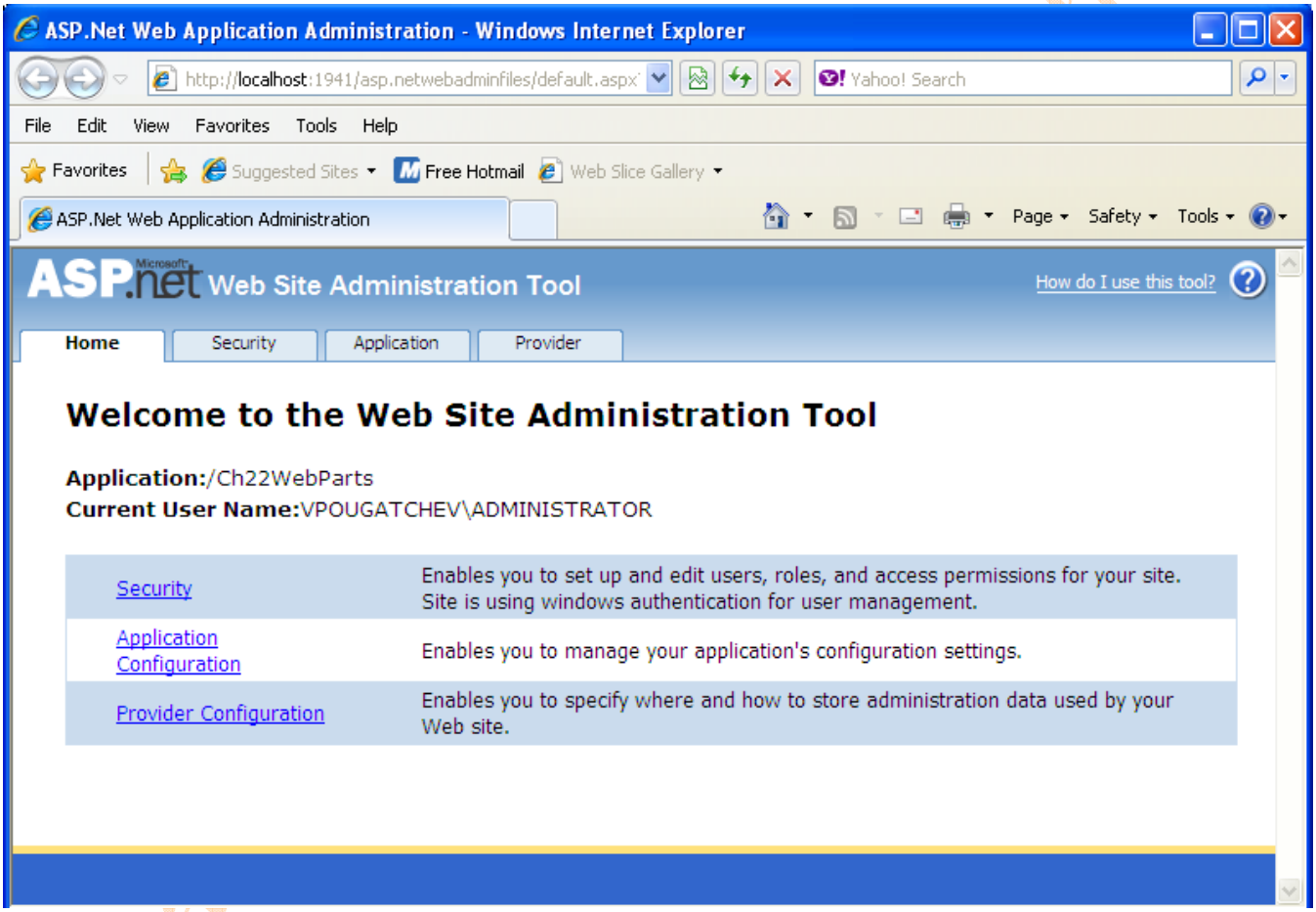
Lecture 8

Configuring and deployment of a web application

Using a Web Site Administration Tool

The figure below, shows the web Site Administration Tool is used a web-based editor that lets you specify certain configuration options. This tool uses a tabbed interface that lets you switch between the home page and the pages that configure security, application, and provider settings.



