



Visual Studio 2005 Technical Articles

Introduction to Web Application Projects

Microsoft Corporation

April 2006

Applies to:

Visual Studio 2005
Visual Studio .NET 2003

Summary: Find out how to use the new project type, the Web application project, as an alternative to the Web site project model already available in Visual Studio 2005. (27 printed pages)

Contents

[Introduction](#)

[Purpose of Web Application Projects](#)

[In This Paper](#)

[Installing Web Application Projects](#)

[Comparing Web Site Projects and Web Application Projects](#)

[Scenario 1: Creating a New Web Application Project](#)

[Step 1: Create a New Project](#)

[Step 2: Open and Edit the Page](#)

[Step 3: Build and Run the Project](#)

[Setting Build and Deployment Properties for Web Application Projects](#)

[Customizing Deployment Options for Web Application Projects](#)

[Scenario 2: Migrating a Visual Studio .NET 2003 Web Project to a Web Application Project](#)

[Step 1: Install the Visual Studio 2005 Web Application Project Preview](#)

[Step 2: Back Up Your Visual Studio .NET 2003 Projects](#)

[Step 3: Open and Verify your Visual Studio .NET 2003 Web Project](#)

[Step 4: Migrate the Solution to Visual Studio 2005](#)

[Step 5: Verify in Visual Studio 2005](#)

[Step 6: Covert Code-Behind Classes to Partial Classes](#)

[Step 7: Examine and Resolve XHTML Compliance Issues](#)

[The Future of Web Application Projects](#)

[Appendix A: Known Issues](#)

[Issue 1: Data Scenarios](#)

[Issue 2: Visual Basic Inline Code Might Not Be Converted Correctly](#)

[Issue 3: WSE and Web Application Projects](#)

[Issue 4: Converting the Club Web Site Starter Kit \(Visual Basic\)](#)

[Issue 5: Converting the Personal Web Site Starter Kit \(Visual Basic\)](#)

[Issue 6: Converting a Visual Studio 2005 Web Site project to a Web Application Project](#)

Introduction

The Web Application Projects add-in provides a Visual Studio 2005 Web project model option that works like the Visual Studio .NET 2003 Web project model. In this paper, the new project type is referred to as a Web application project. You can use Web application projects as an alternative to the Web site project model already available in Visual Studio 2005, which we refer to in this paper as Web site projects.

Purpose of Web Application Projects

The goal of Web application projects is to address some of the feedback we have heard from customers. Some developers find migrating Visual Studio .NET 2003 applications to the new Web site model in Visual Studio 2005 impractical, especially because precompiling (publishing) a Visual Studio 2005 Web site creates multiple assemblies.

The new Web application project type also enables some scenarios where Visual Studio 2005 Web projects are different than in the previous version of Visual Studio. For example, the new model has different semantics for Web subprojects where the subproject is not an ASP.NET or IIS application, but instead feeds its generated assembly to a parent application's Bin folder.

The new project type also provides a model that will feel familiar to developers who do not want to change how they structure their Web projects from how they use Visual Studio .NET 2003 today—for example, they want to continue to use a project file.

The new Web application project type does not replace the Web site project type introduced in Visual Studio 2005, which provides many new features and additional flexibility in how you manage Web applications. Instead, it is an alternative project type that you might choose depending on your requirements and your preferred development workflow. Some developers will find the default Visual Studio 2005 Web site project model natural and easy to use. Other developers will prefer a model in which project resources are defined explicitly (rather than implicitly by simply being in a folder) and in which they have tighter control over their project, and will therefore choose the new Web application project model. Rather than forcing developers to use just one project model, we will support both project models and allow developers to choose whichever Web project model works best for them.

Note Web application projects do not work with Visual Web Developer Express Edition.

In This Paper

This paper describes Web application projects and offers information on when you might choose between a Web application project and a Web site project model in Visual Studio 2005. The paper also walks you through the following common scenarios:

- Creating a new Web application project in Visual Studio 2005.
- Migrating an existing Visual Studio .NET 2003 project to a Visual Studio 2005 Web application project.

In addition, an appendix lists known issues with Web application projects.

Installing Web Application Projects

Adding Web application projects to Visual Studio 2005 requires you to install both an update and an add-in to Visual Studio 2005. The two installations perform the following tasks:

- The update makes changes to Visual Studio 2005 that are required so the Web project conversion wizard and designer will work well with Web application projects. You can download the update from the [Microsoft Visual Studio 2005 - Update to Support Web Application Projects](http://go.microsoft.com/fwlink/?linkid=63636) [<http://go.microsoft.com/fwlink/?linkid=63636>] page on the Microsoft Download Center Web site.
- The add-in makes the new Web application projects available in Visual Studio 2005. You can download it from the [Visual Studio 2005 Web Application Projects](http://go.microsoft.com/fwlink/?linkid=57541) [<http://go.microsoft.com/fwlink/?linkid=57541>] page on the ASP.NET Developer Center.

Early versions of Web application projects (V1 and V2 previews) were released in December 2005 and February 2006. The most recent version adds support for generating code to represent server controls and for converting projects directly from Visual Studio .NET 2003 to Web application projects in Visual Studio 2005.

Comparing Web Site Projects and Web Application Projects

The new Web application project model provides the same Web project semantics as Visual Studio .NET 2003 Web projects. This includes a structure based on project files and a build model where all code in the project is compiled into a single assembly. However, the new project type makes available all the new features of Visual Studio 2005 (refactoring, class diagrams, test development, generics, and so on) and of ASP.NET 2.0 (master pages, data controls, membership and login, role management, Web Parts, personalization, site navigation, themes, and so on).

