

Module 2: Using Microsoft Visual Studio .NET

Contents

Overview	1
Lesson: Overview of Visual Studio .NET	2
Lesson: Creating an ASP.NET Web Application Project	22
Review	32
Lab 2: Using Microsoft Visual Studio .NET	34



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2002 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, Windows, Windows NT, Active Directory, ActiveX, BizTalk, Hotmail, IntelliSense, JScript, MSN, MSDN, PowerPoint, Visio, Visual Basic, Visual C++, Visual C#, Visual J#, Visual Studio, Win32, and Windows Media are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Sample Courseware
Not for Commercial Use or Redistribution

Overview

- Overview of Visual Studio .NET
- Creating an ASP.NET Web Application Project

Introduction

In this module, you will learn how to use the primary features of Microsoft® Visual Studio® .NET to create Microsoft ASP.NET Web applications.

Visual Studio .NET is the comprehensive development environment that can be used to create powerful, reliable, enterprise Web solutions. By offering end-to-end Web development capabilities and scalable, reusable server-side components, Visual Studio .NET will increase your productivity and help you to more effectively create applications and ASP.NET Web sites.

Objectives

After completing this module, you will be able to:

- Navigate the Visual Studio .NET integrated development environment (IDE).
- Create, build, and view an ASP.NET Web application project.

Lesson: Overview of Visual Studio .NET

- Why Visual Studio .NET?
- Start Page
- Available Project Templates
- Practice: Select the Project Template
- Integrated Development Environment (IDE)
- Demonstration: Using the Visual Studio .NET IDE
- Practice: Using the Visual Studio .NET IDE

Introduction

This lesson introduces the Visual Studio .NET IDE. The IDE is the common user interface (UI) and set of tools that are used for all of the different project types and programming languages that are supported by Visual Studio .NET.

Lesson objectives

After completing this lesson, you will be able to:

- Explain the purpose of Visual Studio .NET.
- Explain the IDE opening screen links.
- Identify the available project types and templates.
- Identify the available windows in the IDE.

Why Visual Studio .NET?

- One IDE for multiple languages and multiple project types
- Multiple languages within a project
- Multiple project types within a solution
- Integrated browser
- Debugging support
- Customizable interface

Introduction

Visual Studio .NET simplifies the development of powerful, reliable enterprise Web solutions and increases developer efficiency by providing a familiar, shared development environment. Pre-built components, programming wizards, and the ability to reuse components that are written in any language can reduce development time significantly. Microsoft IntelliSense®-based code completion enables you to produce accurate code more quickly. Powerful, end-to-end, cross-language debugging support, together with cross-language debugging, helps you make your applications operational.

One IDE

Visual Studio .NET has a single IDE that provides a consistent look and feel, regardless of the programming language being used or the application type being developed. Features that were available for only one language are now available to all languages.

Multiple languages

Visual Studio .NET supports development in a number of the Microsoft .NET-based languages. This support of various and diverse languages allows developers to work in their own preferred language, because they no longer need to learn a new language for each new project.

The languages that are included with Visual Studio .NET are:

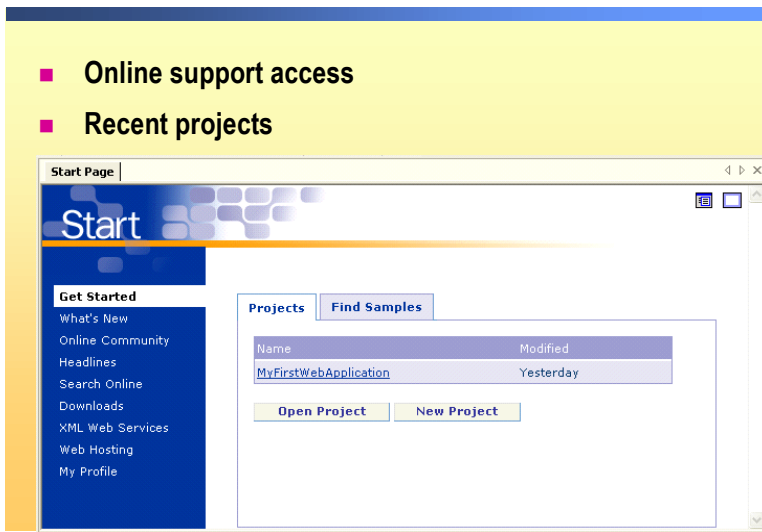
- Microsoft Visual Basic® .NET
- C#
- Microsoft Visual C++®

Note For more information on the available .NET-based languages, see Module 3, “Using Microsoft .NET-Based Languages,” in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*.