The four tabs of the Web Site Administration Tool (WSAT)

- **Home:** Displays the home page
- **Security:** Lets you configure security features.
- **Application:** Lets you create custom application settings, configure SMTP email support, control debugging and tracing settings, and start or stop the application.
- **Provider:** Lets you configure providers for features such as membership and profiles.

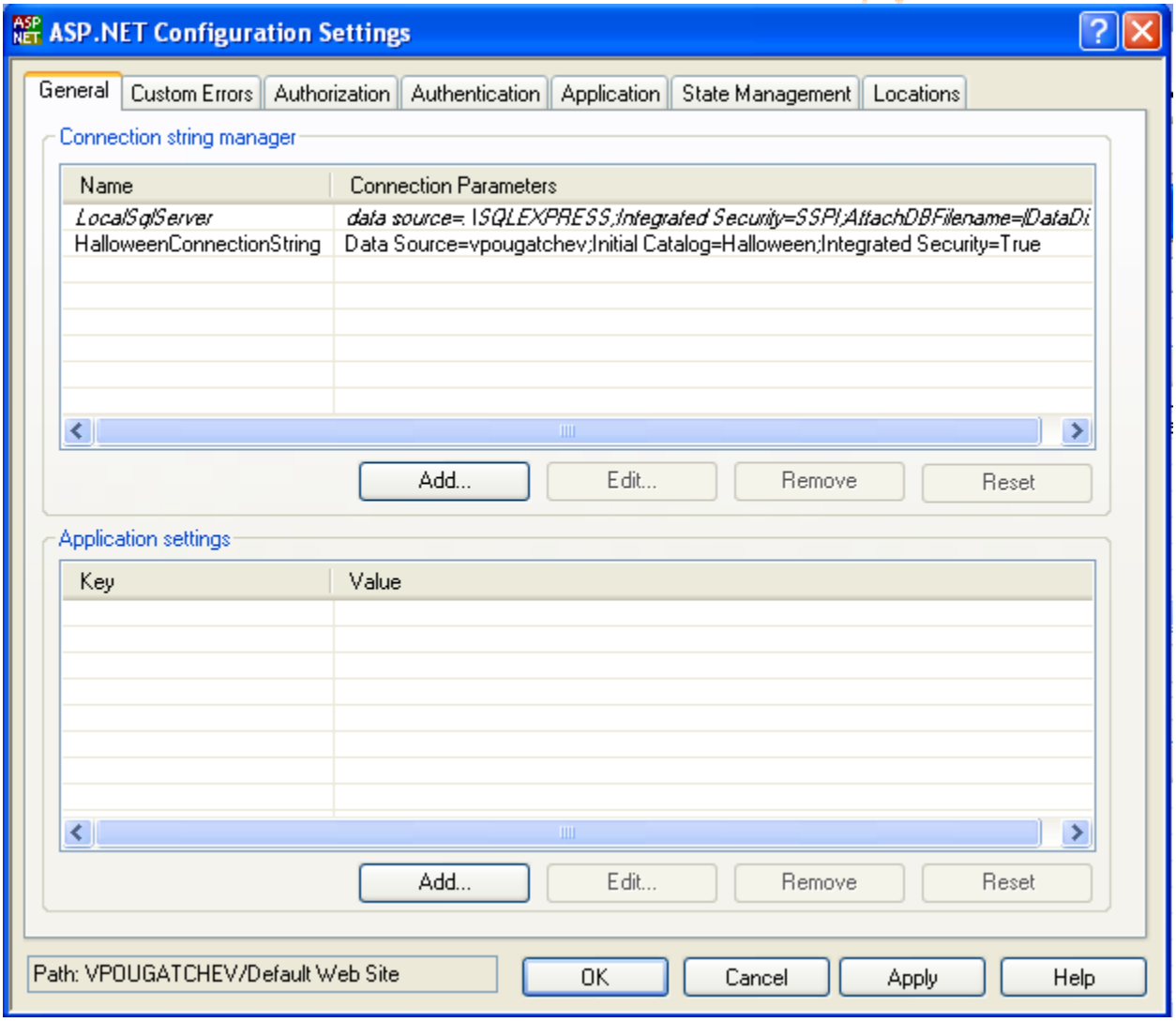
When you use the WSAT, you need to remember that it has two limitations:

1. It doesn't let you set all of the configuration options that are available via the *web.config* file. For example, you cannot use it to change the connection strings stored in the file or specify custom error pages.
2. You can only use the WSAT from within Visual studio 2005. As a result, you can't use this tool for a web site that's been deployed to a production server unless you can open the web site in Visual Studio 2005.