The Web application project model in Visual Studio 2005 also removes two requirements from Visual Studio .NET 2003:

- Using FrontPage Server Extensions (FPSE). These are no longer required, but they are supported if your site already uses them.
- Using a local copy of IIS. The new project type supports both IIS and the built-in ASP.NET Development Server.

The following two tables describe the differences between Web application projects and Web site projects. The first table highlights various scenarios and tasks and suggests which model is best suited to that task. The second table describes in more detail the behavioral differences between each model. Use the tables to guide you in selecting which model to select.

The following table lists Web project options or tasks and indicates which project model best implements those options.

Option or Task	Web Application Projects	Web Site Projects
Need to migrate large Visual Studio .NET 2003 applications	X	
Prefer single-page code model to code-behind model		X
Prefer dynamic compilation and working on pages without building entire site on each page view (that is, save file and then simply refresh the page in the browser).		X
Need to control names of output assemblies	X	
Need to generate one assembly for each page		X
Need stand-alone classes to reference page and user control classes	X	
Need to build a Web application using multiple Web projects	X	
Need to add pre-build and post-build steps during compilation	X	
Want to open and edit any directory as a Web project without creating a project file		X

The following table helps you select a project type by describing some of the key differences between Web application projects and Web site projects.

Scenario	Web Application Project	Web Site Project
Project definition	<p>Similar to Visual Studio .NET 2003. Only files that are referenced in the project file are part of the project, are displayed in Solution Explorer, and are compiled during a build. Because there is a project file, some scenarios are more easily enabled:</p> <p>You can subdivide one ASP.NET application into multiple Visual Studio projects.</p> <p>You can easily exclude files from the project and from source code-control.</p>	<p>Web site projects use the folder structure to define the contents of the project. There is no project file and all files in the folder are part of the project.</p> <p>This project type is desirable if you have an existing folder structure representing an ASP.NET application</p>

		that you want to edit in Visual Studio without having to explicitly create a project file.
Compilation and build outputs	<p>The compilation model for Web application projects is very similar to that in Visual Studio .NET 2003.</p> <p>All code-behind class files and stand-alone class files in the project are compiled into a single assembly, which is placed in the Bin folder. Because this is a single assembly, you can specify attributes such as assembly name and version, as well as the location of the output assembly.</p> <p>Certain other applications scenarios are better enabled, such as the Model-View-Controller (MVC) pattern, because they allow stand-alone classes in the project to reference page and user control classes.</p>	<p>The Build command compiles Web site projects only to test them. To run Web site projects, you deploy source files and rely on ASP.NET dynamic compilation to compile pages and classes in the application.</p> <p>Alternatively, you can precompile the site for performance, which uses the same compilation semantics as ASP.NET dynamic compilation. The ASP.NET dynamic compilation system has two modes—batch mode (the default) and fixed-names mode. In batch mode, many assemblies (typically one per folder) are produced when precompiling the site. In fixed mode, one assembly is produced for each page or user control in the Web site.</p>
Iterative development	<p>To run and debug pages, you must build the entire Web project. Building the entire Web application project is usually fast, because Visual Studio employs an incremental build model that builds only the files that have changed.</p>	<p>You can configure build options Visual Studio 2005 for when you run the site: build the site, an individual page, or nothing at all. In the last case, when you run a Web site, Visual Studio simply launches the browser and passes to it the current or start page. The request then invokes ASP.NET dynamic compilation.</p> <p>Because pages are compiled dynamically and compiled into different assemblies as</p>

		<p>needed, it is not required that the entire project compile successfully in order to run and debug a page.</p> <p>By default, Visual Studio completely compiles Web site projects whenever you run or debug any page. This is done to identify compile-time errors anywhere in the site. However, a complete site build can significantly slow down the iterative development process, so it is generally recommended that you change the build project option to compile only the current page on run or debug.</p>
Deployment	<p>Because all class files are compiled into a single assembly, only that assembly needs to be deployed, along with the .aspx and .ascx files and other static content files.</p> <p>In this model, .aspx files are not compiled until they are run in the browser. However, when used with Web Deployment Projects (a downloadable add-in to Visual Studio 2005), the .aspx files can also be compiled and included in a single assembly for deployment.</p> <p>Each time you deploy the single assembly produced in this model, you replace the code for all pages in the project.</p>	<p>Both .aspx files and code-behind files can be compiled into assemblies using the Publish Website command in Visual Studio. (Note that the Build command does not create a deployable set of assemblies.) The updateable publish option supports compiling only code-behind files while leaving .aspx files unchanged for deployment.</p> <p>The default mode for precompiling produces several assemblies in the Bin folder, typically one per folder. The fixed-names option produces one assembly per page or user control and can be used to create deployable versions of individual pages. However, the fixed-names option increases the</p>

		number of assemblies and can result in increased memory usage.
Upgrade from Visual Studio .NET 2003	Because the Web application project model is the same as in the Visual Studio .NET 2003, upgrade is generally simple and will usually not require any restructuring of the application.	The compilation option for Web site projects is significantly different than Visual Studio .NET 2003. A conversion wizard is available to upgrade existing Visual Studio .NET 2003 Web projects to Web site projects. For any reasonably complex Visual Studio .NET 2003 projects, manual fix-up is usually required after the conversion. For most scenarios, it is preferable to upgrade existing Visual Studio .NET 2003 projects to Web application projects in Visual Studio 2005.

Scenario 1: Creating a New Web Application Project

This section walks you through creating a new Web application project. It also examines how page code is handled in a Visual Studio 2005 Web application project.

Note Web application projects do not work with Visual Web Developer Express Edition.

The examples shown here are in C#. The steps for working with Visual Basic are very similar, but file names and code will differ slightly.

Step 1: Create a New Project

To start, create a new Web application project. In Visual Studio, from the **File** menu, click **New Project**, displays the **New Project** dialog box.