Multiple project types	<p>Visual Studio .NET supports the development of multiple project types, ranging from Microsoft Windows®-based applications, to ASP.NET Web applications, to XML Web services.</p> <p>This support for multiple project types allows you to simultaneously work on several projects without having to change development environments and learn new tool interfaces or languages.</p>
Integrated browser	<p>Visual Studio .NET contains a built-in browser that is based on Microsoft Internet Explorer. The browser is integrated into the IDE and can be accessed from multiple windows and menus.</p> <p>This browser accessibility allows you to view your Web site during the development cycle without having to transfer to another program and retype Uniform Resource Locator (URL) strings.</p>
Debugging support	<p>Visual Studio .NET is designed to support debugging from your initial code through to the application release. Debugging support includes breakpoints, break expressions, watch expressions, and the ability to step through code one statement or one procedure at a time.</p> <hr/> <p>Note For more information on debugging, see Module 6, “Tracing in Microsoft ASP.NET Web Applications,” in Course 2310B, <i>Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET</i>.</p> <hr/>
Customizable interface	<p>Because the Visual Studio .NET IDE can be customized at the window and tool level, you can show only those tools or windows that you are using at any given time and hide the remainder.</p>

Sample Courseware
Not for Commercial Use or Redistribution

Start Page



Introduction

Each time you start Visual Studio .NET, the Start page is displayed. This page provides an essential location for setting preferred options, reading product news, accessing discussions with fellow developers, and obtaining other information that can be used to get started within the Visual Studio .NET environment.

You can view the Start page at any time when you are working in the development environment.

► To view the Start Page

- On the **Help** menu, click **Show Start Page**.

Get Started

Clicking **Get Started** sets Dynamic Help to display topics about starting new projects, and displays the following two folders:

■ Projects

The Projects folder displays links to the latest projects that you have been working on. This folder allows you to quickly open Visual Studio .NET and load all of the files that are related to your current projects.

■ Find Samples

The Find Samples folder displays a search engine that finds code samples by language and keyword online from `ms-help://MS.VSCC`.

What's New

Clicking **What's New** provides you with access to updates on Visual Studio Resources, Partner Resources, and Product Information.

Online Community

Clicking **Online Community** provides you with access to the Microsoft Visual Studio .NET Web sites and related newsgroups.

Headlines

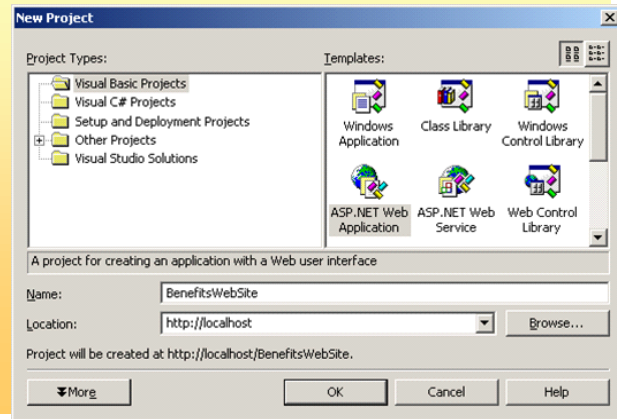
Clicking **Headlines** provides you with access to Visual Studio .NET features, technical articles, and the Microsoft Visual Studio .NET Knowledge Base.

