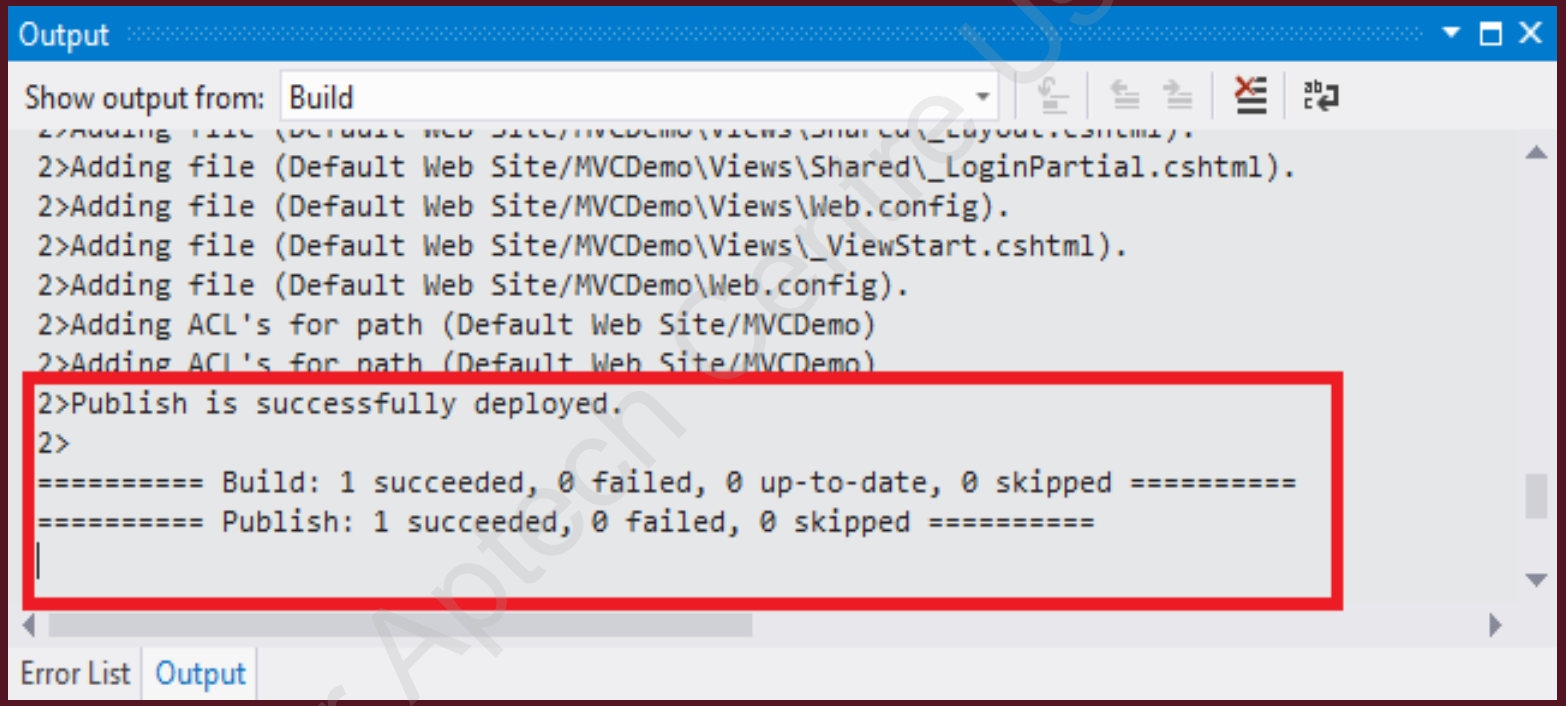


  - Click Validate Connection. When the connection is valid a Correct mark is displayed by the side of the Validate Connection button.
  - Click Next. The Publish Web dialog box is displayed.

Creating a Publish Profile 4-5

- Click Publish. The Output window of Visual Studio 2013 displays a message to indicate that the project has been successfully published.

- Following figure shows the Output window:

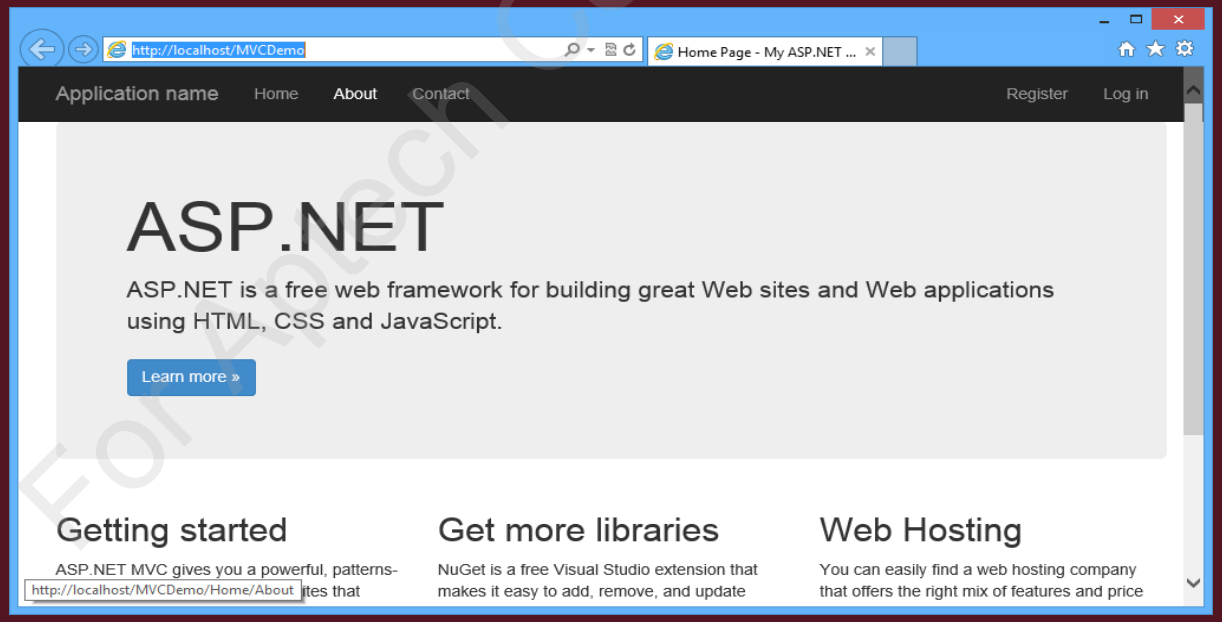


Creating a Publish Profile 5-5

- To test the published application type the following URL in the address bar of the browser.

  http://localhost/MVCDemo

- The Home page of the application published on IIS is displayed on the browser.
- Following figure shows the Home page of the published application on the browser:



Summary

- Unit testing is a technique that allows you to create classes and methods in your application and test their intended functionality.
- While developing and executing an ASP.NET MVC application using Visual Studio 2013, the application is automatically deployed on IIS Express.
- To deploy the application on IIS, first you need to identify the files and folders that are required to be copied on the destination server.
- While using Visual Studio 2013, by default it deploys only those files and folders that are required to run the application.
- Precompiling a Web application is a process that involves compilation of the source code into DLL assemblies before deployment.
- To deploy an ASP.NET MVC application, you need to install IIS, create an application in IIS, create a publish profile for the application, and finally publish the project.
- A publish profile represents various deployment options, such as the target server to be used for deployment, the credentials needed to log on to the server to deploy.