Note To create a Web application project, you do not choose the **New Web Site** command, as you do to create a Web site project. Instead, you choose the **New Project** command.

Under **Project types**, open the **Visual C#** or **Visual Basic** node, and then under **Visual Studio installed templates**, click **ASP.NET Web Application**:

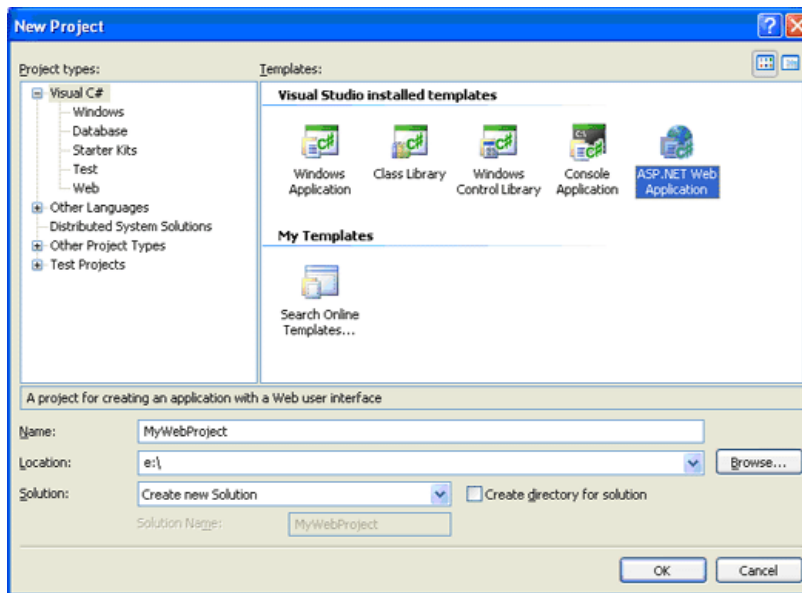


Figure 1. Creating a new Web Application Project

Name the project and specify a location. When you click **OK**, Visual Studio creates and opens a new Web project with a single page named `Default.aspx`, an `AssemblyInfo.cs` file (.vb file), and a `Web.config` file:

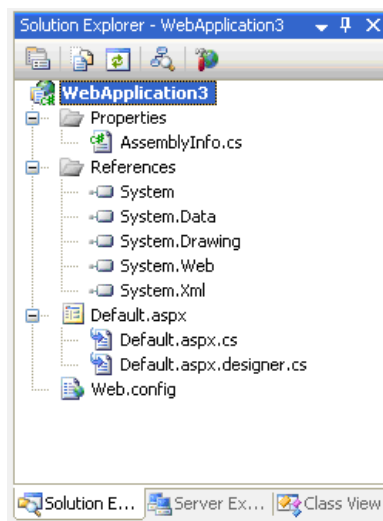


Figure 2. New Web application project

This list of project files, references, compilation information, and other metadata is stored in the project file, which is created in whatever location you specify in the **New Project** dialog box.

There are two class files (.cs files or .vb files) associated with the `Default.aspx` page. The `Default.cs` file (.vb file) contains the page class where you put your code and event handlers. The `Default.aspx.designer.cs` file (.vb file) is a new file used in Web application projects to contain the code that is generated and maintained by Visual Studio for the page. Both files contain a partial class. During compilation, the files are combined into a single code-behind class.

Step 2: Open and Edit the Page

Open the `Default.aspx` file and copy the following markup into it, which defines several ASP.NET server controls:

[Copy Code](#)

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication3._Default" %>

<html>
<head runat="server">
  <title>My Visual Studio 2005 Web Application Project Test</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>

      <h1>My Visual Studio 2005 web Application Project Test</h1>
      <h3>Pick a date: </h3>
    </div>
  </form>
</body>
</html>
```

```

<asp:Calendar ID="Calendar1" runat="server"
    BorderColor="#999999" DayNameFormat="Shortest"
    Font-Names="Verdana">
    <SelectedDayStyle BackColor="#666666"
        Font-Bold="True" ForeColor="White" />
    <TodayDayStyle BackColor="#CCCCCC" ForeColor="Black" />
    <SelectorStyle BackColor="#CCCCCC" />
    <WeekendDayStyle BackColor="#FFFACC" />
    <OtherMonthDayStyle ForeColor="#808080" />
    <NextPrevStyle VerticalAlign="Bottom" />
    <DayHeaderStyle BackColor="#CCCCCC" Font-Bold="True"
        Font-Size="7pt" />
    <TitleStyle BackColor="#999999" BorderColor="Black"
        Font-Bold="True" />
</asp:Calendar>
<br />
<div>
    <asp:Label ID="Label1" runat="server" Text="Label"/>
</div>
</div>
</form>
</body>
</html>

```

In a Web application project, the Default.aspx.designer.cs file is automatically updated when you add or remove controls in the Default.aspx file. The following snippet shows what the Default.aspx.designer.cs file looks like after you add the preceding markup:

[Copy Code](#)

```

namespace WebApplication3
{
    public partial class _Default
    {
        protected System.Web.UI.HtmlControls.HtmlForm form1;
        protected System.Web.UI.WebControls.Calendar calendar1;
        protected System.Web.UI.WebControls.Label label1;
    }
}

```

The Default.aspx.designer.cs file contains a control declaration for each server control in the .aspx (or .ascx) file. By default, the controls are declared as **protected**. You can open the .designer.cs file and change a field's accessor to **public** if you want to access the variable from outside of its class.

Note It is not recommended that you set the accessibility of control declarations to **public**, because it is a better to use a property to change the accessibility of a server control variable. However, editing the access modifier for server variables in the designer file is supported for backward compatibility with Visual Studio .NET 2003 Web projects that used this coding technique.

In Web application projects, code generation for server controls is improved over the Visual Studio .NET 2003 experience in two important ways:

- You no longer have to switch the page to Design view to update the control declarations. The designer monitors both Design view and Source view and updates declarations appropriately.
- Control declarations in the base class of a page are honored and are not duplicated in a page's code-behind class.

Because the .designer.cs file is automatically updated with the control declarations, you can go directly to the Default.aspx.cs code-behind class file and program any of the controls on the page, as in the following example:

[Copy Code](#)

