* 1. 

Hands-On Lab

Web Development in Visual Studio 2010

* 1. Lab version: 1.1.0
  2. Last updated: 1/6/2011
  3. 

Contents

[Overview 3](#_Toc282089415)

[Exercise 1: Using HTML Code Snippets in Visual Studio 2010 6](#_Toc282089416)

[Task 1 – Adding a New ListView to an HTML Page Using an HTML Code Snippet 6](#_Toc282089417)

[Task 2 – Creating a Custom HTML Snippet 9](#_Toc282089418)

[Exercise 1: Verification 15](#_Toc282089419)

[Exercise 2: Transforming a Web.Config File for Deployment 18](#_Toc282089420)

[Task 1 – Creating a Staging Configuration in Visual Studio 2010 18](#_Toc282089421)

[Task 2 – Adding Code to Populate Text Value of EnvName from a Value in the Web.config 21](#_Toc282089422)

[Task 3 – Adding a Logging Database Connection String to the Web.config File 23](#_Toc282089423)

[Task 4 – Writing a Transform to Change the Environment and Logging Connection Strings in the Staging Web.config 24](#_Toc282089424)

[Task 5 – Generating a Transformed Web.config File from the Command Line 25](#_Toc282089425)

[Task 6 – Generating a Transformed Web.config File from Visual Studio 28](#_Toc282089426)

[Exercise 2: Verification 31](#_Toc282089427)

[Exercise 3: Packaging and Deploying Web Applications for the Visual Studio Development Web Server 32](#_Toc282089428)

[Task 1 – Opening the Project Properties Publish Page and Creating a Package 33](#_Toc282089429)

[Task 2 – Preparing the Web Server for Deployment 36](#_Toc282089430)

[Task 3 – Deploying the Package 39](#_Toc282089431)

[Exercise 3: Verification 43](#_Toc282089432)

[Exercise 4: Packaging and Deploying Web Applications for Internet Information Server (IIS) 45](#_Toc282089433)

[Task 1 – Opening the Project Properties Publish Page, Setting the Web Application to Use IIS and Create a Package. 46](#_Toc282089434)

[Task 2 –Deploying the Package 49](#_Toc282089435)

[Exercise 4: Verification 54](#_Toc282089436)

[Summary 55](#_Toc282089437)

Overview

Microsoft Visual Studio 2010 offers many new features for development of ASP.NET web applications. The goal of these new features is to aid developers in creating and deploying high quality and fully featured web applications quickly and easily.

Visual Studio 2010 now supports HTML code snippets. Snippets are a great way to generate high-quality code. There are many snippets that come with Visual Studio, and developers have the ability to create their own.

Another new feature which Visual Studio 2010 provides is the ability to transform the Web.config file for deployment. Visual Studio 2010 supports a rich configuration transformation language that allows developers to change elements in the Web.config file including connection string, file path locations, and service addresses.

Deployment of ASP.NET web applications is now easier due to new packaging and deployment features in Visual Studio 2010. Now web applications can be easily published to a specific location on the build machine (or even to a ZIP file) and easily deploy to either the Visual Studio Web Server or Windows Internet Information Server (IIS).

# Objectives

* 1. In this Hands-On Lab, you will learn how to:
  + Use HTML code snippets to aid in development of ASP.NET web application pages
  + Create and deploy your own custom HTML code snippets
  + Create and automatically transform the Web.config as the web application is deployed across environments
  + Create a deployment package for your ASP.NET web application and deploy the application to either the Visual Studio Development web server or Windows Internet Information Server (IIS)

# Exercises

* 1. This Hands-On Lab is comprised by the following exercises:
  + Using HTML Code Snippets in Visual Studio 2010
  + Transforming a Web.config file for deployment
  + Packaging and deploying a web application to the Visual Studio Development Web Server
  + Packaging and deploying a web application to the Windows Internet Information Server (IIS)
  1. Estimated time to complete this lab: **60 minutes**.

# System Requirements

* 1. You must have the following items to complete this lab:
  + Microsoft Visual Studio 2010
  + IIS 7.0

# Setup

* 1. All the requisites for this lab are verified using the **Configuration Wizard**. To make sure that everything is correctly configured, follow these steps.
  2. Run the **Configuration Wizard** for the Training Kit if you have not done it previously. To do this, browse to **Source\Setup** folder of this lab, and double-click the **Dependencies.dep** file. Install any pre-requisites that are missing (rescanning if necessary) and complete the wizard.
     1. **Note:** The Configuration Wizard is used for checking dependencies and setting up the environment. If the Configuration Wizard is not installed on your machine, you must install it running the DependencyChecker.msi file located on the %VS2010TKInstallationFolder%\Assets\DependencyChecker folder (e.g. by default the Training Kit is installed under C:\VS2010TrainingKit).

# Starting Materials

* 1. This Hands-On Lab includes the following starting materials:
  + Visual Studio solutions**:** The exercise provides a Visual Studio solution that you can use as starting point for the lab exercise. You will find the starting point at: ***%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab***
    - HTMLLab.sln: A solution containing a simple web application used as a starting point for Exercise 1 and the subsequent exercises.
      1. **Note:** For each exercise, there is an End folder which contains the resulting solution you should obtain after completing that exercise.
    1. **Note:** User will perform appropriate name value replacement in all path references containing <Username> tag.
    2. To verify that each step is correctly performed, it is recommended that you build the solution at the end of each task.

# Next Step

Exercise 1: Using HTML Code Snippets in Visual Studio 2010

Exercise 1: Using HTML Code Snippets in Visual Studio 2010

* 1. In this exercise, you will learn how to use HTML code snippets in Visual Studio 2010. You will also learn how to create your own custom HTML snippets.

Task 1 – Adding a New ListView to an HTML Page Using an HTML Code Snippet

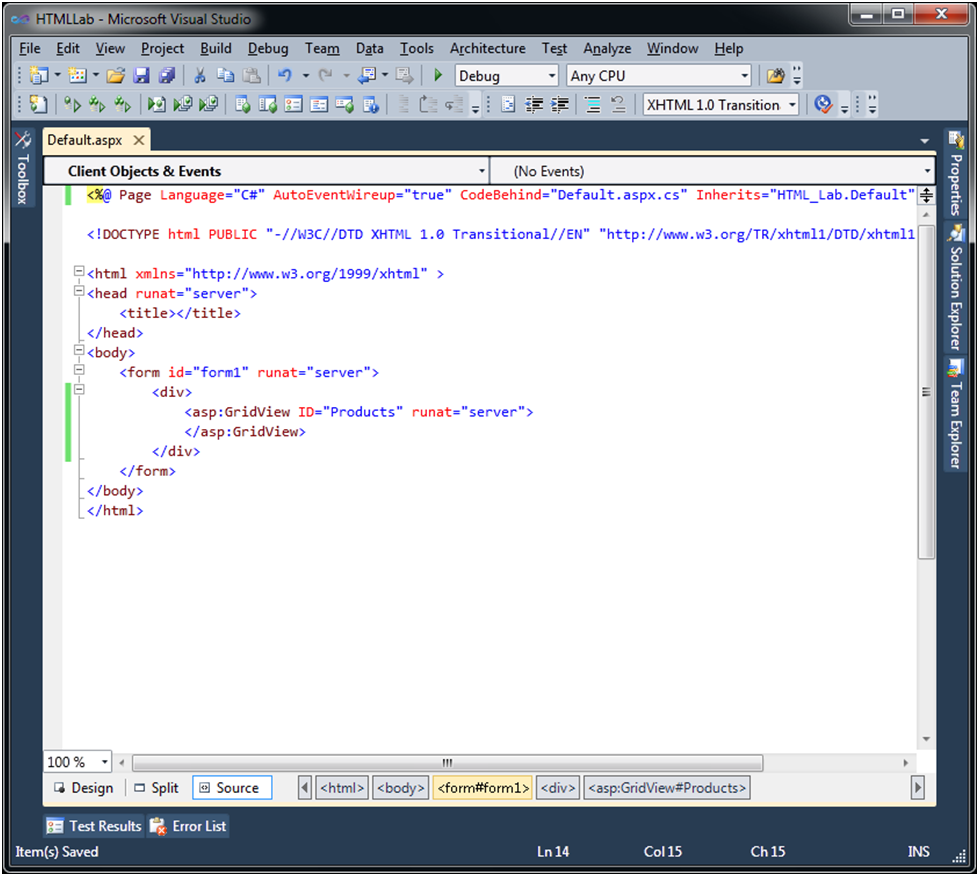
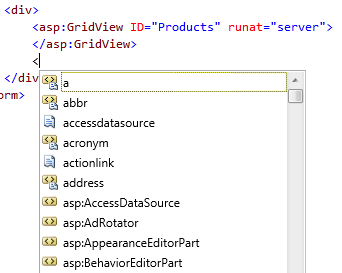
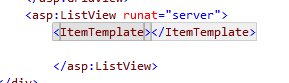
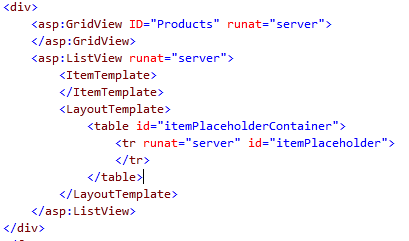
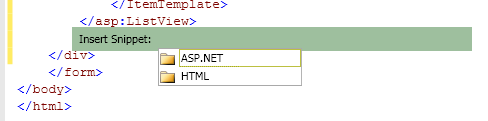
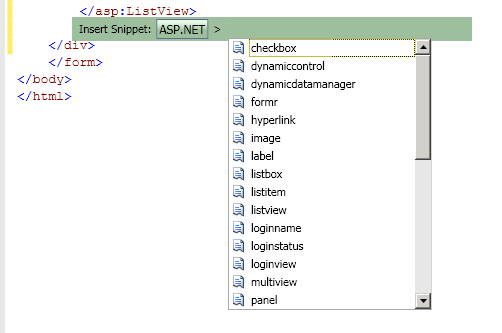
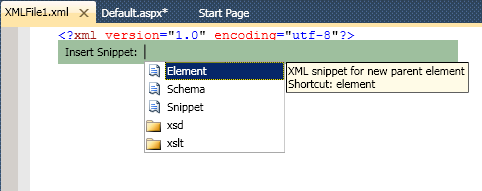
* 1. In this task, you will open an existing ASP.NET web application and use a snippet to add a ListView control to an existing webpage.
  2. Navigate to **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Microsoft Visual Studio 2010**. Select the **File** | **Open** | **Project/Solution…** menu command. In the **Open Project** dialog, navigate to ***\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab*** and select HTMLLab.sln file.
  3. In Solution Explorer, open the **Default.aspx** file by double-clicking it. Examine the code present on the page. Note that the page contains a **GridView** control with an **ID** of **Products**.
     1. 
     2. Figure 1
     3. Existing Grid View control in Default.aspx
  4. On the Default.aspx page, you will add a **ListView** control. Place the cursor after closing GridView tag (**</asp:GridView>**) and press the **ENTER** key. Press the “**<**” key to invoke the completion list.
     1. 
     2. Figure 2
     3. Completion List
     4. **Note:** The small icons next to each completion indicate what type of completion it is:
     5. s2a.png Indicates that a completion entry is also a snippet keyword
     6. s2b.png Indicates a snippet keyword
     7. s2c.png Indicates a completion entry
  5. Select **ListView** from the completion list and press the **TAB** key twice to insert the snippet.
     1. 