Using the IIS Management Console

To open the Internet Information Services management console you can use two ways. The first one:

1. On the ***Start*** menu, click ***Run***.
2. In the ***Open*** edit box, type ***inetmgr***.
3. Click ***OK***.
   The IIS management console opens:



The second one:

1. Press *Start* button, then select *Control panel*
2. In *Control panel* use an *Administrative tools*, and then use an Internet *Administration Services*.

Then right click the web site, and chose *Properties*. Then click the ASP.NET tab and click the *Edit Configuration* button.

With this dialog box, you can configure most of the settings that are specified via the *web.config* file. However, to use this dialog box, you must have the authorization to run the IIS Management Console on the server that hosts the application.

The seven tabs of the ASP.NET Configuration Settings dialog box

- **General:** Creates connection string and application settings
- **Custom Errors:** Configures custom error pages
- **Authorization:** Creates authorization rules
- **Authentication:** Specifies the authenticatuon mode and configures membership and role providers.
- **Application:** Configures application settings such as the default master page and theme.
- **State Management:** Configures session state settings
- **Location:** Adds Location elements to the *web.config* file that let you apply configuration settings to specific parts of the application.

Deployment of the web application

Deployment refers to the process of coping an ASP.NET web application from the development system to the production server on which the application will be run.

Three ways to deploy a web application

XCopy deployment

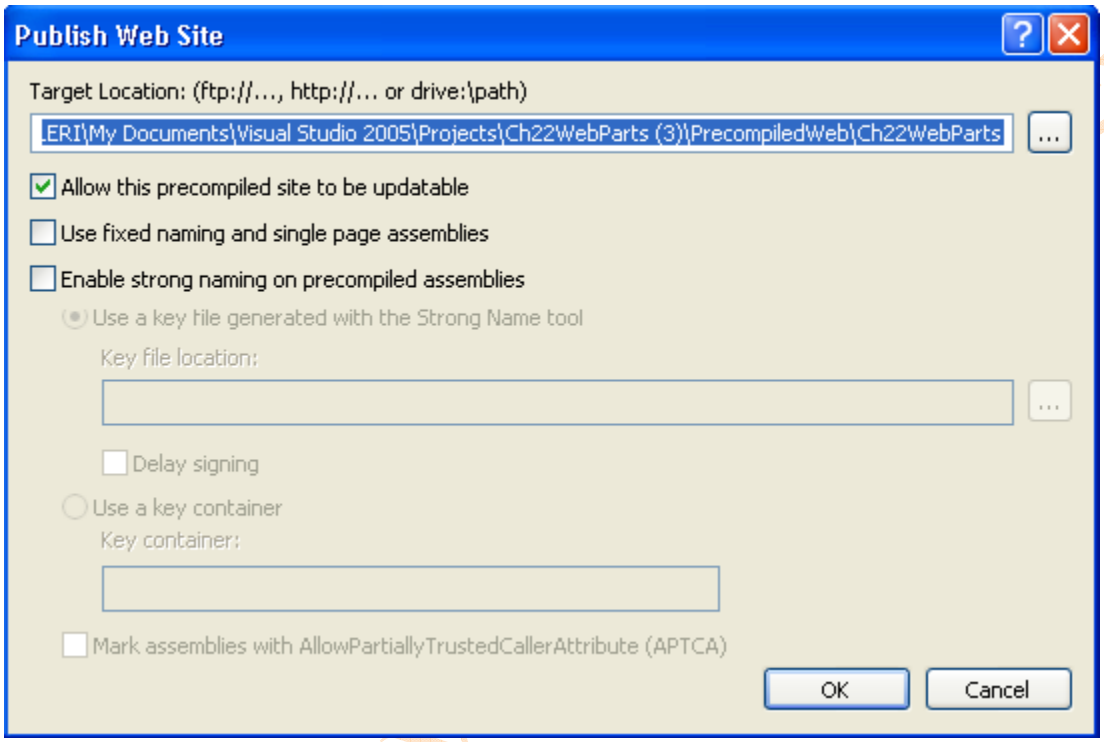
This approach is commonly called *XCopy* deployment because it simply copes the files required by the application to the production server.