```

using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

namespace WebApplication3 {
    public partial class _Default : System.Web.UI.Page {
        protected void Page_Load(object sender, EventArgs e) {
            Label1.Text = "Current date: " + Calendar1.SelectedDate;
        }
    }
}

```

Step 3: Build and Run the Project

Run the project in debug mode by pressing **F5** or clicking **Run** in the **Debug** menu. By default, Web application projects use the built-in ASP.NET Development Server using a random port as the root site (for example, the site's URL might be `http://localhost:12345/`):



Figure 3. Running a Web application project

When you build Web application projects, all code-behind classes, stand-alone class files, and embedded resources are compiled into a single assembly that is placed in the project's Bin folder.

You can change the output location if you want to—for example, to build the project into a parent application directory. Typically, you set the build location when you have multiple subprojects in subfolders of a single ASP.NET application. Select the properties of the project to set the build location, as shown in the following screenshot:

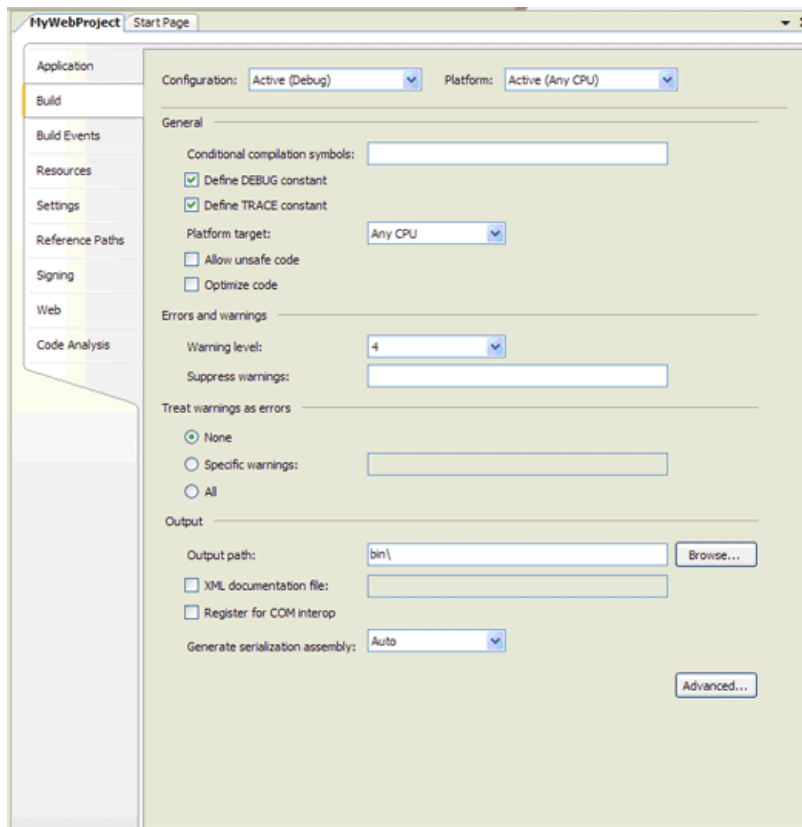


Figure 4. Setting output build location

After building the project, you can examine the results. In Solution Explorer, click **Show All Files**:

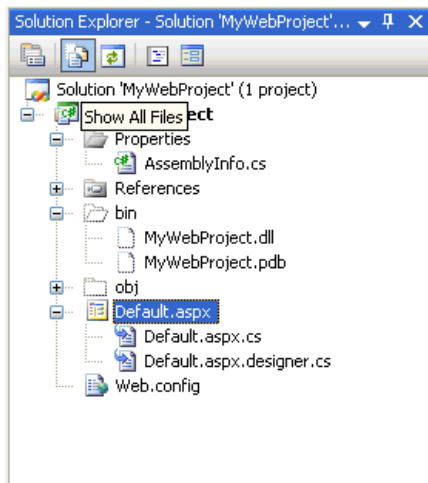


Figure 5. Results of building a Web application project

This works the same as it does in Visual Studio .NET 2003 ASP.NET Web projects.

Setting Build and Deployment Properties for Web Application Projects

ASP.NET Web application projects use the same configuration settings and behaviors as Visual Studio 2005 class-library projects. You can access these configuration settings by right-clicking the project name in Solution Explorer and selecting **Properties**. This displays the project properties editor. You can use the editor to change properties such as the name of the generated assembly, the build compilation settings for the project, its references, its resource string values, and code-signing settings:

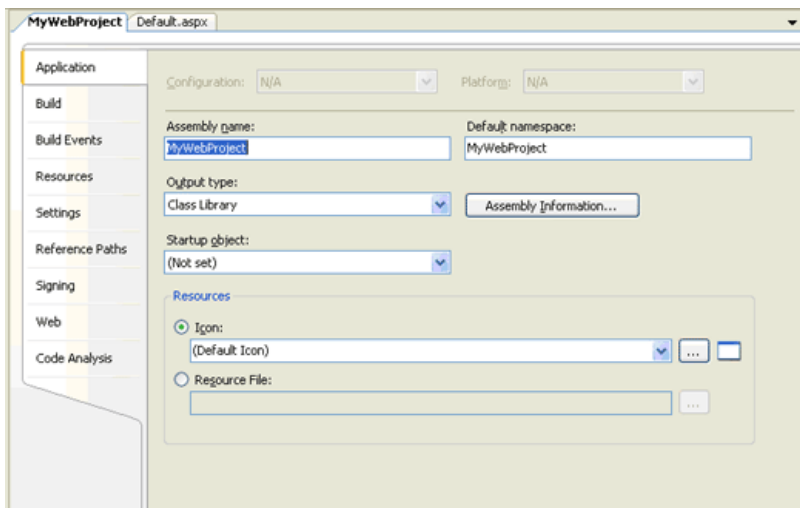


Figure 6. Changing properties of a Web application project

ASP.NET Web application projects add a tab named **Web** to the project properties list. You can use this tab to configure how a Web project is run and debugged. By default, ASP.NET Web application projects are configured to launch and run using the built-in ASP.NET Development Server on a random HTTP port on the machine. The following screenshots show using the ASP.NET Development Server: the Taskbar icon, and a page in the browser where you can see the server's URL in the **Address** box.

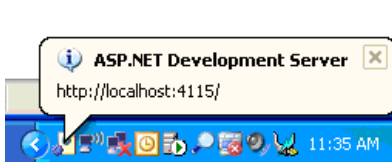


Figure 7. Taskbar icon for ASP.NET Development Server

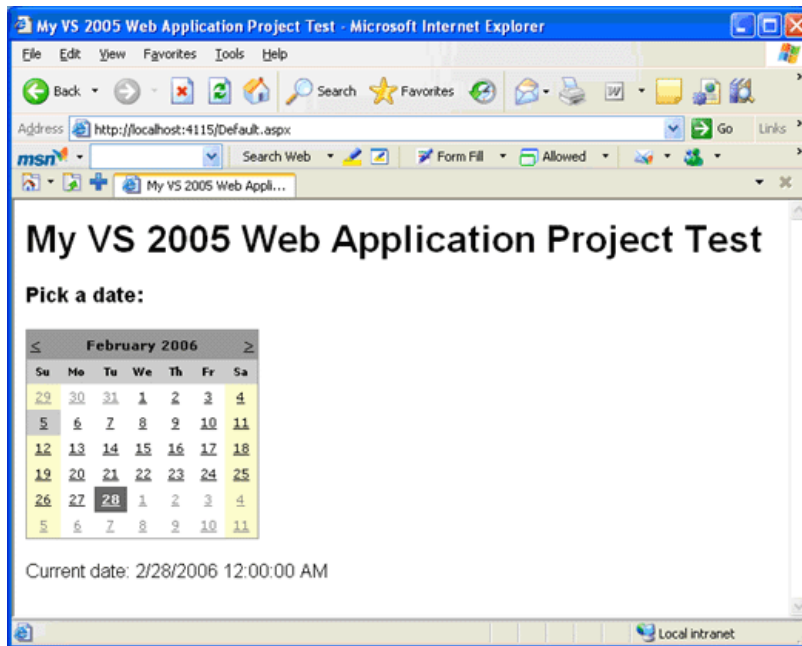


Figure 8. Running a Web application project using the ASP.NET Development Server

You can change the port number if the selected port is already in use or if you want to run the Web project on a specific port:

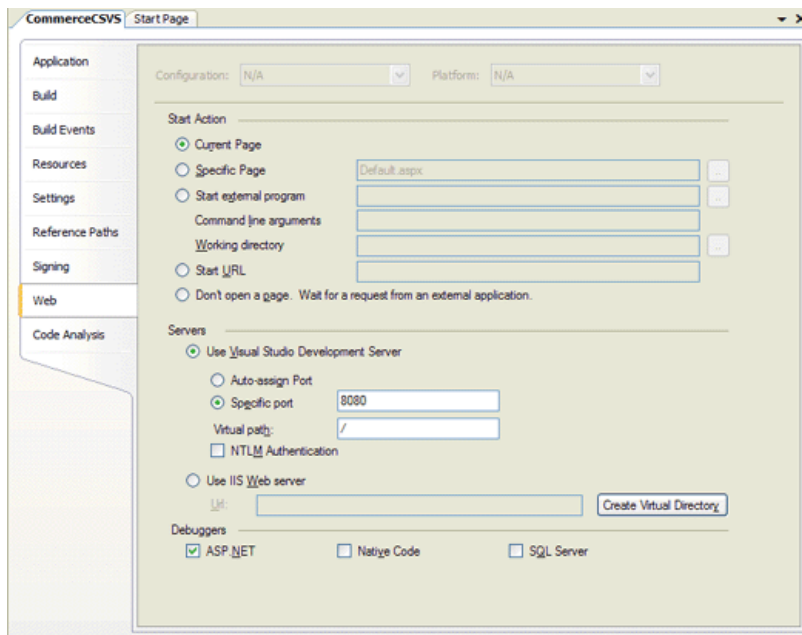


Figure 9. Changing the port used by the ASP.NET Development Server

Alternatively, Visual Studio can run and debug the Web application using IIS. To use IIS, select **Use IIS Web Server** and enter the URL of the Web application:

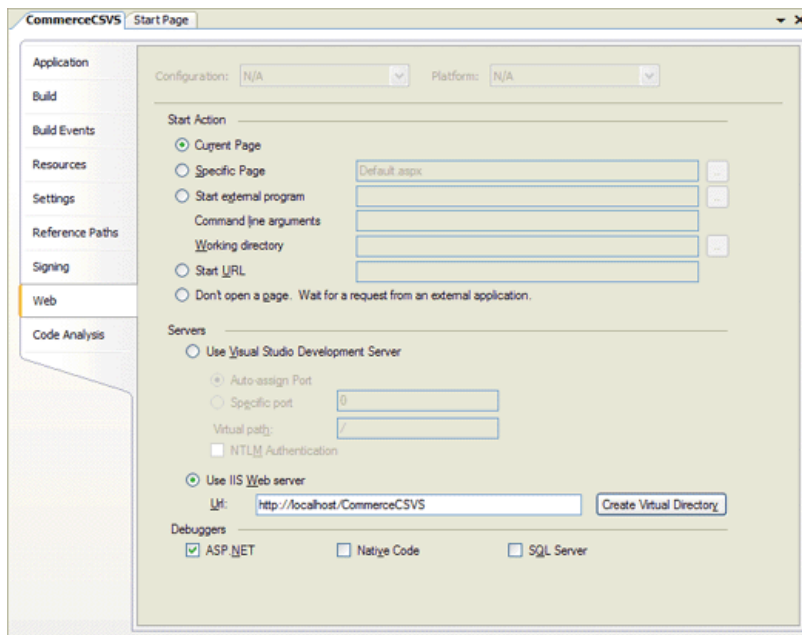


Figure 10. Configuring a Web application project to use IIS

Note ASP.NET Web application projects do not automatically create an IIS virtual root or application for you. To create the virtual directory and application in IIS, click the **Create Virtual Directory** button next to the **Url** box.

When you run the application, Visual Studio compiles the project into a single assembly, which follows the build semantics of Visual Studio .NET 2003 Web projects. When the application is running, Visual Studio attaches a debugger to the Web server process.

All project settings are saved in a standard MSBuild project file.

Customizing Deployment Options for Web Application Projects

After creating a Web application project, you can optionally use a Visual Studio 2005 Web Deployment project to set deployment options for the Web application project.

Note Web Deployment Projects is available as a Visual Studio 2005 add-in. You can download the add-in from the [Visual Studio 2005 Web Deployment Projects](http://go.microsoft.com/fwlink/?linkid=55111) [<http://go.microsoft.com/fwlink/?linkid=55111>] page on the ASP.NET Developer Center.