Figure 3

* + 1. The inserted snippet
  1. To progress you will need to add the layout, to put in a temporary placeholder. Enter the following code to define the **LayoutTemplate**:
     1. XML
     2. **<LayoutTemplate>**
     3. **<table id="itemPlaceholderContainer">**
     4. **<tr runat="server" id="itemPlaceholder">**
     5. **</tr>**
     6. **</table>**
     7. **</LayoutTemplate>**
     8. 
     9. Figure 4
     10. The completed snippet
     11. **Note:** In this task, you brought up the completion list by typing the “<” character. Another way to show that list is to use the CTRL+J shortcut. Yet another option is to invoke the snippet picker by using the CTRL+K, CTRL+X shortcut. This may be faster in some cases because you can narrow the snippets by group as shown below:
     12. 
     13. By selecting the ASP.NET folder and pressing the TAB key, you will see a list of the ASP.NET specific snippets:
     14. 
     15. You can abort this action by pressing the ESC key. Do not add any more controls to this page at this time.

Task 2 – Creating a Custom HTML Snippet

* 1. In this task, you will create a custom HTML snippet for use inside of the Visual Studio 2010 IDE. The new code snippet will create a label control with the preset default values. Any custom Snippets implemented are stored as XML files in a well-known directory.
  2. In Visual Studio 2010 select **File** | **New** | **File** from the menuto open the **New File** dialog. In the **New File** dialog select **General** from Categories, **XML File** from Templates and click **Open**.
  3. To create the structure of the snippet you will use an existing snippet. In the XML file editor, press **CTRL+K**, **CTRL+X** to display the snippets list.
     1. 
     2. Figure 5
     3. Snippets list
  4. Select **Snippet** from the list. An XML code snippet will be inserted in the file.
     1. 
     2. Figure 6
     3. XML definition of a code snippet list
  5. Supply the following values for the indicated XML elements:

|  |  |
| --- | --- |
| Field | Value |
| Title | “Address Block” |
| Shortcut | “AddrBlock” |
| Description | “XML snippet to create a quick address block.” |

* + 1. Snippet picker uses the **Title** value to display the name of the snippet. Completion list uses the **Shortcut** value to select the snippet.
  1. Under the **<SnippetTypes>** node, delete the first **<SnippetType>** XML element containing the value **SurroundsWith**. Be sure to leave the **<SnippetType>** XML element that contains the value **Expansion**.
  2. Delete the entire **Declarations** XML node.
  3. Delete the **Code** XML node, including the CDATA value within it from the Snippet XML element.
  4. Delete the XML element at the top of the file. When finished with these steps, your snippet should look like this:
     1. XML
     2. <CodeSnippet Format="1.0.0" xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
     3. <Header>
     4. <Title>Address Block</Title>
     5. <Author>author</Author>
     6. <Shortcut>AddrBlock</Shortcut>
     7. <Description>XML Snippet to create a quick address block.</Description>
     8. <SnippetTypes>
     9. <SnippetType>Expansion</SnippetType>
     10. </SnippetTypes>
     11. </Header>
     12. <Snippet>
     13. </Snippet>

</CodeSnippet>

* 1. In the **Snippet** element, add the following code (enclosing **Snippet** tags shown for guidance).
     1. XML
     2. <Snippet>
     3. **<Code Language="html">**
     4. **<![CDATA[<label for="CustomerType">Customer Type:</label>**
     5. **<select>**
     6. **<option>Federal</option>**
     7. **<option>State</option>**
     8. **<option>Corporate</option>**
     9. **<option>Residential</option>**
     10. **</select>**
     11. **<br />**
     13. **<label>Name: </label>**
     14. **<input id="name" name="name"/><br />**
     16. **<label >Address Line 1: </label>**
     17. **<input id="AddressLine1" name="AddressLine1"/><br />**
     19. **<label>Address Line 2: </label>**
     20. **<input id ="AddressLine2" name="AddressLine2"/><br />**
     22. **<label>City </label> <input id="City" name="City"/><br />**
     24. **<label>State </label><input id="State" name="State"/><br />**
     26. **<label >Zip Code </label> <input id="zip" name="zip"/><br /><br />**
     28. **$end$]]>**
     29. **</Code>**

</Snippet>

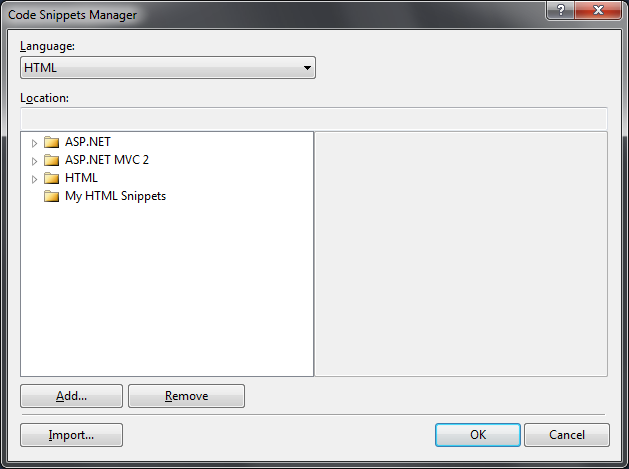
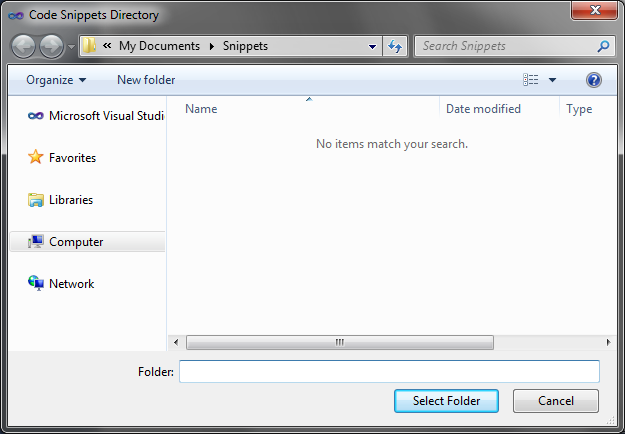
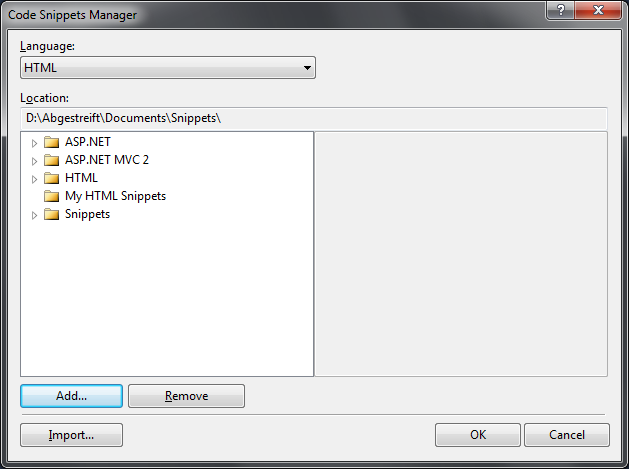
* + 1. This will add a label control with the ID of **AddrBlock**.
  1. Using **Windows Explorer**, navigate to C:\Users\<Username>\Documents\ folder. Create a sub-folder “Snippets” in the C:\Users\<Username>\Documents\ folder.

**Note:** There are two ways to deploy code snippets to the environment.

Option 1: Save the XML as a .snippet file to the C:\Users\<Username>\Documents\Visual Studio 10\Code Snippets\Visual Web Developer\My HTML Snippets\ folder. Visual Studio will automatically find these snippets and make them available for use.

Option 2: Save created snippet XML as a .snippet file in some other location and use the **Code Snippets Manager** to add the snippet to Visual Studio.

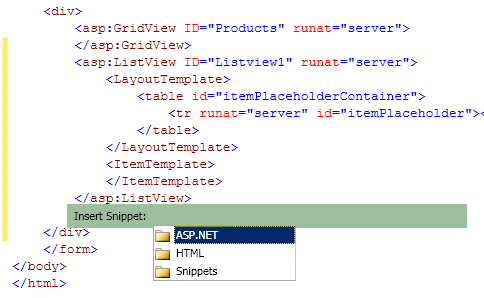
Snippets added using the **Code Snippets Manager** will not be available in the auto complete list.

* 1. Select the **File** | **Save As** menu command. In the **Save File As** dialog, change the path to C:\Users\<Username>\Documents\Snippets folder. Change the file name to “AddrBlock”, set the **Save as type** dropdown list to **Snippet Files (\*.snippet)** and click **Save**.
  2. Open the **Code Snippets Manager** by using the **Tools** | **Code Snippets Manager…** menu or the CTRL + K, CTRL + B shortcut. **The Code Snippets Manager** dialog appears.
     1. 
     2. Figure 7
     3. Code Snippets Manager
  3. In the **Language** dropdown list, select **HTML**. Click the **Add** button to open a **Code Snippet Directory** dialog.
     1. 
     2. Figure 8
     3. Add a code snippets folder
  4. In the **Code Snippet Directory** dialog, navigate to the folder you created (C:\Users\<Username>\Documents\Snippets\) and click the **Select Folder** button. The **Snippets** folder appears in the list of snippet locations. Click the **OK** button to dismiss the dialog.
     1. 
     2. Figure 9
     3. Snippets folder added to list of snippet locations

# Next Step

Exercise 1: Verification

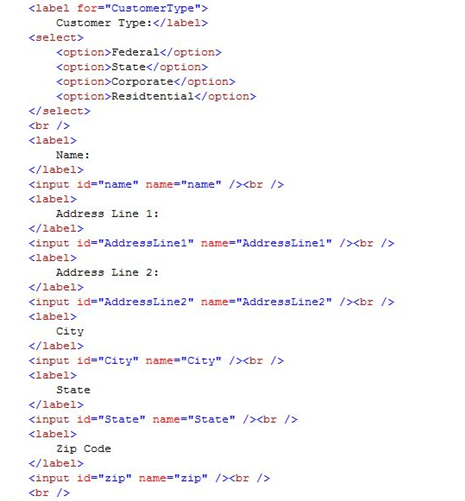
### Exercise 1: Verification

* 1. You will verify that you can use the custom HTML snippet from Visual Studio 2010 and that the snippet renders the Label control correctly.
  2. Open the source view of the Default.aspx page in the **HTMLLab** Web Application. To do this, right-click Default.aspx, and select **View Markup**.
  3. Place the cursor at the end of XML tag of the **ListView** with the Id **Listview1** and press **ENTER** to create a new line.
  4. Use the **CTRL+K**, **CTRL+X** shortcut to open the snippets list. Notice that the **Snippets** folder you included in the **Code Snippets Manager** is now also in the list.
     1. 
     2. Figure 10

Snippets list with the Snippets folder included

* 1. Using the mouse or the down arrow, select **Snippets** from the list and press the **TAB** key to see the list of snippets available in that folder.
     1. 
     2. Figure 11

List of snippets in the Snippets folder

* 1. Using the mouse or the down arrow key, select the **Address Block** snippet and press the **TAB** key. The snippet will expand and add your **Address Book** element to the page:
     1. 
     2. Figure 12
     3. The AddressBlock snippet expanded into the page

**Note:** You did not specify any user input for any of the elements of the snippet. Use the Declaration section to specify the behavior of the snippet. Each specified field will be highlighted; the user can TAB between fields, and will press **ENTER** when finished.