- To manually copy the files of an ASP.NET web site to a server, you can use the *XCopy* command from a command prompt. Then, you can use the IIS Management Console to create a virtual directory that's mapped to the directory that you copied the web site to.
- To automate the deployment, you can create a batch file for the *XCopy* command. Then, you can run the batch file any time you make changes to the application and want to deploy the updated code. The help information can be called by *"xcopy /?"* command
- You can also do *XCopy* deployment from Visual Studio 2005 by using the *Copy Web Site* command from the *Website* option main Visual Studio menu.

Precompiled deployment

This is a new type of deployment that lets you precompile a web application and copy the precompiled assemblies to a target server. To use this method of deployment, you can use the *Publish Web Site* command from within Visual Studio 2005 as shown below. It will allow you:

- Deploys precompiled assemblies to the specified server
- Lets you deploy the web site with or without the source files.
- Can be done from within Visual Studio using the *Publish Web Site* command (the *Build* option of the main Visual Studio menu) or from a command prompt using the *aspnet_compiler* command.



Setup program deployment

The third way to deploy a web application is to develop a *Web Setup* project that creates a Windows *Setup* program for the application. Then you can run this *Setup* program on the production server to install the application. Generally this way allows you:

- To use a *Web Setup* project to build a Windows *Setup* program that can be run to deploy a web application to a server
- It is useful if you want to distribute a web application to multiple servers.
- Can be used to deploy precompiled assemblies and can be configured to include or omit the source files.
- An application that's installed by a Setup program can be removed by using the *Add* or *Remove* Programs dialog box that can be accessed from the Control Panel.

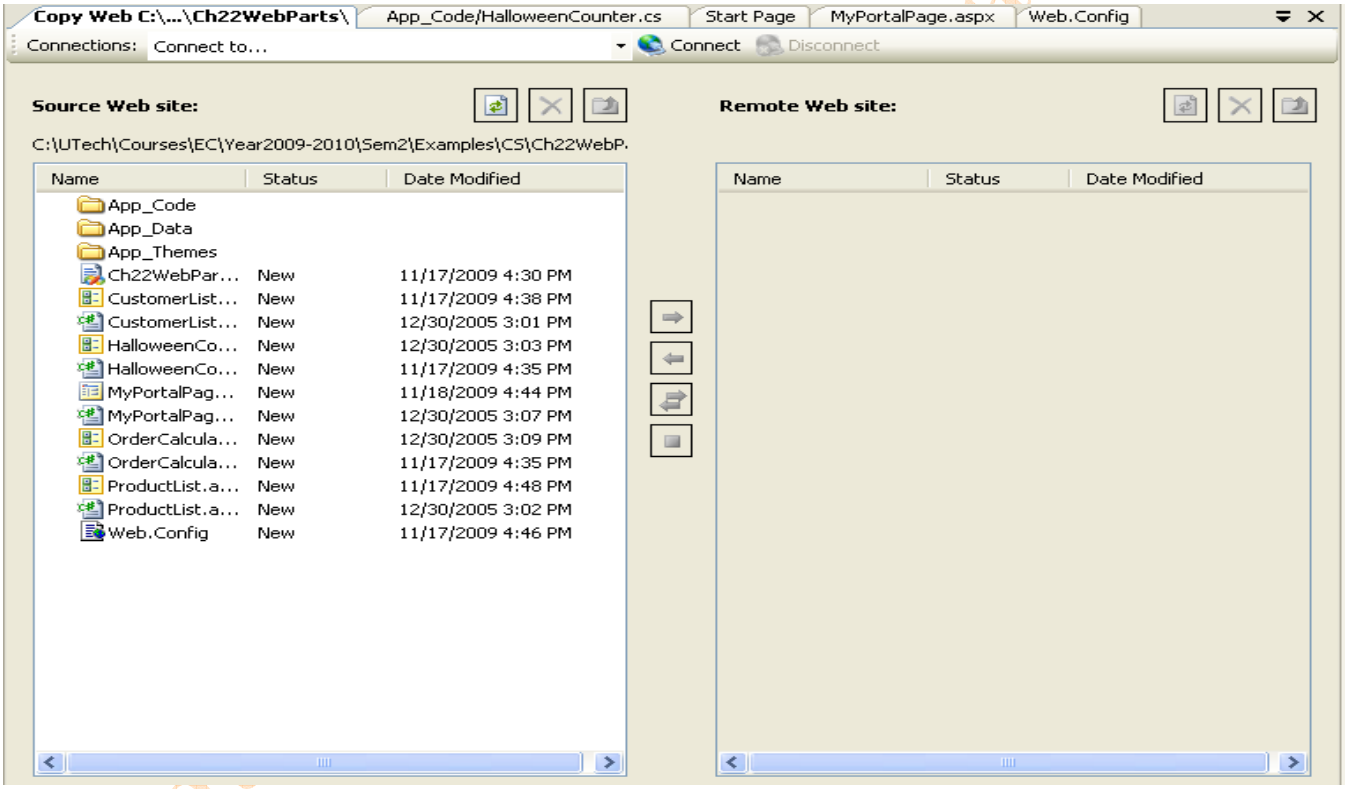
Which of these deployment alternatives is the best choice depends on the particular needs of each application. *XCopy* deployment is the easiest, and is often used during development to create copies of an application on different servers for testing purposes. For small applications, *XCopy* deployment may also be the best choice for production deployment.