Search Online	Clicking Search Online provides you with a search engine that accesses the Microsoft MSDN® Online Library.
Downloads	Clicking Downloads provides you with access to Visual Studio .NET-related downloads, code samples, and reference materials.
XML Web Services	Clicking XML Web Services provides you with tools to find an XML Web service by running a query in a directory of XML Web services called Universal Description, Discovery, and Integration (UDDI) Service. In addition, you can use the XML Web Services selection to register an XML Web service in the UDDI directory.
Web Hosting	Clicking Web Hosting provides you with a list of links to hosting providers.
My Profile	<p>Clicking My Profile allows you to set a user profile that adjusts the Toolbox, Default New Project, and Dynamic Help to match your programming preferences. You can change your profile at any time to modify these custom settings.</p> <p>You have the option of choosing a pre-existing profile, such as Visual Basic Developer, or modifying each profile item manually.</p>

Sample Courseware
Not for Commercial Use or Redistribution

Available Project Templates

- The list of available project templates is based on your Profile and Project Type selections



Introduction

Visual Studio .NET provides templates that support the creation of a number of common project types. These templates contain all of the required files and work with your profile to adjust the IDE into the correct configuration for the selected project.

These templates allow you to make use of your time by focusing on adding functions to the project and avoiding having to set up the infrastructure every time you change project types.

Solutions and projects

When you create a project in Visual Studio .NET, you also create a larger container called a Solution. This Solution can contain multiple projects in the same way that a project container can contain multiple pages.

Solutions allow you to concentrate on the project or set of projects that are required to develop and deploy your application, instead of having to focus on the details of managing the objects and files that define them. In using the concept of a *Solution* as a container, a solution allows you to:

- Work on multiple projects within the same instance of the IDE.
- Work on items, settings, and options that apply to a group of projects.
- Manage miscellaneous files that are opened outside the context of a Solution or a Project.
- Use Solution Explorer, which is a graphical view of your solution, to organize and manage all of the projects and files that are needed to design, develop, and deploy an application.

Project templates

Visual Studio .NET includes multiple project templates that are sorted by language and type. To select the correct template, you first need to specify the language in which you will be working.

The available Visual Basic project and Microsoft Visual C#™ project templates include:

Project templates	Description
Windows Application	The Windows Application project template is used to create standard Windows applications. This template automatically adds the essential project references and files that are needed as a starting point for your application.
Class Library	The Class Library template is used to create reusable classes and components that can be shared with other projects.
Windows Control Library	The Windows Control Library template is used to create custom controls that are to be used on Windows Forms.
ASP.NET Web Application	The ASP.NET Web Application project template is used to create an ASP.NET Web application on a computer that has Internet Information Services (IIS) version 5.0 or later installed. This template creates the basic files that are needed on the server to help you start designing your application.
ASP.NET Web Service	The ASP.NET Web Service project template is used to write an XML Web service that can be consumed by other Web services or applications on a network. XML Web services are components that are available over the Internet and are designed to only interact with other Web applications.
Web Control Library	The Web Control Library template is used to create custom Web server controls. This template adds the necessary project items that are needed to start creating a control that can then be added to any Web project.
Console Application	The Console Application project template is used to create console applications. Console applications are typically designed without a graphical UI and are compiled into a stand-alone executable file. A console application is run from the command line with input and output information being exchanged between the command prompt and the running application.
Windows Service	The Windows Service template is used to create Windows Service applications, which are long-running executable applications that run in their own Windows session.
Empty Project	The Empty Project template is used to create your own project type. The template creates the necessary file structure that is needed to store application information. Any references, files, or components must be manually added to the template.

(continued)

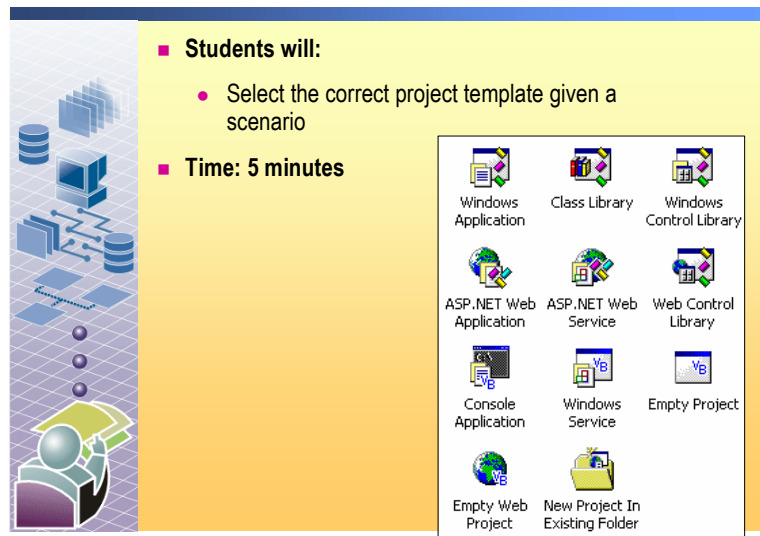
Project templates	Description
Empty Web Project	The Empty Web Project template is for advanced users who want to start with an empty project. This template creates the necessary file structure for a server-based project on an IIS server. References and components (such as Web Forms pages) must be added manually.
New Project in Existing Folder	The New Project in Existing Folder project template is used to create a blank project within an existing application directory. You can then choose to add the files from the preexisting application directory into this new project by right-clicking each of these items in Solution Explorer and selecting Include in Project on the shortcut menu.