# Next Step

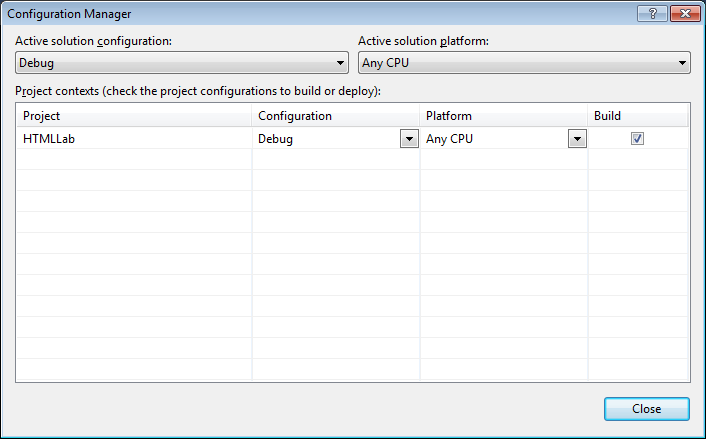
Exercise 2: Transforming a Web.Config File for Deployment

Exercise 2: Transforming a Web.Config File for Deployment

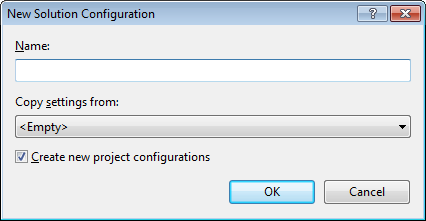
* 1. Generally, a web application goes through a chain of environments before it makes it into a Production environment. Some of these environments are Development (the developer’s local machine), Quality Assurance (QA), and User Acceptance Testing (UAT) / Staging / Pre-Production. As applications transition through these environments, various settings in the configuration files must change. In addition to database connection strings, many Enterprise scale applications also rely on configuration settings for logging destinations, file drop shares and addresses for service endpoints.
  2. Visual Studio 2010 adds a new Web.config transformation model that you can use to automate changes to the Web.config file during the deployment of the application when you use MSDeploy to promote your code.

**Note:** To complete the tasks in this exercise, you must have completed all the tasks in Exercise 1, or use the solution from the ***%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex02-TransformingWebConfig\begin\C#\HTMLLab*** folder.

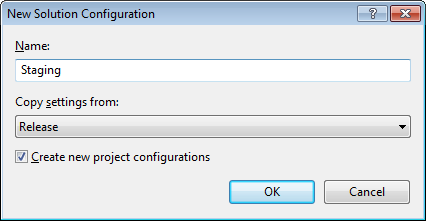
Task 1 – Creating a Staging Configuration in Visual Studio 2010

* 1. Open the **HTMLLab** Web Application completed at the end of the previous exercise.
  2. Select **Build** | **Configuration Manager** from the menu. The **Configuration Manager** dialog appears:
     1. 
     2. Figure 13

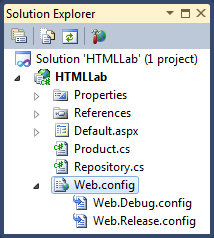
Configuration Manager

* 1. Select **<New…>** from the **Active solution configuration** dropdown list to bring up the **New Solution Configuration** dialog.
     1. 
     2. Figure 14

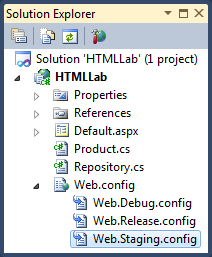
New Solution Configuration

* 1. In the **Name** field type “Staging” and from the **Copy setting from** dropdown list select **Release**. Leave the **Create new project configurations** checkbox checked. Your dialog should look similar to the figure below:
     1. 
     2. Figure 15

New Solution Configuration for the new Staging configuration

* 1. Click the **OK** button. Click the **Close** button to close the **Configuration Manager**.
  2. In the **Solution Explorer**, notice that there is an expandable node next to the Web.config file. Click the node to expand the tree view. Notice that there are two files under Web.config that follow a *Web.<configuration>.config* naming convention.
     1. 
     2. Figure 16

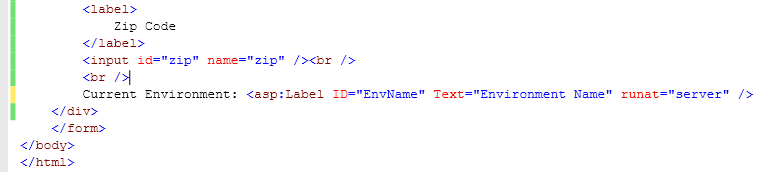
Debug and Release configuration files

* + 1. The changes between the environments are stored in these delta files. By default, all web applications start with a Debug and Release configuration file. Visual Studio can add a new transformation for new configurations that developers create.
  1. Add a new Web.config transformation. To do this, right-click on the Web.config file and from the menu select **Add Config Transforms**. Notice that new Staging configuration file is automatically added:
     1. 
     2. Figure 17

Staging configuration added

**Note:** If from the right-click menu the **Add Config Transforms** is disabled, do a full rebuild of the application.

Task 2 – Adding Code to Populate Text Value of EnvName from a Value in the Web.config

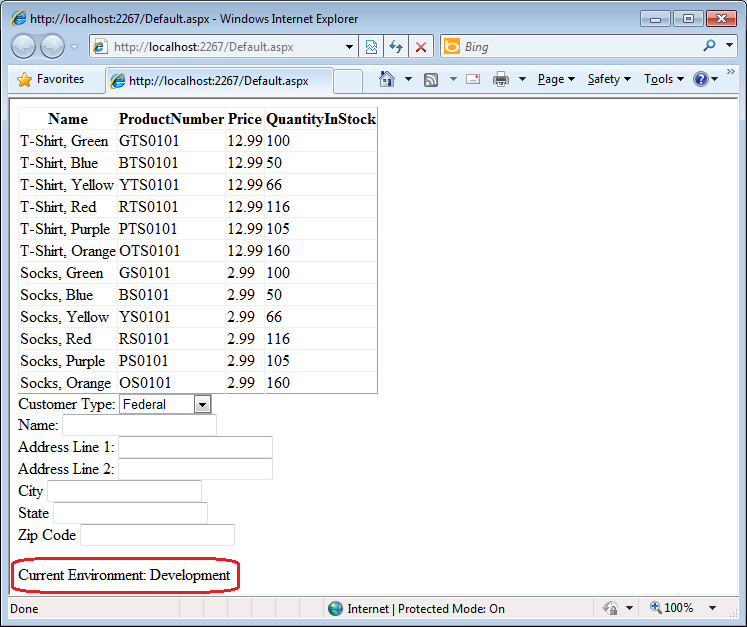
* 1. Before we start this task, you will need to add a label to the Default.aspx page to contain an environment variable. Here’s what the label should look like when you are done.
  2. ASP.NET
  3. **Current Environment: <asp:Label ID="EnvName" Text="Environment Name" runat="server" />**
  4. 
  5. Figure
  6. Label in Default.aspx
  7. In this task, you will add functionality to take the environment name from the Web.config file and populate the text attribute of the label.
  8. In the **HTMLLab** Web Application, open the **Web.config** file.
  9. Add the **<appSettings >** XML element with the following value:
     1. XML

1. <configuration>
2. ...
   * 1. **<appSettings>**
     2. **<add key="EnvironmentName" value="Development" />**

**</appSettings>**

1. ...

</configuration>

* 1. In the **Solution Explorer**, right-click Default.aspx and select **View Code** from the menu.
  2. Locate the **Page\_Load** method. Add the following code to get the value for key **EnvironmentName** and assign it to the **Text** property of the **EnvName** Label:
     1. C#
     2. protected void Page\_Load(object sender, EventArgs e)
     3. {
     4. Products.DataSource = Repository.GetProductsList();
     5. Products.DataBind();
     6. **EnvName.Text = ConfigurationManager.AppSettings["EnvironmentName"];**
     7. }
  3. Run the application and verify that the word **Development** appears as the text in the **EnvName** Label.
     1. 
     2. Figure 19

Staging configuration added

* 1. Close the web browser.

Task 3 – Adding a Logging Database Connection String to the Web.config File

* 1. In this task, you will add a connection string to logging application messages to the database.
  2. In the **HTMLLab** Web Application, open the Web.config file.
  3. Add the **connectionStrings** XML element with the following value added:
     1. XML

1. <configuration>
2. ...
   * 1. **<connectionStrings>**
     2. **<add name="LoggingConnectionString" connectionString="server=(local);database=Logger;Integrated Security=SSPI;" providerName="System.Data.SqlClient"/>**
     3. **</connectionStrings>**
3. ...

</configuration>

* 1. Save Web.config file changes.

Task 4 – Writing a Transform to Change the Environment and Logging Connection Strings in the Staging Web.config

In this task, you will create a transformation to update Web.config sections when the application is deployed with MSDeploy. In the Web.config file you will update the value of the Environment to “Staging” and change the server name value in the connection string to reflect QA server.

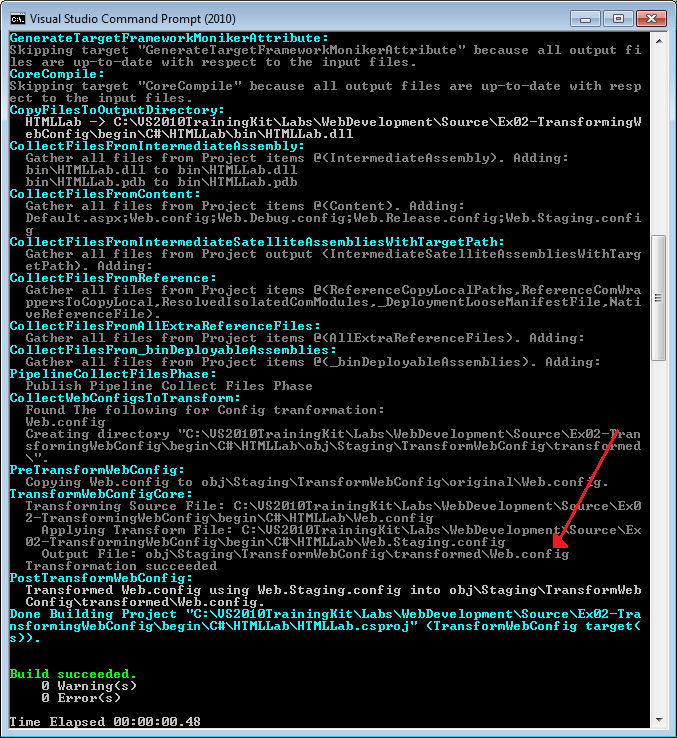
* 1. To display the configuration specific transformation files, expand the Web.config node and open the **Web.Staging.config** file.
  2. The transform file is an XML file that you use to specify the actions (add, delete, update) which Visual Studio should perform on nodes, sections, and attributes. The file contains a reference to the XML Document Transform namespace (http://schemas.microsoft.com/XML-Document-Transform). There are two attributes in this schema: **Transform** and **Locator**. **Transform** will make changes to nodes and attributes, while the **Locator** provides methods to find the specific transformation node and attributes in the Web.config.