Precompiled deployment has several advantages over *XCopy* deployment. For example, precompiled deployment provides better performance for the first users that access the site. In addition, it provides increased security because you don't have to copy the application's source files to the server.

For applications that are deployed to one or just a few servers, precompiled deployment is usually the best choice. However, if you're distributing an application to many different servers, you should consider creating a Setup program for the application.

Using a Copy Web Site command *XCopy* deployment

A figure below shows a powerful *Copy Web Site* command. This command lets you copy a web site to a file system, local IIS, FTP, or remote IIS web site. In addition, it lets you copy all of the files in the web site or just selected files, and you can use it to synchronize web sites so that both sites have the most recently updated versions of each file.

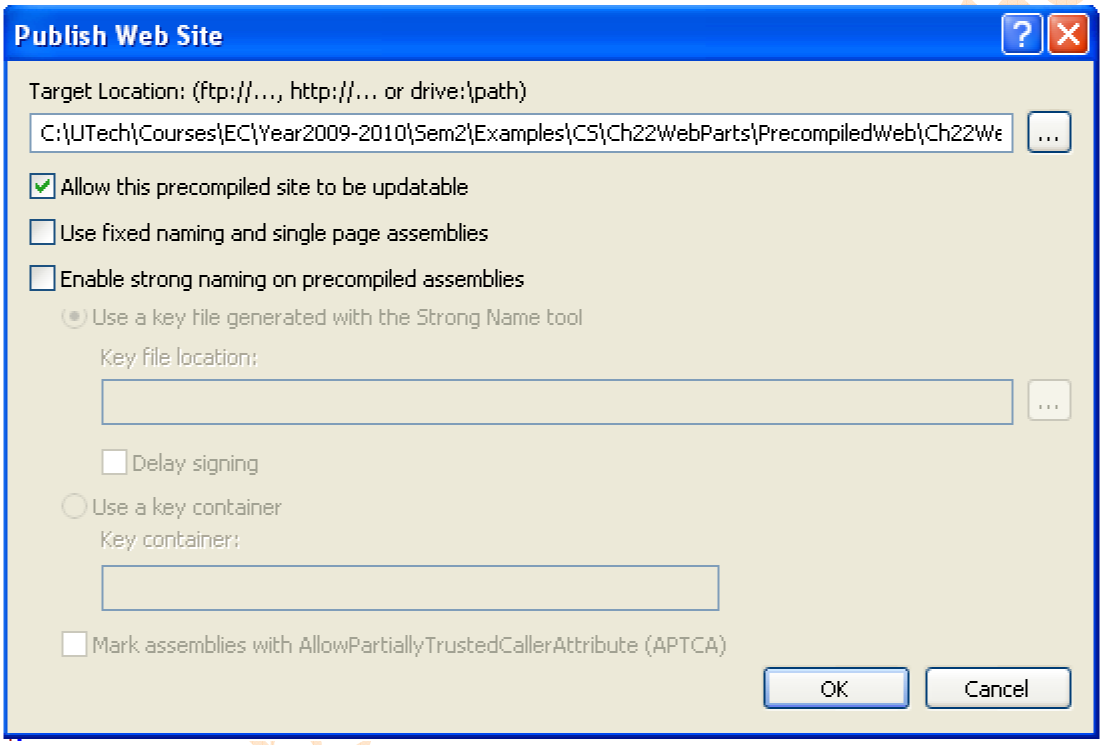


How to use the Copy Web Site command

1. In Visual Studio, open the web site you want to deploy and choose the *Website* → *Copy Web Site* command.
2. Click the *Connect* button to display an *Open Web Site* dialog box that lets you choose the destination you want to copy the web site to.
3. Select the files you want to copy. (Press *Ctrl+A* to select all of the site's files)
4. Click the → button to copy the files from the source web site to the remote web site.

Using a Publish Web Site command for precompiled deployment

The *Publish Web Site* command lets you recompile a web application and copy the precompiled assemblies to a target server. To get a dialog box, presented below you need to make a choice *The Build → Publish web site* command from the main Visual Studio menu.



If you check the *Allow this precompiled Site to be updatable* box, the source files are deployed to the server along with the executable files. If you leave this box unchecked, the source files aren't copied to the server.

This approach has three advantages:

1. Because all of the pages are compiled before the web site is deployed to the server, this avoids the delay that can be encountered by the first visitors to the web site. In contrast, when precompiled deployment isn't used, each page is compiled by the ASP.NET runtime the first time the page is accessing by a user. As a result, the first user to retrieve each page will encounter a delay while the page is compiled.