The following table lists the additional project templates that are available in Visual Studio .NET.

Template Group	Description
Visual C++ Projects	Templates include: <ul style="list-style-type: none"> ▪ Active Template Library (ATL) projects ▪ Managed Applications ▪ Managed Class Library ▪ Managed Web Services
Setup and Deployment Projects	The Setup and Deployment Projects template allows you to create installers to distribute an application. The resulting Windows Installer (.msi) file contains the application, any dependent files, information about the application, such as registry entries, and instructions for installation.
Other Projects	Templates include: <ul style="list-style-type: none"> ▪ Database projects ▪ Enterprise projects ▪ Extensibility projects ▪ Application center test projects

Note The Class Library, Empty project, New Project in Existing Folder, Web Control Library, Windows Service, Windows Control Library, and Empty Web Project templates are not available in the Standard Edition of Visual Studio .NET.

Practice: Select the Project Template



■ **Students will:**

- Select the correct project template given a scenario

■ **Time: 5 minutes**

Windows Application	Class Library	Windows Control Library
ASP.NET Web Application	ASP.NET Web Service	Web Control Library
Console Application	Windows Service	Empty Project
Empty Web Project	New Project In Existing Folder	

► **Choose the appropriate project template for each of the following scenarios**

You want to create a control with a UI that you can reuse in any Windows application.

Windows Control Library

You want to build an application that will run on a single computer running Windows.

Windows Application

You want to create a dynamic Web application that includes Web pages and may use XML Web services.

ASP.NET Web application

You want to create a reusable component that is accessible to several Windows or Web applications.

Class Library

You want to create a user-defined Web control that can be used in several Web pages.

Web Control Library

You want to create an application that will run from a command line.

Console Application

You want to create a class in which the methods are accessible through the Internet by any Web application.

ASP.NET Web Service

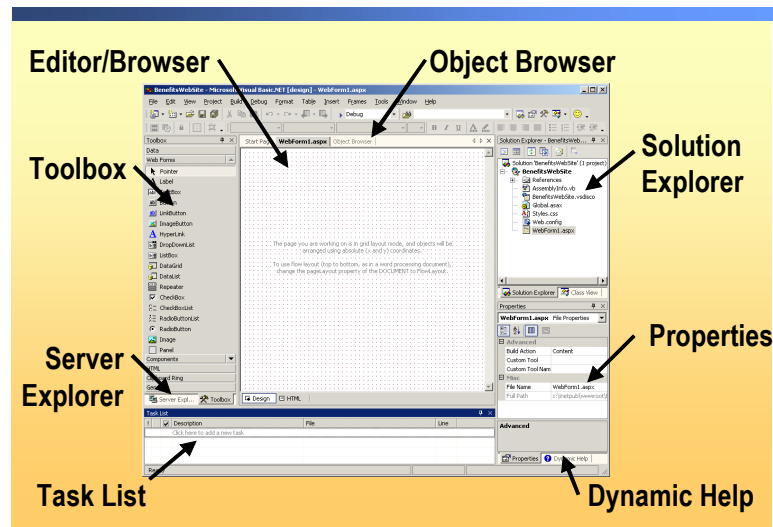
You want to create a Windows service that will run continuously, regardless of whether a user is logged on or not.

Windows Service

Previous versions of Visual Basic require you to use third-party products or low-level application programming interface (API) calls to create these types of applications.