**Note:** Check MSDN for the complete reference for the **Transform** and **Locator** attributes.

* 1. The Staging configuration transform file will already have a node indicating changes to the system.web section of the Web.config file, specifically removing the debug attribute.
     1. XML
     2. <?xml version="1.0"?>
     3. <configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transfrom">
     4. ...
     5. <system.web>
     6. <compilation xdt:Transform="RemoveAttributes(debug)" />
     7. </system.web>
     8. </configuration>
  2. Below the **system.web** node, add the following code.
     1. XML
     2. **<appSettings>**
     3. **<add key="EnvironmentName" value="Staging" xdt:Transform="SetAttributes(value)" xdt:Locator="Match(key)" />**
     4. **</appSettings>**
     5. When deployed, the transform above will instruct MSBuild process to locate the **EnvironmentName** node in the **appSettings** XML node and replace the value attribute with the one defined here: “Staging.”
  3. Transform **connectionString** XML in a similar matter by adding the following code:
     1. XML
     2. **<connectionStrings>**
     3. **<add name="LoggingConnectionString" connectionString="Server=QADatabaseServer;Database=Logging;Integrated Security=SSPI" providerName="System.Data.SqlClient” xdt:Transform="Replace" xdt:Locator="Match(name)" />**
     4. **</connectionStrings>**
     5. In this code, **connectionString** is transformed from using (local) server to the QADatabaseServer for logging.
     6. **Note:**
     7. xdt:Transform=”Replace” replaces the first matched node.

xdt:Transform=”SetAttribute(attributeName)” creates or changes values of the existing attributes.

Task 5 – Generating a Transformed Web.config File from the Command Line

* 1. In this task, you will use the MSBuild command line tool to create a build and transform the Web.config file based on the Staging configuration as part of the build process.
  2. Navigate to **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Visual Studio Tools**. To open the **Visual Studio Command Prompt**, click on **Visual Studio Command Prompt (2010)**.
  3. Change directory to the HTMLLab project directory. For example, if you are still using the begin solution for Exercise 1, type:
     1. CMD
     2. **cd %TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab**
  4. At the command prompt, invoke MSBuild by typing the following command and hitting **ENTER**:
     1. CMD
     2. **MSBuild *HTMLLab.csproj* /t:TransformWebConfig /p:Configuration=Staging**
     3. MSBuild builds the application and transforms the Web.config file according to the Staging transform rules. The output files placed in the HTMLLab\obj\Staging\TransformWebConfig\transformed folder.
     4. 
     5. Figure 20

MSBuild output

* 1. In Visual Studio, select the **File** | **Open** | **File** menu command and navigate to the project directory and then to obj\Staging\TransformWebConfig\transformed folder. Select the **Web.config** file and click **Open**.

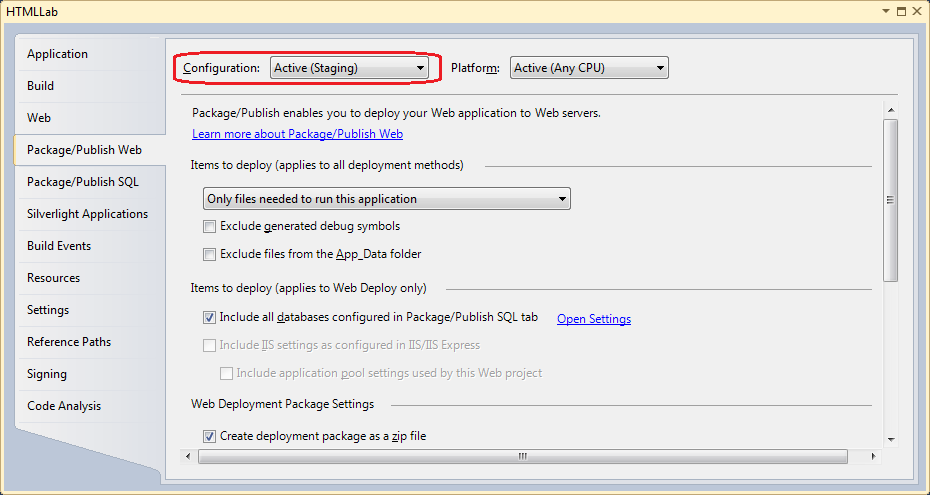
Notice that in the Web.config file the value for key **EnvironmentName** has changed to **Staging** and the connection string has been updated from **(local)** to **QADatabaseServer**.

* + 1. XML
    2. <appSettings>
    3. <add key="EnvironmentName" value="Staging" />

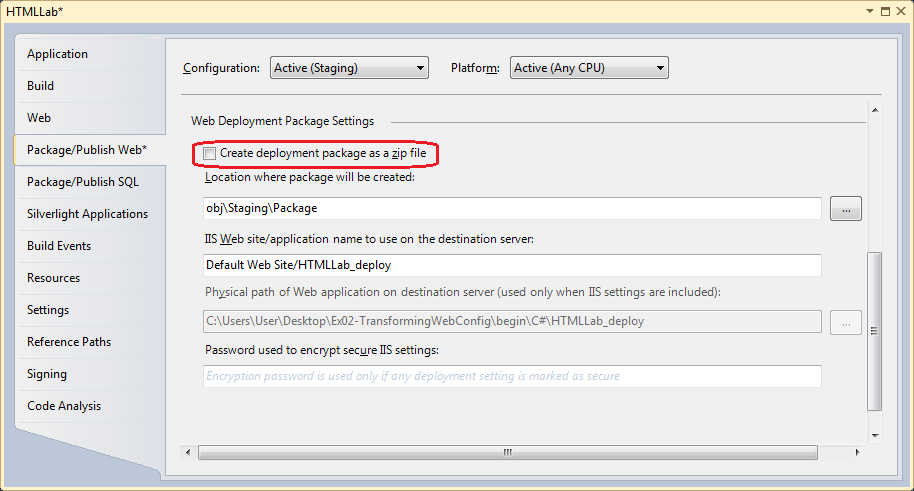
</appSettings>

* + 1. <connectionStrings>
    2. <add name="LoggingConnectionString" connectionString="Server=QADatabaseServer;Database=Logger;Integrated Security=SSPI;" providerName="System.Data.SqlClient"/>
    3. </connectionStrings>
  1. Close the Web.config file.

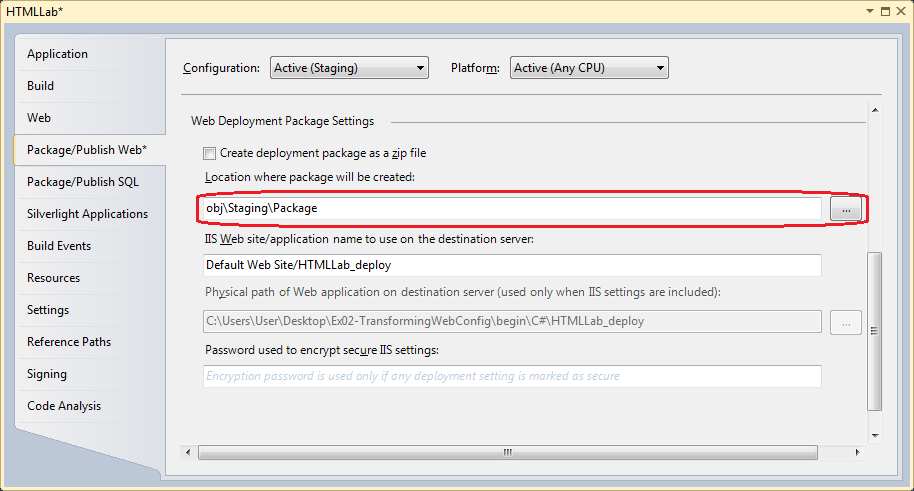
Task 6 – Generating a Transformed Web.config File from Visual Studio

* 1. In this task, you will use Visual Studio to create a build to transform the Web.config file as part of that build.
  2. In the Solution Explorer, right-click the **HTMLLab** project and select **Properties**.
  3. On the project properties page, click the **Package/Publish Web** tab.
  4. At the top of the properties page, make sure that the active configuration is set to **Staging**:
     1. 
     2. Figure 21

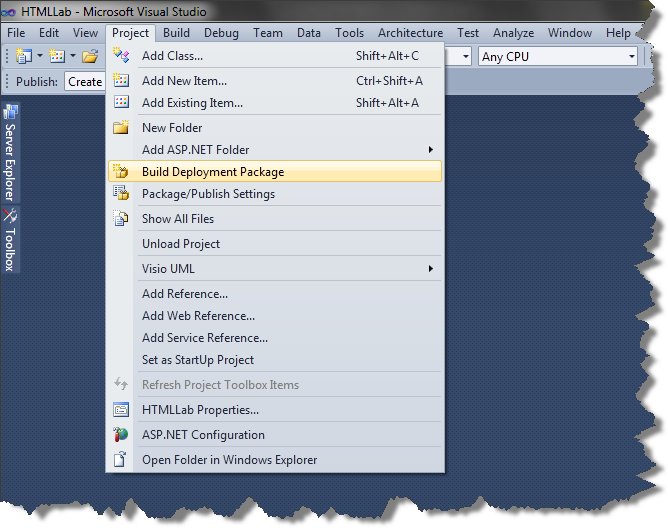
Verify that the Staging configuration selected.