2. Any compiler error will be found before the application is displayed. Although unlikely, it's possible for pages in a production web site to become out of sync when the application is deployed. For example, a supporting class might be accidentally omitted. Then, when a page that uses that class is first accessed by a user, a compiler error will occur. By precompiling

the entire application, though, you can eliminate the chance of users encountering these errors.

3. You can deploy a precompiled application without the source code. This can be useful for two reasons. First, it avoids the security risk that's inherent when you place your application's source code on the production server because there's always the possibility that a hacker might exploit vulnerability in IIS and access your code. Second, if you develop a commercial application and don't want your customers to be able to access your source code, this makes that possible.

To omit the source code from a precompiled application, you uncheck the *Allow this precompiled site to be updatable* box. Then, the source files won't be copied to the production server. Instead, dummy files with the same names as the source files will be copied to the server. If you open of these files, you'll find that it contains a single line with the following text:

This is a marker file generated by the precompilation tool, and should not be deleted

Although these dummy files are required for the application to work, their contents are ignored by the ASP.NET runtime.

Using the *aspnet_compiler* command for precompiled deployment

When you use the *aspnet_compiler* command, the precompiled assemblies are copied to the target directory.

The syntax of the *aspnet_compiler* command

aspnet_compiler –v virtual-directory [-u] [-d] [-f] [target-directory]

Switches

Switch	Description
-v	Precedes the name of the virtual directory of the existing web site to be precompiled
-u	The precompiled web site will be updatable
-d	Debug information will be included in the compiled assemblies.
-f	Overwrites the target directory if it already exists.