Sample Code
Not for Commercial Use
Redistribution

Integrated Development Environment (IDE)



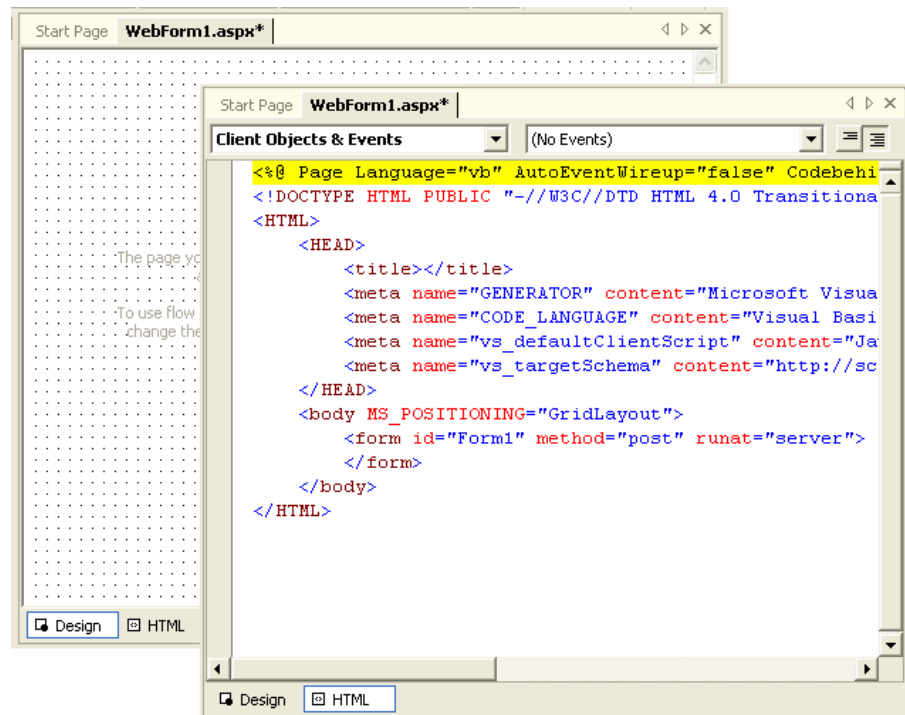
Introduction

The Visual Studio .NET IDE contains multiple windows that provide a variety of tools and services. Many of the features of Visual Studio .NET are available from several of the IDE windows, menus, and toolbars.

You can move or hide IDE windows depending on your personal preference. Use the **View** menu to select which windows to display. Click the thumbtack **Auto Hide** button to turn static windows into pull-out windows.

Editor/browser

The editor/browser is the primary interface window in Visual Studio .NET. In editor mode, the editor/browser displays code for editing and provides a What You See Is What You Get (WYSIWYG) graphical interface for control placement. You can use drag-and-drop editing to create the visual design of your application. You can then manage the logical design of your application by modifying the default Web control code.



The screen options for the editor are Design mode and HTML mode:

- Design mode

In Design mode, the editor allows you to move controls and graphic elements around the window by a simple drag-and-drop operation. Visual Studio .NET provides two control positioning schemes for designing Web pages: **FlowLayout** and **GridLayout**. In **FlowLayout**, controls follow each other across the page, while **GridLayout** allows you to exactly position each control by automatically adding dynamic Hypertext Markup Language (DHTML) tags to the controls.

When you add a control to a Web page in **Design** mode, Visual Studio .NET adds the supporting code and default properties to the Web Form. You can then switch to **HTML** mode to bring up that code for you to edit.