* 1. At the bottom of the properties page, uncheck the **Create deployment package as a zip file** checkbox.
     1. 
     2. Figure 22

Uncheck the deploy as a ZIP file box

* + 1. **Note:** If **Create deployment package as a zip file** checkbox is left checked, the same build will occur but in addition, Visual Studio will create a ZIP file.
  1. Note the package location.
     1. 
     2. Figure 23

Package location

* + 1. This is the directory where Visual Studio will create the package that contains the deployable image for the Web application, including the modified Web.config file.
  1. Save all the changes by pressing CTRL+SHIFT+S.
  2. Using Windows Explorer, navigate to the HTMLLab\obj\Staging\TransformWebConfig\transformed folder. Delete the Web.config file located there, if exists.
  3. In Visual Studio, select the **Project** menu and click **Build Deployment Package**. Visual Studio will re-build and create a deployment package for the web application.
     1. 
     2. Figure 24

Build Deployment Package

# Next Step

Exercise 2: Verification

### Exercise 2: Verification

* 1. In order to verify that you have correctly performed all steps of the exercises, proceed as follows.
  2. In Visual Studio 2010 select **File** | **Open** | **File** from the menu to open **File** dialog. Navigate to the HTMLLab project directory and then to obj\Staging\TransformWebConfig\transformed folder. Select the Web.config file and click **Open**.
  3. Notice that in the Web.config file the value for **EnvironmentName** has been changed to **Staging**.
     1. XML
     2. <appSettings>
     3. <add key="EnvironmentName" value="Staging" />

</appSettings>

* 1. Locate connection string in the Web.config file and verify that the database server name has been updated to QADatabaseServer.
     1. XML
     2. <connectionStrings>
     3. <add name="LoggingConnectionString" connectionString="Server=QADatabaseServer;Database=Logger;Integrated Security=SSPI;" providerName="System.Data.SqlClient"/>
     4. </connectionStrings>
  2. Close the Web.config file.

# Next Step

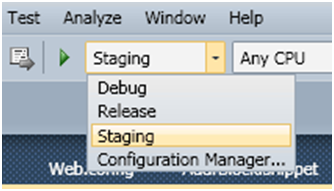
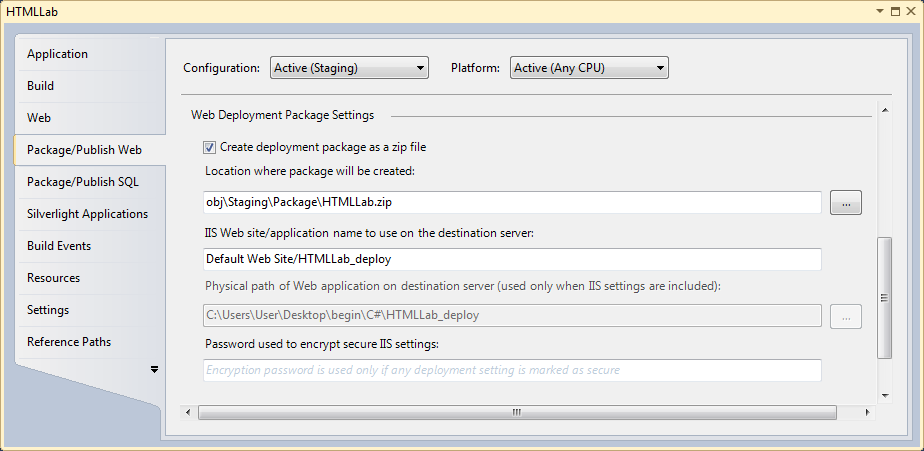
* 1. Exercise 3: Packaging and Deploying Web Applications for the Visual Studio Development Web Server

Exercise 3: Packaging and Deploying Web Applications for the Visual Studio Development Web Server

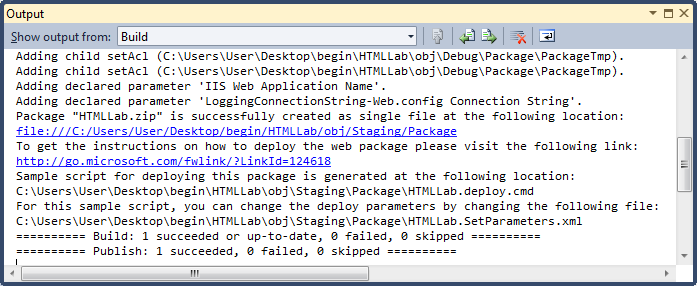
* 1. Most web sites need a lot of artifacts and resources to function properly. Deploying all of these correctly is not a trivial task. Visual Studio 2010 contains a new feature which allows you to create a ZIP file called **Web Package** which is a self-describing entity containing all the assets and resources that should be deployed to the server by MSDeploy.
  2. In this exercise, you will use this new feature to deploy the web application to the VS Development Web Server.

**Note:** To complete the tasks in this exercise, you must have completed all the tasks in Exercise 2, or use the solution from the ***%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex03-PackageDeployDevServer\begin\C#\HTMLLab*** folder.

Task 1 – Opening the Project Properties Publish Page and Creating a Package

* 1. In this task, you will open the product properties page and examine some of the features and settings available.
  2. Open project property pages by right-clicking the HTMLLab project and selecting **Properties** from the menu. Click the **Package/Publish Web** tab.
  3. Verify that the active build configuration is set to **Staging** from the build option menu.
     1. 
     2. Figure
     3. Verifying the Active Build Configuration
  4. Select the **Create deployment package as a zip file** option, if it is not already selected. Note the path in **Package Location**. This is where the package will be placed after it is built.
     1. 
     2. Figure 26

Create package as ZIP file and output path

* 1. Using Windows Explorer, navigate to the **HTMLLab** project directory and then to the **obj** folder and delete all the content of the folder, if the folder exists.
  2. Select the **Project** menu and click **Build Deployment Package**. In the **Output** window observe that the build and publish process have succeeded.
     1. 
     2. Figure 27

Successful build and publish

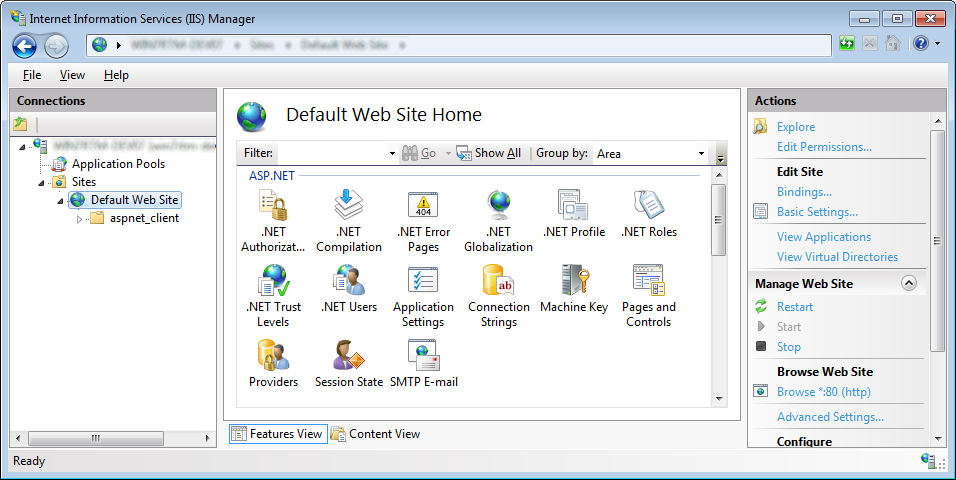
* 1. Using Windows Explorer, browse to the relative path specified in the package location textbox referenced above. Note a newly created folder for the selected configuration. Double-click the folder. Double-click the **Package** folder to open it.
     1. 
     2. Figure 28

Packaged web application

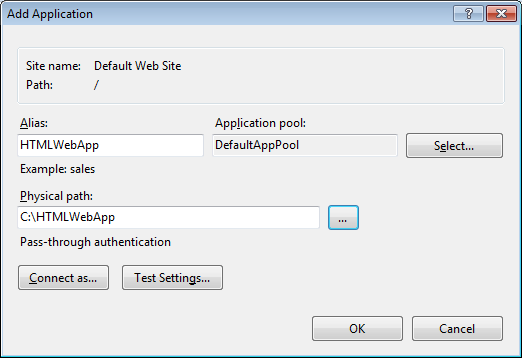
Note that this folder contains the following items:

* + - HTMLLab.deploy.cmd – It is a batch file generated by Visual Studio to help with the installation of the web application on the destination server.
    - HTMLLab.deploy-readme.txt – contains detailed information about how you can parameterize the deployment using Web Deploy (msdeploy.exe).
    - HTMLLab.SetParameters.xml –This file provides the installation path. By default, this file points to the path specified in the application name to use in the destination server textbox. If you choose to install the package elsewhere, you will need to change the path in this file.
    - HTMLLab.SourceManifest.xml –Visual Studio uses this file internally to create the package. The lab ignores this file.
    - HTMLLab.zip – The deployable web package.
    1. **Note:** In most situations, you will be deploying your application to a different machine than the one in which you built the package. In those cases, the three files that you will need to move to the target machine are the *ProjectName.deploy.cmd*, *ProjectName.SetParameters.xml*, and *ProjectName.zip* files.

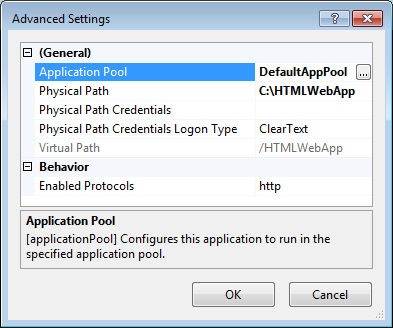
Task 2 – Preparing the Web Server for Deployment

* 1. In this task, you will create a new application on the web server to which this application is to be deployed.
  2. Using Windows Explorer, create a new folder “C:\HTMLWebApp”.
  3. Browse to **Start** | **All Programs** | **Accessories**, and click **Run**. In the **Run** box type: “inetmgr” and press **ENTER** to bring up **Internet Information Services (IIS) Manager**.
  4. In the **Connections** pane, located on the left, expand the node for the local machine. Within that node, expand the node for **Sites**. You should now see a list of web sites:
     1. 
     2. Figure 29

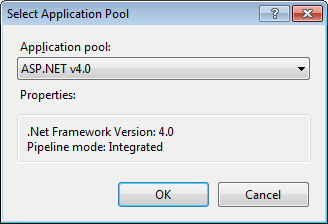
List of local web sites

* 1. Right-click the **Default Web** **Site** and select **Add Application**.
  2. In the **Add Application** dialog set the Alias to **HTMLWebApp**, and the **Physical path** to **C:\HTMLWebApp**. Take note of the **Physical path** as you will be deploying the package there. Click the **OK** button to close the dialog.
     1. 
     2. Figure 30

List of local web sites

* 1. In **IIS Manager**, right-click the new **HTMLWebApp** Web Application, point to **Manage Application** and select **Advanced Settings**. The **Advanced Settings** dialog will be displayed.
     1. 
     2. Figure 31

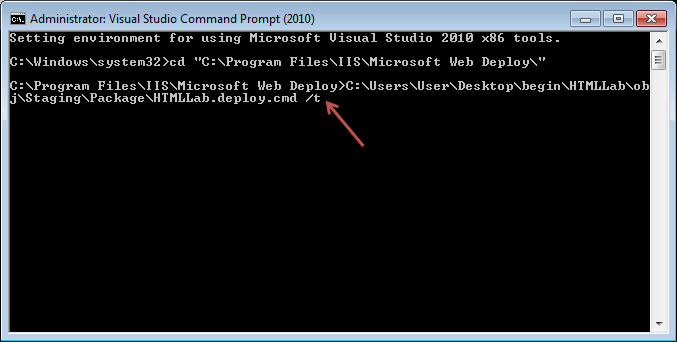
Web application Advanced Settings

* + 1. Note that the **Application Pool** is set to **DefaultAppPool**.
  1. Select **Application Pool**, and then click the ellipsis button next to the **DefaultAppPool** to bring up the **Select Application Pool** dialog.
     1. 
     2. Figure 32