Web application projects compile all code into a single assembly, so you do not need a Visual Studio 2005 Web Deployment project to combine assemblies. However, Web Deployment projects provide additional support for precompiling the .aspx content of a project, as well as for making post-build changes to the published configuration. Using the latest build of the Web Deployment project add-in, you will be able to do add these features for both Visual Studio 2005 Web site projects and Visual Studio 2005 Web application projects.

Scenario 2: Migrating a Visual Studio .NET 2003 Web Project to a Web Application Project

This section walks you through the process of converting an existing Visual Studio .NET 2003 Web project to a Web application project in Visual Studio 2005. The Web application project model uses the same conceptual approach as a Web project in Visual Studio .NET 2003, including a project file to include and exclude files, compilation to a single assembly, and so on. In general, Web application projects do not require any architectural changes after the conversion.

As in the preceding section, this section assumes that you are working in C#. The steps for working with Visual Basic are very similar, but file names and some of the code will differ.

Step 1: Install the Visual Studio 2005 Web Application Project Preview

Be sure you have installed Web Application Projects in Visual Studio 2005 by following the steps in the [Installing Web Application Projects](#) section earlier in this paper.

Step 2: Back Up Your Visual Studio .NET 2003 Projects

Make absolutely sure to save a backup of your Visual Studio .NET 2003 Web projects and solutions before attempting any migration. The following steps have been tested using the preview version of Web Application Projects. However, there could still be issues, and you might need to restore the Visual Studio .NET 2003 solution.

Step 3: Open and Verify your Visual Studio .NET 2003 Web Project

Before migrating a project, open your existing Visual Studio .NET 2003 solution, compile it, and then run it. Spending a few minutes to verify that everything works before you migrate can save trouble later, especially if the cause of errors is a last-minute directory change or similar.

Step 4: Migrate the Solution to Visual Studio 2005

Close the solution in Visual Studio .NET 2003, and then start Visual Studio 2005. On the **File** menu, click **Open File**, and then browse to the .sln file for the solution you want to migrate. This launches the Visual Studio 2005 Conversion wizard:



Figure 11. Visual Studio 2005 conversion wizard

Click **Next** to proceed through the wizard, accepting all defaults. Visual Studio 2005 will convert the solution and its contained project files to use the MSBuild format, which is the new project file format in Visual Studio 2005.

As part of the migration, Visual Studio 2005 generates an XML-based log file that provides a summary of the conversion process, flagging any issues encountered during migration. By default, the conversion log file is saved in the same directory as the .sln file. If you have issues later when compiling the project, you might need to refer back to the conversion log file.

Step 5: Verify in Visual Studio 2005

After the solution and project files are upgraded to Visual Studio 2005 format, validate that your application builds without errors and runs as expected. At this point, the most common errors or warnings will be of these types:

- Conflicts with names in introduced in the .NET Framework version 2.0.
- Warnings about using obsolete members.

To fix naming conflicts, you can either fully qualify existing names with a namespace to remove ambiguity, or rename the members so they do not conflict.

If you see a warning about using obsolete members, the warning message will usually suggest alternative APIs to use. In these cases, you can continue to use the obsolete members in the 2.0 version of the .NET Framework. However, the members will be removed in the next major release of the .NET Framework, so it is a good practice to remove the members and substitute the suggested alternatives.

When running your Web application, you might see an error that says "Directory Listing Denied" that suggests that a virtual directory does not allow its contents to be listed. If you see this error message, do the following:

1. Close the browser.
2. In Solution Explorer, right-click the application's start page and then click **Set as Start Page** to ensure that the correct page is invoked when the application runs.
3. Run the application again.

Step 6: Covert Code-Behind Classes to Partial Classes