Examples

Precompiles an existing web site

aspnet_compiler –v ProductList c:\Deploy\ProductList

Precompiles a web site in place

aspnet_compiler –v ProductList

Creates an updatable precompiled web site with debugging info

aspnet_compiler –v ProductList –u –d c:\Deploy\ProductList

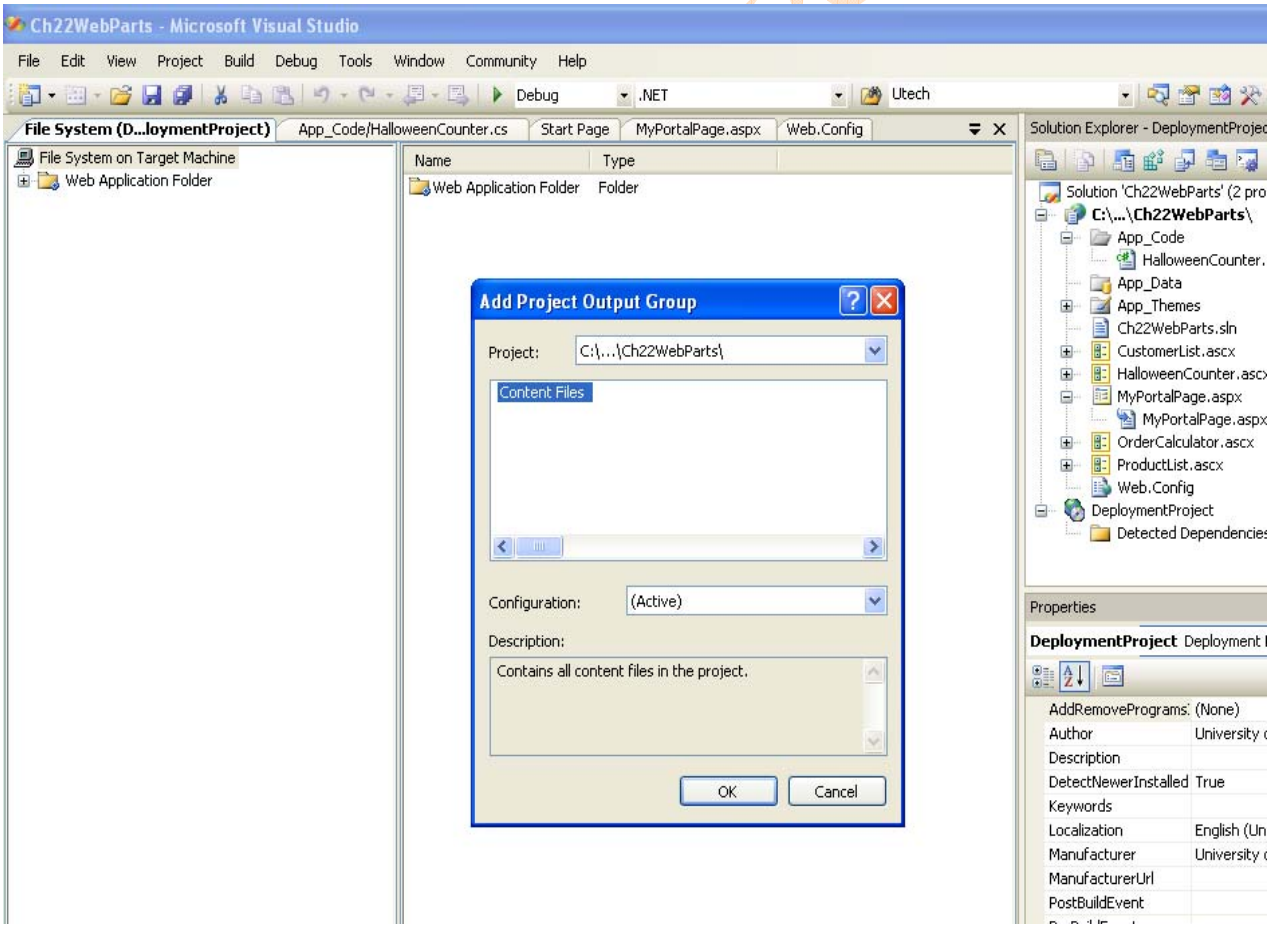
The aspnet_compiler command is located in the .NET Framework directory, which is

%systemroot%\Microsoft.NET\Framework\v2.0.xxxxx,

where *xxxxx* is the five-digit builds number. This location may vary depending on the release of ASP.NET 2.0 that you're using.

Creating a Web Setup project

Another way to deploy a web application is to develop a Web Setup project that creates a standard Windows Setup program that you can use to install the web application on an IIS server. In figure below, you can start by adding a Web Setup project to an existing web site. Then, you configure the Web Site project so it installs the files required by the web application. You can also use the setup editors to configure many custom options that control how the application will be installed.