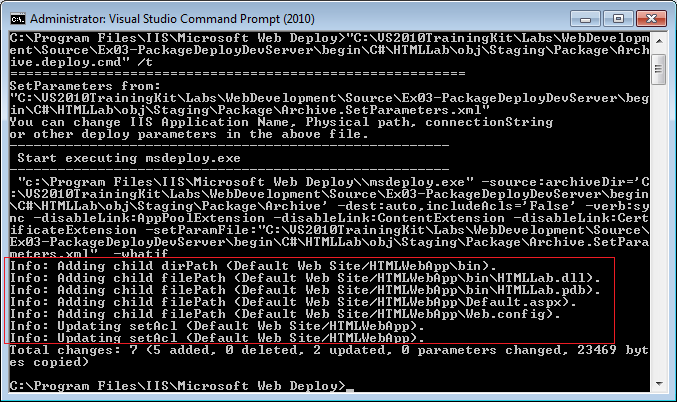
Select Application Pool

* 1. Using the dropdown list box labeled **Application pool** select **ASP.NET v4.0**. Click **OK** to close the **Select Application Pool** dialog. Click **OK** to close the **Advanced Settings** dialog.
     1. **Note:**
     2.  Watch Out
     3. If IIS was not enabled during the Visual Studio 2010 installation you may need to install ASP.NET into IIS by running the following command from a Visual Studio 2010 Command Prompt with administrator privileges: ***aspnet\_regiis.exe –iru***. After doing this, you will need to close and reopen IIS Manager to see “ASP.NET V4.0”.
     4. If you do not see a **Deploy** menu option it could mean that IIS Web Deployment Tool is not installed or may need to be reconfigured. To fix this you should do the following:
     5. – Download the IIS Web Deployment tool from <http://www.iis.net/extensions/WebDeploymentTool>
     6. – Start the setup
     7. If setup shows you the Change /Repair / Remove screen, select Change and install the IIS Manager UI module.

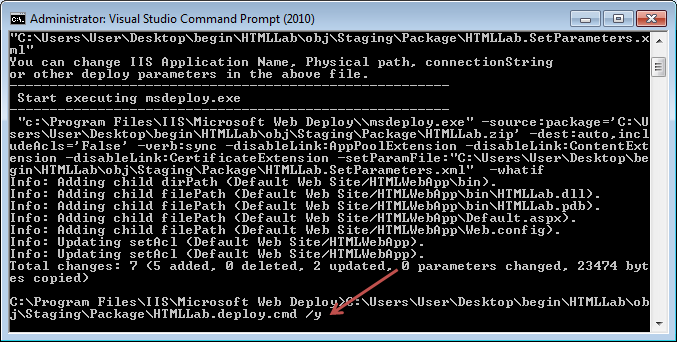
Task 3 – Deploying the Package

* 1. In this task, you will deploy the package to the web application created in the previous task.
  2. To deploy the package to the earlier specified physical location, you will need to change the **HTMLLab.SetParameters.xml** file. In Windows Explorer, select the **HTMLLab.SetParameters.xml** file and open it with Visual Studio. The current file content should appear similar to the following.
  3. XML
     1. <?xml version="1.0" encoding="utf-8"?>
     2. <parameters>
     3. <setParameter name="IIS Web Application Name" value="Default Web Site/HTMLLab\_deploy" />
     4. <setParameter name="LoggingConnectionString-Web.config Connection String" value="Server=(local);Database=Logging;Integrated Security=SSPI" />
     5. </parameters>
  4. Replace the **ISS Web Application Name** element with the following:
  5. XML
  6. <setParameter name="IIS Web Application Name" value="Default Web Site/HTMLWebApp" />
     1. **Note:** The HTMLLab.SetParameters.xml file should be located in the **obj\Staging\Package** folder of the HTMLLab project.
  7. Click **File** | **Save** to save the HTMLLab.SetParameters.xml file.
  8. Next, you will deploy the application in Trial mode.
     1. **Note: Trial** or **What If** mode does not actually perform the deployment, but shows you what will happen if you install the package. This is very useful in situations where you are handing off your package to a deployment team or server administrator. The team or administrator can then run the package in **What If** mode to see the impact on the server.
  9. Browse to **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Visual Studio Tools**. Right-click **Visual Studio Command Prompt (2010)** and select **Run as administrator** to open the command prompt.
  10. At the command prompt, change the current directory to **C:\Program Files\IIS\Microsoft Web Deploy\** by typing:
      1. CMD
      2. **cd "C:\Program Files\IIS\Microsoft Web Deploy\"**
  11. Execute the HTMLLab.deploy.cmd batch file with the **/t** (for Trial) flag. For example, if you are still using the HTMLLab project from Exercise 1, type:
      1. CMD
      2. **"%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab\obj\Staging\Package\HTMLLab.deploy.cmd" /t**
      3. 
      4. Figure 33

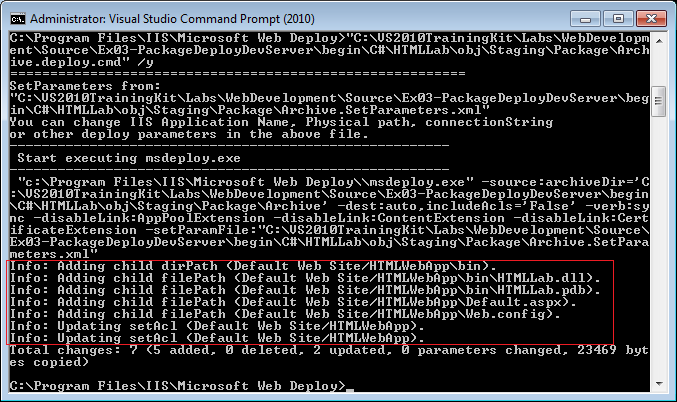
Executing the deployment command batch file in Trial mode

* 1. Examine the output of the Trial deployment.
     1. 
     2. Figure 34

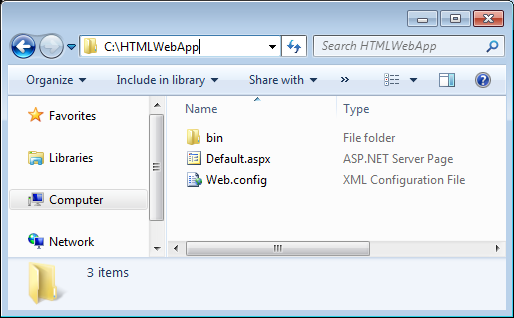
Results of the Trial deployment

* + 1. MSDeploy has added five additional files to the specified location.
  1. Execute the deployment by re-running this HTMLLab.deploy.cmd batch file. In this instance remove the **/t** (trail) flag and replace it with the **/y** (for Yes) flag by typing:
     1. CMD
     2. **"%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab\obj\Staging\Package\HTMLLab.deploy.cmd" /y**
     3. 
     4. Figure 35

Executing the deployment command batch file in Yes mode

* 1. Examine the output of the deployment.
     1. 
     2. Figure 36

Executing the deployment command batch file in Yes mode

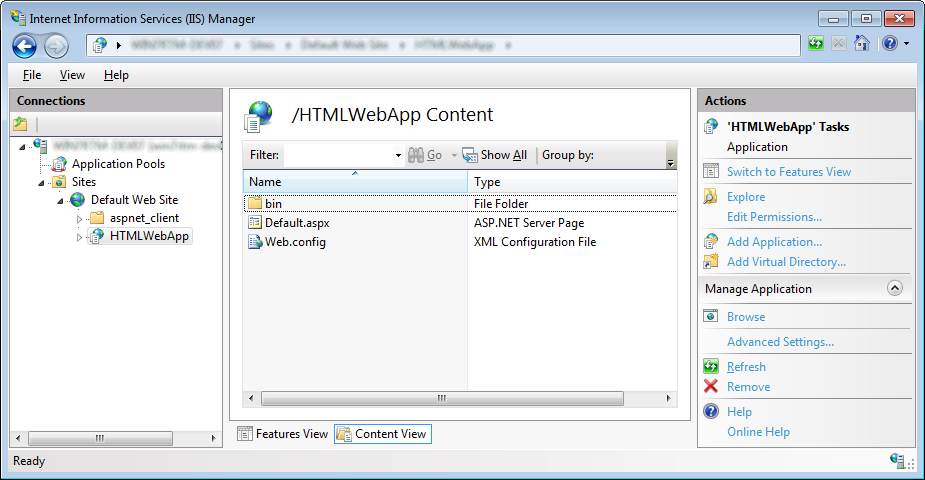
* + 1. MSBuild has deployed the package to the specified physical location.
    2. 
    3. Figure 37

Deployed web application package

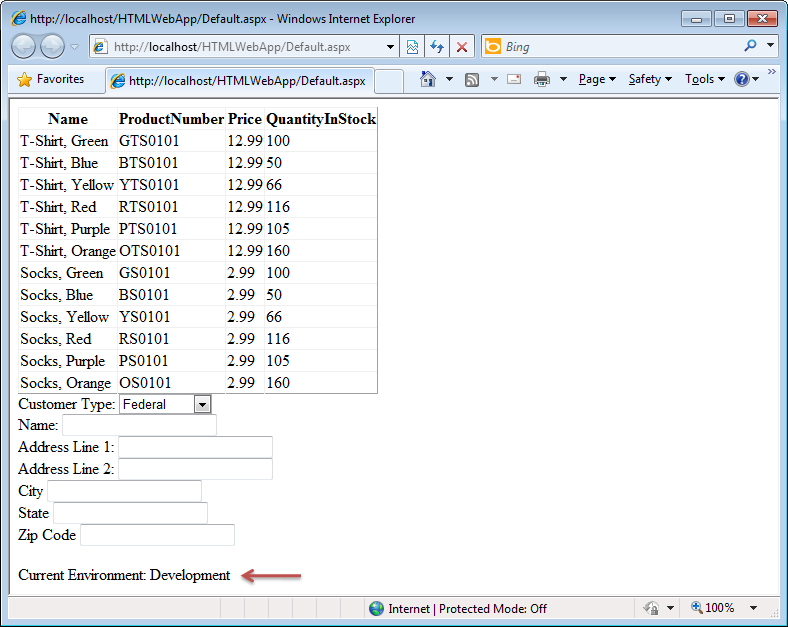
# Next Step

Exercise 3: Verification

### Exercise 3: Verification

* 1. In this verification, you will validate successful deployment of the web application.
  2. In **IIS Manager**, right-click on the HTMLWebApp Web Application and select **Switch to Content View** from the menu. The content view for the HTMLWebApp appears.
     1. 
     2. Figure 38