As noted earlier, in Visual Studio 2005, Web application projects use partial classes to store designer-generated code. These classes are stored in a separate file from the code-behind file. By default, the Visual Studio 2005 Web project conversion wizard does not create a *.designer.cs (or *.designer.vb) file for Web pages (.aspx files) or user controls (.ascx files). Instead, the code will look and work just like it did in Visual Studio .NET 2003.

Note The wizard makes the minimum number of changes to the code files during migration to help ensure a smooth conversion to Web application projects in Visual Studio 2005.

You can keep your code in this format. If you do, you must manually update the control field declarations in the code-behind files, but everything else will work fine in Visual Studio 2005.

To take advantage of the editor's ability to maintain field declarations in generated code, you should update your pages and controls to use the partial-class model introduced in ASP.NET 2.0. Partial classes make it easier to organize the generated code and custom code for your code-behind files. For more information on .designer.cs files, see [Scenario 1: Creating a New Web Application Project](#) earlier in this paper.

To migrate your code to use partial-class model, first make sure that your code compiles without errors. Then in Solution Explorer, right-click the project name and click **Convert to Web Application**. This command iterates through each page and user control in the project, moves all control declarations to a .designer.cs file, and adds event handler declarations to the server-control markup in the .aspx and .ascx files.

Note Another option is to use the **Convert to Web Application** command on individual pages. You might do this first on a few pages if you want to watch closely the changes made for each page before applying the change to the entire application.

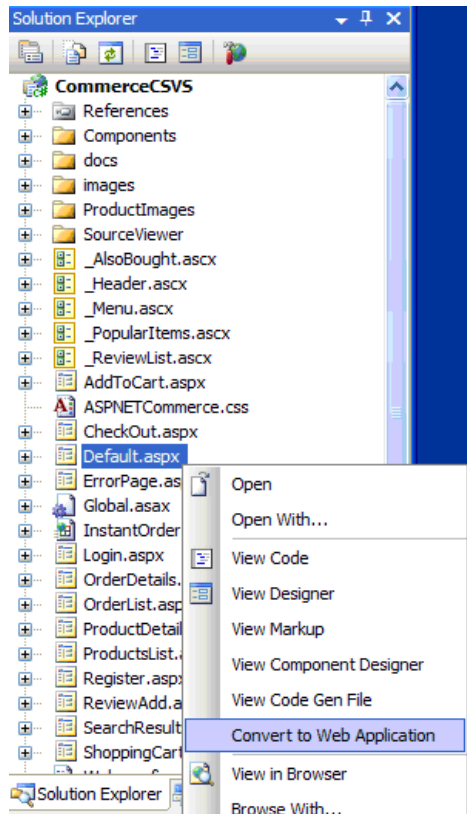


Figure 12. Converting a single page to Web application project format

When the process has finished, check the **Task List** window to see if any conversion errors are reported. If the task list displays errors, right-click the relevant page in Solution Explorer and choose **View Code** and **View Code Gen File** to examine the code and fix problems. Note that errors and warnings that appear in the **Task List** window persist between Visual Studio sessions. After you have fixed errors listed in the window, you can clear items from the task list.

Note The **Convert to Web Application** command cannot be undone in Visual Studio. The best method for reverting the changes is to restore from a backup of the Visual Studio .NET 2003 project, and then to re-run the Visual Studio 2005 migration as described in Step 4 above.

Now recompile your project to ensure it compiles without errors. A possible source of errors is if you have modified the control declarations in a code-behind class and the conversion wizard mishandles your changes.

From this point, when you add new pages into your Web project, they will by default use the partial-class template.

Step 7: Examine and Resolve XHTML Compliance Issues

By default, Visual Studio 2005 generates and validates XHTML-compliant markup. This helps you build Web applications that are standards compliant and helps minimize issues with browser-specific rendering. Visual Studio .NET 2003 did not generate XHTML-compliant markup, so you might see validation and rendering issues with pages created in Visual Studio .NET 2003.

Note Validation errors are informational only and are flagged as warnings. Validation errors do not prevent a page from running.

If you do not want to see validation errors, switch the HTML validation setting from **XHTML Transitional to Internet Explorer 6.0** (which was the Visual Studio .NET 2003 default setting). In the **Tools** menu, click **Options**. In the **Options** dialog box, open the **Text Editor** node, then the **HTML** node, and then the **Validation** node. In the **Target** list, select **Internet Explorer 6.0**, and then unselect the **Show Errors** check box. Note that this does not fix XHTML validation errors; it simply switches the validation schema to one that is more compatible with the way markup was generated by Visual Studio .NET 2003, and it suppresses validation warnings.

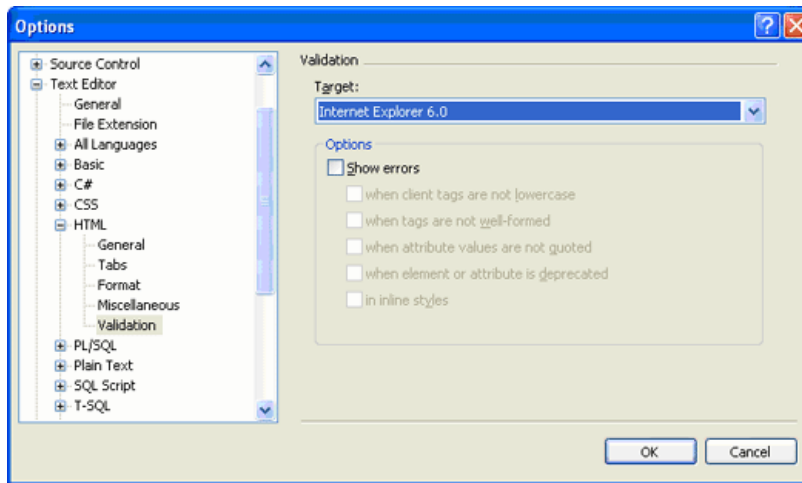


Figure 13. Configuring HTML validation

You can also add the following section to your project's Web.config file, which causes ASP.NET to render legacy (non-XML compliant) markup from server controls:

```
<system.Web>
<xhtmlConformance mode="Legacy" />
</system.Web>
```

This will avoid the slight rendering differences you might see between pages displayed using ASP.NET 1.1 and using ASP.NET 2.0. For more information, see [ASP.NET and XHTML](http://go.microsoft.com/fwlink/?linkid=64052) [http://go.microsoft.com/fwlink/?linkid=64052] in the MSDN Library.

The Future of Web Application Projects

Web application projects provide a compilation and build model very similar to the one used in Visual Studio .NET 2003. Depending on their requirements, some users will find the new Web site project option in Visual Studio 2005 more appropriate for their applications, while other users will prefer the Web application project option. Web application projects provide the best path for upgrading existing Visual Studio .NET 2003 applications to Visual Studio 2005, and are highly recommended for that scenario.

We want to emphasize these important points about the future of Visual Studio 2005 Web application projects:

- Going forward, we will fully support both the Visual Studio 2005 Web site project model and Visual Studio 2005 Web application project model. You can choose whichever model works best for you.
- In future versions of Visual Studio, the Web application project model will be built in, and both the Web application project model and Web site project model will be supported.

Appendix A: Known Issues

This appendix lists known issues with Web application projects.

Issue 1: Data Scenarios

- There are known issues when using data-bound controls and SQL Server 2005 Express with the April 2006 release of Web application projects. For a list of issues and workarounds, see the whitepaper named "Using Data-Bound Controls and SQL Server Express with Web Application Projects," which is available on the [Visual Studio 2005 Web Application Projects](http://go.microsoft.com/fwlink/?linkid=57541) [http://go.microsoft.com/fwlink/?linkid=57541] page on the ASP.NET Developer Center.

Issue 2: Visual Basic Inline Code Might Not Be Converted Correctly

When you upgrade a Visual Studio .NET 2003 project to Visual Studio 2005, Visual Basic code in single-file Web pages (.aspx files) or user controls (.ascx files) is not converted to use the new Visual Basic 2005 syntax. This can lead to compile time errors when building your converted Web project. You will need to manually fix these errors and recompile the affected pages. Visual Basic code in code-behind files is converted and should compile correctly.

Issue 3: WSE and Web Application Projects

Web Services Enhancements (WSE) is an add-in to Visual Studio 2005 that provides the latest advanced Web services capabilities. When Web Application Projects is installed, the WSE configuration command on the shortcut menu in Solution Explorer will not work correctly. As a workaround, follow these steps to use the stand-alone WSE 3.0 configuration tool instead:

1. Start the configuration tool. In the Windows **Start** menu, click **All Programs**, click **Microsoft WSE 3.0**, and then click **Configuration Tool**.
2. In the configuration tool, in the **File** menu, click **Open**.
3. Select the Web.config file for the project.

Issue 4: Converting the Club Web Site Starter Kit (Visual Basic)

(This issue applies only when using Visual Basic.) When converting the Visual Basic version of the Club Web Site starter kit to a Web application project, you will see a run-time error similar to the following:

[Copy Code](#)