Creating a Setup project:

- Choose the *File → Add → New Project* command to display the *Add New Project* dialog box. Then, choose *Setup and Deployment* in the *Other Project Types* list, select *Web Setup Project* as the template, enter a name for the *Web Setup* project, and click *Ok*.
- In the Solution Explorer, right-click the *Web Setup project* and choose *the Add → Project Output* command to display the *Add Project Output Group* dialog box. Then, click *Ok* to add the content files from your web site to the *Web Setup project.*
- Use the buttons that are displayed at the top of the Solution Explorer when the *Web Setup project* is selected to access setup editors that let you customize various aspects of the *Web Setup project.*

You can use the File System Editor to add files and folders to the setup project. To add a *ReadMe* file, for example, you can right-click the *Web Application Folder* in the left pane of the File System Editor and then select *Add → File*. Note, that the *Web Application Folder* represents the virtual directory where the web application will be installed, and it has properties that define that virtual directory. The two properties you're most likely to change are *VirtualDirectory*, which specifies the name of the virtual directory, and *DefaultDocument*, which specifies the page that's displayed by default.

You can use the Registry Editor to specify the keys and values that should be added to the registry on the target computer. And you can use the File Types Editor to establish a file association between a file extension and an application on the target computer. Since most web applications don't need to adjust registry settings or create file type associations, you shouldn't need either of these editors. However, other three editors can be useful.

The user Interface Editor lets you customize the dialog that is displayed when you run the Setup program to install a web application. For example, you can change the *BannerBitMap* property of any dialog box to display your company's logo in that dialog box. This editor also lets you add your own dialog boxes to those that are displayed by the Setup program.

The Custom Action Editor lets you add actions to those that are performed by the Setup program. For example, you can use a custom action to create a SQL Server database when the application is installed. Before you can add a custom action to a Setup project, though, you must first develop a batch file, a script, or an executable program that implements the action.

Finally, the Launch Conditions Editor lets you set conditions that must be met before the application can be installed. For example, you can use a launch condition to check the operation system version or to see if a particular file exists on the target system. If the condition isn't met, the Setup program aborts the installation.

In addition to using the setup editors, you can set properties of the Setup project. Some of those properties provide support information and are displayed in the *Support Info* dialog box. The user can display this information by opening *Add or Remove Programs* in the *Control Panel,* selecting the application, and clicking the *Click Here for Support Information* link for the application. Other properties provide additional information about the application, such as the icon that's displayed for the application in the *Add or Remove Programs* dialog box and the product name that's displayed by the Setup program and in the *Add or Remove Programs* dialog box.

Creating and using a Setup program

Once you've configured the Web Setup project, you can create the Setup program and install the application. To start, you should choose whether you want to create a debug or release version of the application. Because the debug version contains symbolic debugging information that you don't

typically need in production application, and because this version isn't optimized, you'll usually create a release version. On the other hand, you may want to create a debug version during testing. To determine which version is created, you can use the Configuration Manager.

Next, you build the Setup project, following steps, described below. This creates the *setup* files (*Setup.exe* and *Setup.msi*) in the *Debug* or *Release* folder of the project, depending on whether a debug or release version is created. Then, in most cases, you'll copy these files to a network server or burn them to a CD or DVD. Finally, you can install the application by running the *Setup.exe* program from the server that will host the application.

Note that the *Setup* program itself is a standard Windows application, not a web application. As a result, the *Setup.exe* and *Setup.msi* files aren't found in the same directory as the web application. Instead, you'll find these files in the project directory for the Web Setup project, which you can find under *My Documents\Visual Studio\Projects* by default.

When you run a Setup program to install the web application on the host server, it steps you through the installation process.

To create a Setup program you need:

- Right-click the *Setup* project in the Solution Explorer and use the *Build* command. When you do, Visual Studio creates files named *Setup.exe* and *Setup.msi. Setup.exe* is the file you run to install the application, and *Setup.msi* contains all of the files to be installed
- The *Setup.exe* and *Setup.msi* files are stored in the Web Setup project's *Debug* or *Release* folder, depending on whether Visual Studio is configured to create a debug or release version of the *Setup* project. In most cases, you'll want to create a release version because it's fully optimized and contains no debugging information.
- To change the version of the *Setup* program that Visual Studio creates, use the *Build→ Configuration Manager* command. Then, use the Configuration drop-down list for the *Web Setup* project to select the version you want to create.
- To install the web application, copy the *Setup.exe* and *Setup.msi* files to a network server or burn them to a CD or DVD. Then, run the Setup program on the server that will host the application.