HTMLWebApp contents view

* 1. Right-click the Default.aspx file, and select **Browse** from the menu.
  2. Verify that the Default.aspx page for the HTMLWebApp loads.
     1. 
     2. Figure 39

Deployed web application

# Next Step

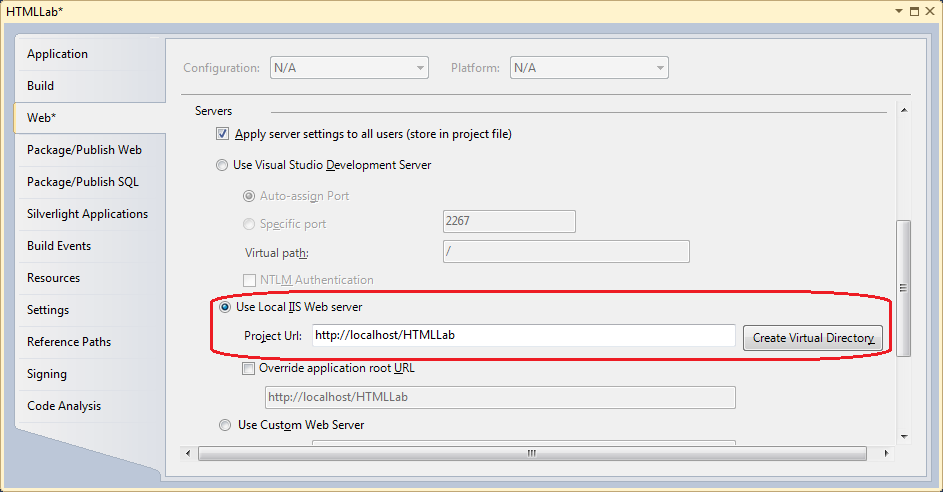
Exercise 4: Packaging and Deploying Web Applications for Internet Information Server (IIS)

Exercise 4: Packaging and Deploying Web Applications for Internet Information Server (IIS)

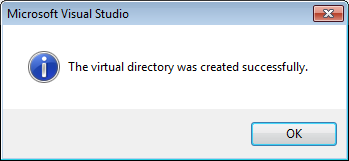
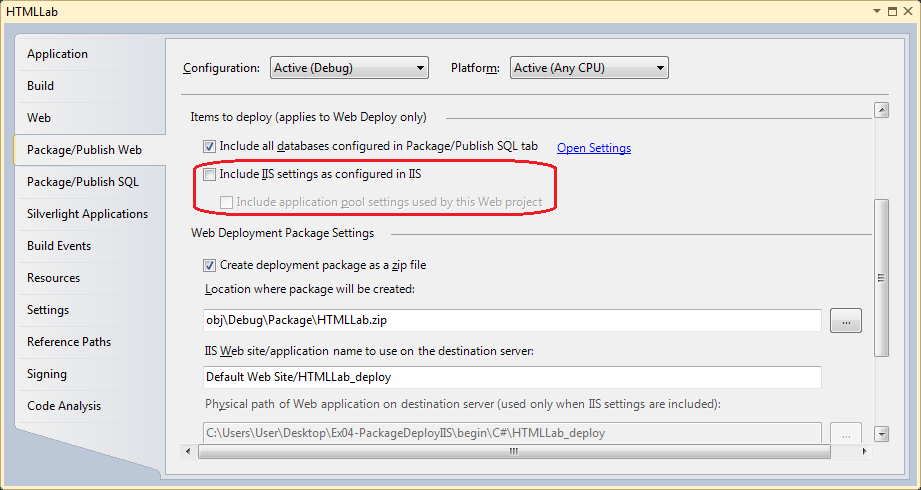
* 1. In the previous exercise, you deployed a web application to the Visual Studio Development Studio. In this exercise, you will deploy the same web application to IIS. **In order to complete this exercise, you will need to launch Visual Studio as Administrator, by right-clicking the Visual Studio 2010 icon in the Start menu and selecting *Run as administrator***.

1. **Note:** To complete the tasks in this exercise, you must have completed all the tasks in Exercise 3, or use the solution from the ***%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex04-PackageDeployIIS\begin\C#\HTMLLab*** folder.

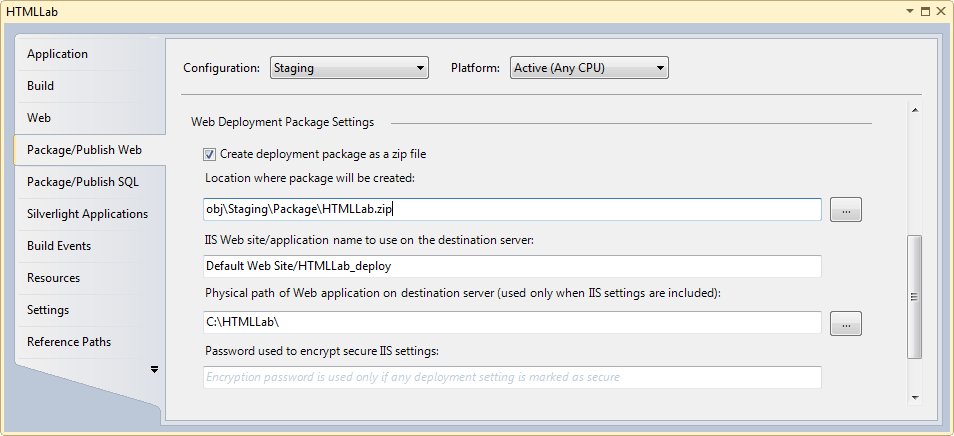
Task 1 – Opening the Project Properties Publish Page, Setting the Web Application to Use IIS and Create a Package.

* 1. In this task, you will open the product properties page, set the web application to use IIS and examine some of the features and settings available.
  2. Using Windows Explorer, create a new folder “C:\HTMLLab”.
  3. In Visual Studio, open the properties page for the HTMLLab project by right-clicking the HTMLLab project and selecting **Properties** from the menu.
  4. Click the **Web** tab.
  5. In the **Server** section of the properties page, click the radio button **Use Local IIS Web server**:
     1. 
     2. Figure 40

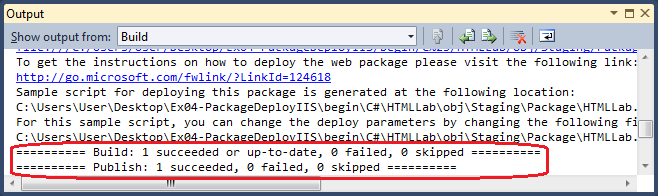
Setting web application to use local IIS Server

* 1. Click the **Create Virtual Directory** button to create a virtual directory on the local instance of IIS. You should see a message box notifying that the virtual directory was created successfully.
     1. 
     2. Figure
     3. Virtual directory created successfully
  2. Click the **Package/Publish Web** tab. In the **Items to deploy (applies to Web Deploy only)** section, notice that **Include IIS settings as configured in IIS** checkbox is now enabled.
     1. 
     2. Figure 42

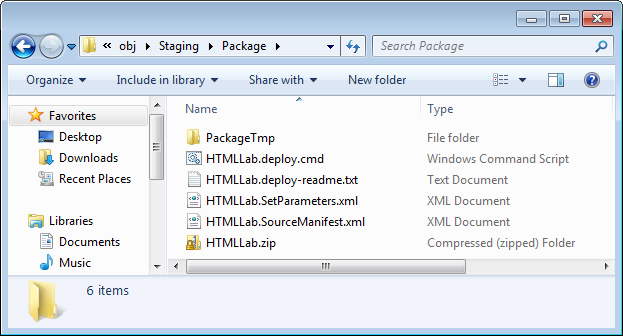
IIS settings is enabled

* + 1. Checking this box ensures that all of the IIS settings for your web application are packaged by MSBuild.
  1. Make sure that the active configuration is set to **Staging**.
  2. Check the **Include IIS settings as configured in IIS** checkbox and the **Include application pool settings used by this Web project** settings.
  3. Check the **Create deployment package as a zip file** checkbox. Note the path in the location textbox. This is where the build process will create the package.
  4. Set **C:\HTMLLab\** as Destination Application Physical Path
     1. 
     2. Figure 43

Create package as ZIP file and output path

* 1. Press **CTRL+SHIFT+S** to save all the changes.
  2. In the **Project** menu click **Build Deployment Package**. In the **Output** window observe that the build and publish process succeeded.
     1. 
     2. Figure 44

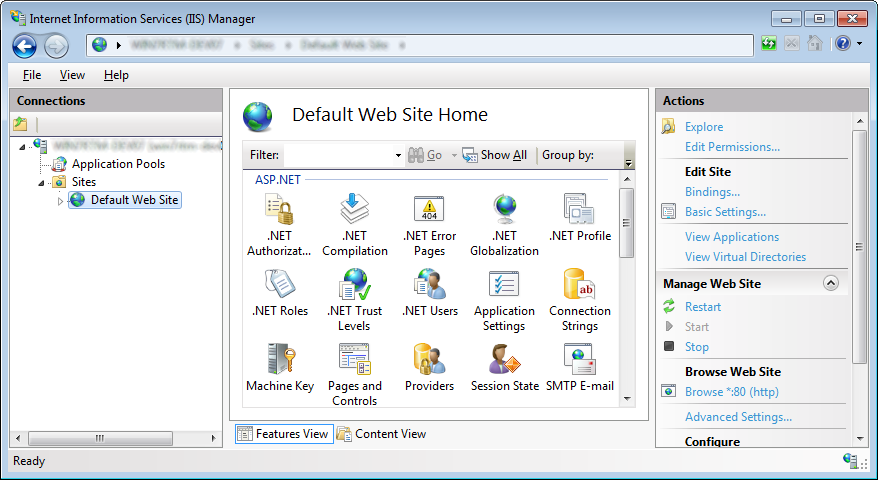
Successful build and publish

* 1. Using Windows Explorer, navigate to the relative path in the location textbox referenced above. For example, if you are still using the HTMLLab project from Exercise 1, navigate to the **%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab\obj** folder.
  2. Note that a folder for the selected configuration (Staging) has been created. Double-click on it. Within that folder, a new folder **Package** is created. Double-click on **Package** folder to open it.
     1. 
     2. Figure 45

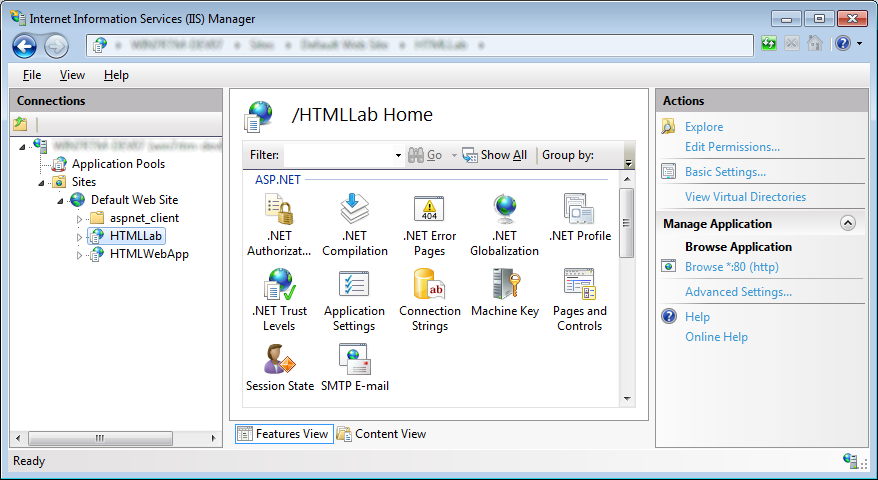
Packaged web application

* + 1. Note that this folder contains the same items as in the previous Exercise.