```
Server Error in '/' Application.  
Compiler Error Message: BC30451: Name 'truncate' is not declared.  
Source Error:  
Line 129: <asp:Label ID="Label2" runat="server" Text='<## truncate(CStr(Eval("description")))' %>' />  
Source File: C:\vs05\328\test-club1\wap1\Default.aspx Line: 129
```

To resolve the error, you must add a namespace. You can do this for individual pages or for the project as a whole. Assuming your project namespace is "wap1", follow these steps.

To add a namespace to an individual page:

1. Open the page.
2. Under the last **@ Register** directive, add an **@ Import** directive that references the namespace you want to use, as in the following example:

[Copy Code](#)

```
<%@ Register TagPrefix="Club" Namespace="wap1.Clubsite" Assembly="wap1">  
<%@ Import Namespace="wap1" %>
```

To add a namespace to the project:

1. Open the Web.config file.
2. Add or edit the **<namespaces>** element as a child of the **<pages>** element, as in the following example:

[Copy Code](#)

```
<pages>  
  <namespaces>  
    <add namespace="wap1"/>  
  </namespaces>  
</pages>
```

Issue 5: Converting the Personal Web Site Starter Kit (Visual Basic)

(This issue applies only when using Visual Basic.) When converting the Visual Basic version of the Personal Web Site starter kit to a Web application project, you will see a run-time error similar to the following:

[Copy Code](#)

```
Server Error in '/' Application  
The type specified in the TypeName property of ObjectDataSource 'ObjectDataSource1' could not be found
```

To resolve the error, you must add the project namespace to the **TypeName** property. Assuming your project namespace is "wap1", follow these steps.

1. Open the Member_List.aspx page.
2. Change the type name in the **ObjectDataSource** declaration to be fully qualified, as in the following example:

[Copy Code](#)

```
<asp:ObjectDataSource TypeName="wap1.MemberDetails" >
```

Issue 6: Converting a Visual Studio 2005 Web Site project to a Web Application Project

This paper will be updated in the future to show the steps needed to convert a Visual Studio 2005 Web site project to a Web application project. In the meantime, see the entry, [Tutorial 8: Migrating from VS 2005 Web Site Projects to be a VS 2005 Web Application Projects](http://webproject.scottgu.com/csharp/migration2/migration2.aspx) [<http://webproject.scottgu.com/csharp/migration2/migration2.aspx>] on Scott Guthrie's blog.