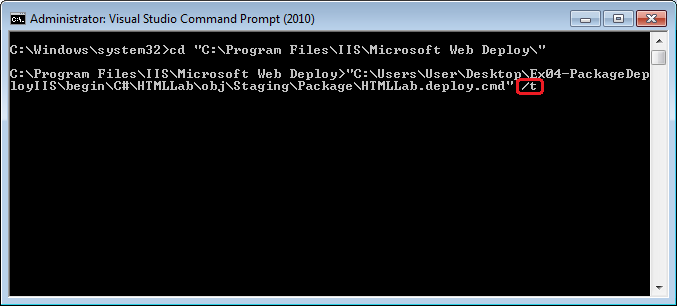
Task 2 –Deploying the Package

* 1. In this task, you will deploy the package to the web application created in the previous task.
  2. Navigate to **Start** | **All Programs** | **Accessories** and click **Run**. In the **Run** box type: “inetmgr” and click OK to bring up **Internet Information Services Manager**.
  3. In the **Connections** tree view on the left, expand the node for the local machine. Within that node, expand the node for sites.
     1. 
     2. Figure 46

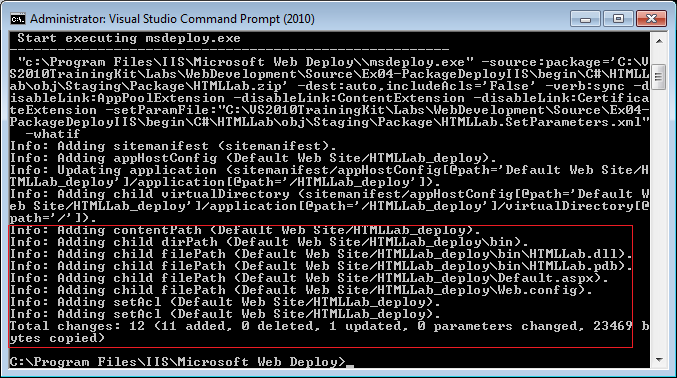
List of local web sites

* 1. Expand the **Default Web Site** node. Verify that the **HTMLLab** web application is listed (this was created when you clicked the **Create Virtual Directory** button in the previous task.)
     1. 
     2. Figure 47

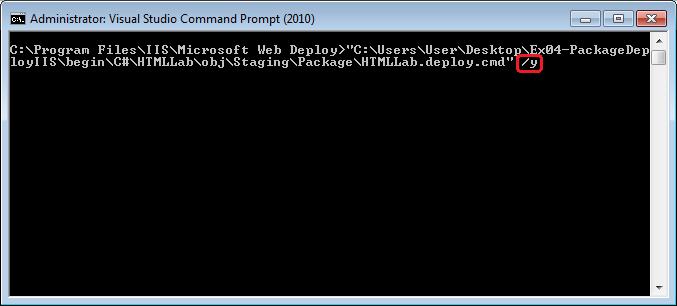
HTMLLab virtual directory created

* 1. Next you will deploy the application in Trial mode.
     1. **Note:** **Trial** or **What If** mode does not actually perform the deployment, but shows you what will happen if you install the package. This is very useful in situations where you are handing off your package to a deployment team or server administrator. The team or administrator can then run the package in Trial mode to see the impact on the server.
  2. Navigate to **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Visual Studio Tools**. Right-click **Visual Studio Command Prompt (2010)** and select **Run as administrator** to open the command prompt.
  3. At the command prompt, change the current directory to **C:\Program Files\IIS\Microsoft Web Deploy\** by typing:
     1. CMD
     2. **cd "C:\Program Files\IIS\Microsoft Web Deploy\"**
  4. Execute the HTMLLab.deploy.cmd batch file with the **/t** (for Trial) flag. For example, if you are still using the HTMLLab project from Exercise 1, type:
     1. CMD
     2. **"%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab\obj\Staging\Package\HTMLLab.deploy.cmd" /t**
     3. 
     4. Figure 48

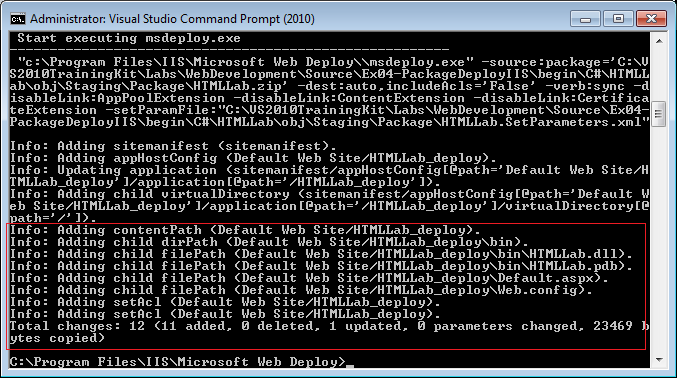
Executing the deployment command batch file in Trial mode

* + 1. **Note:** If you receive the following error after running the previous command:
    2. “*The application pool that you are trying to use has the 'managedRuntimeVersion' property set to 'v2.0'. This application requires 'v4.0'*”
    3. Execute the **aspnet\_regiis.exe –i** command from a Visual Studio 2010 Command Prompt with administrator privileges and then try again.
  1. Examine the output of the Trial deployment.
     1. 
     2. Figure 49

Results of the Trial mode deployment

* + 1. In this case, you can see that the MSDeploy will make 12 changes to the server.
  1. Execute the deployment by re-running the HTMLLab.deploy.cmd batch file. In this instance remove the **/t** (trial) flag and replace it with the **/y** (yes) flag. For example, if you are still using the HTMLLab project from Exercise1, type:
     1. CMD
     2. **"%TrainingKitInstallFolder%\Labs\WebDevelopment\Source\Ex01-HTMLCodeSnippets\begin\C#\HTMLLab\obj\Staging\Package\HTMLLab.deploy.cmd" /y**
     3. 
     4. Figure 50

Executing the deployment command batch file in Yes mode

* 1. Examine the output of the deployment.
     1. 
     2. Figure 51

Executing the deployment command batch file in Yes mode

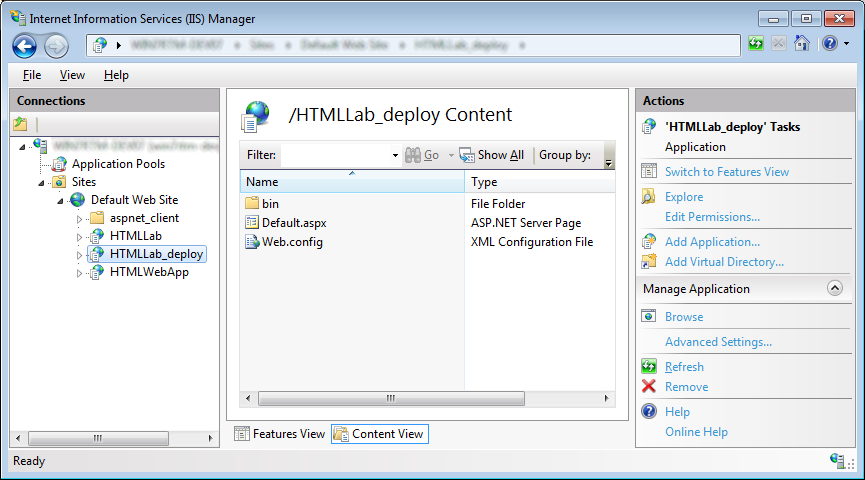
* + 1. MSBuild deployed the package to the specified physical location.

# Next Step

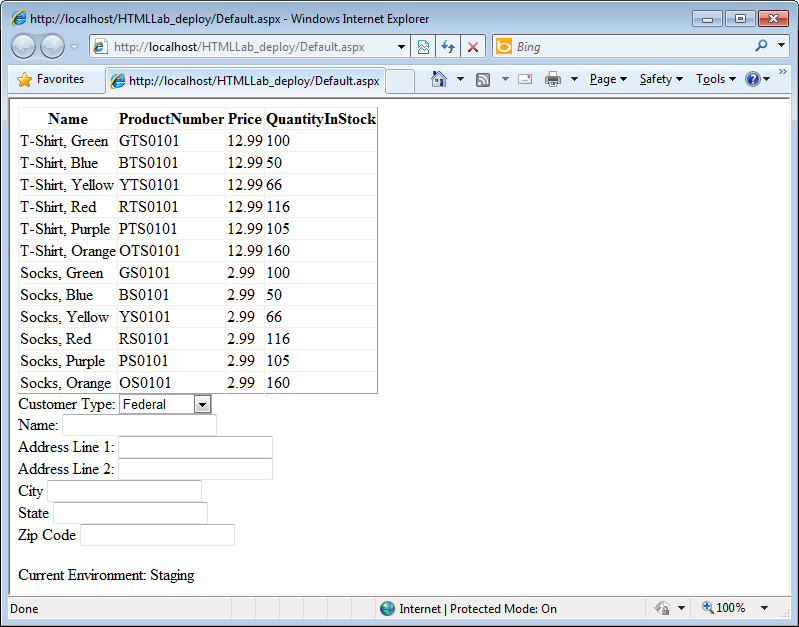
Exercise 4: Verification

### Exercise 4: Verification

In this verification, you will verify the successful deployment of the web.

* 1. In **IIS Manager**, right-click the **Default Web Site** and click **Refresh**.
  2. You should now see an **HTMLLab\_deploy** web application under Default Web Site.
  3. Right-click on the **HTMLLab\_deploy** application and select **Switch to Content View** from the menu. The content view for the HTMLLab\_deploy application should now be visible:
     1. 
     2. Figure 52

HTMLLab contents view

* 1. Right-click the Default.aspx file and select **Browse** from the menu.
  2. Verify that the default page for the HTMLLab\_deploy application appears.
     1. 
     2. Figure 53

Deployed web application

# Next Step

Summary

Summary

1. In this lab, you learned to use an HTML snippet to accelerate web development and created your own custom snippet. You also created an XML transform file that will make the appropriate changes to the Web.config file of your web application as it is promoted through various environments. Finally, you learned about the new packaging and publishing capabilites for ASP.NET web applications.