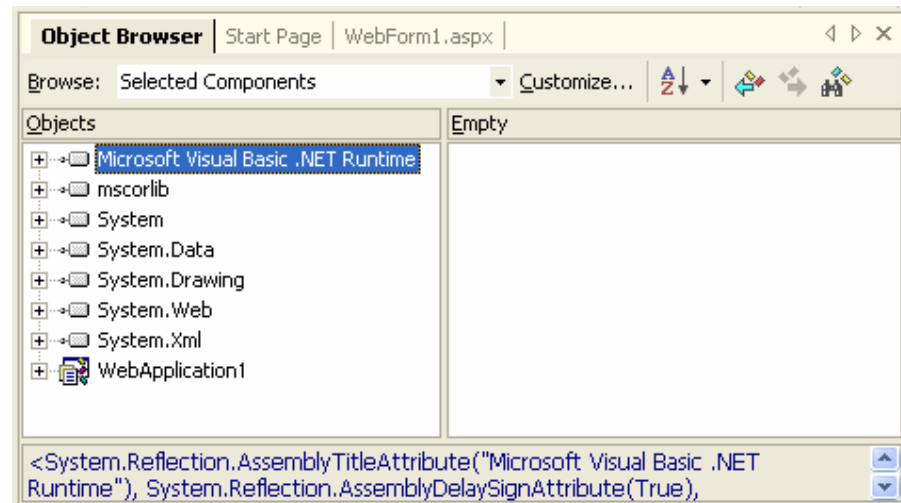
- HTML mode

In HTML mode, Visual Studio .NET highlights your code so that the different elements, such as variable names and key words, are instantly identifiable. The IntelliSense feature provides you with auto completion suggestions and allows you to build functions by simply selecting from lists of available syntax.

When you use the editor window in HTML mode, two drop-down lists appear at the top of the window: the **Class Name** list, which is on the left, and the **Method Name** list, which is on the right. The **Class Name** list shows all of the controls on the associated form. If you click a control name in the list, the **Method Name** list then shows all of the events for that control. Events are actions that the control can perform and that can be interpreted by your application. By using the **Class Name** and **Method Name** lists together, you can quickly locate and edit the code in your application.

Object Browser

The Object Browser is a tool that provides information about objects and their methods, properties, events, and constants.

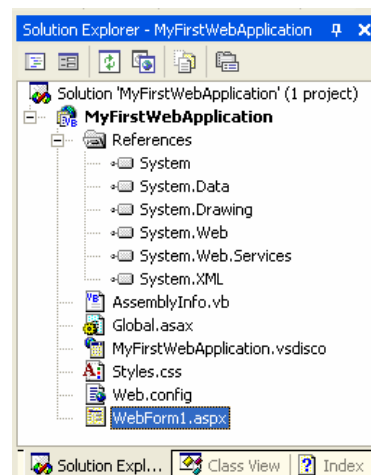
**Solution Explorer**

Solution Explorer displays the hierarchy of project files. From this window, you can move and modify files, including:

- Use a drag-and-drop operation to rearrange items.
- Select an item in Solution Explorer and the Properties window will show that item's properties. This allows you to change properties at the project or page level.
- Right-click the file, project, or solution to see the available options, including adding, building, and editing pages.

The file types shown in Solution Explorer include:

- Project References that list the classes that are used by the page and Web controls.
- All of the Web Forms in the project.
- All of the code-behind pages that contain the logic that supports the Web Forms.
- Project-related folders and sub-items.



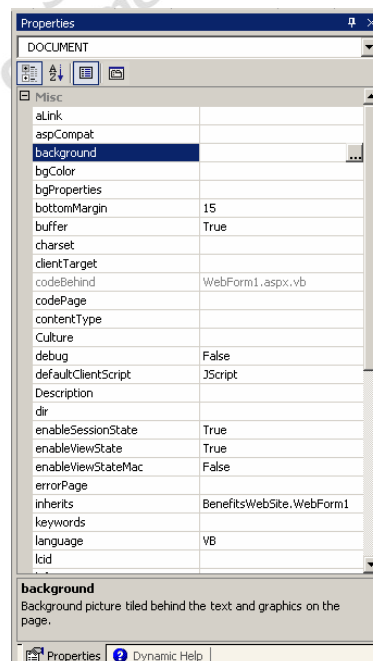
Dynamic Help

Dynamic help provides access to local and online help topics, based on the settings in **My Profile**, the **Project Type**, and the present location of the cursor. As you move around the IDE or edit code, the available options in dynamic help adjust to match your activity.



Properties

Visual Studio .NET lets you adjust the properties of documents, classes, and controls by using a common Properties window. When you create or select an item, the Properties window automatically displays the related properties. As shown on the following illustration, the available properties are listed in the left column, while the settings are listed on the right.



Task List

The Task List allows you to track the status of tasks as you develop applications. Visual Studio .NET also uses the Task List to flag errors when you build your application.

There are a number of ways to add a task to the Task List, including:

- Adding tasks manually by clicking the Task List and entering items.
The top task in the following Task List screen shot is a manually added task.
- Visual Studio .NET automatically adds a task for tokens, such as the 'TODO' comment in the code.

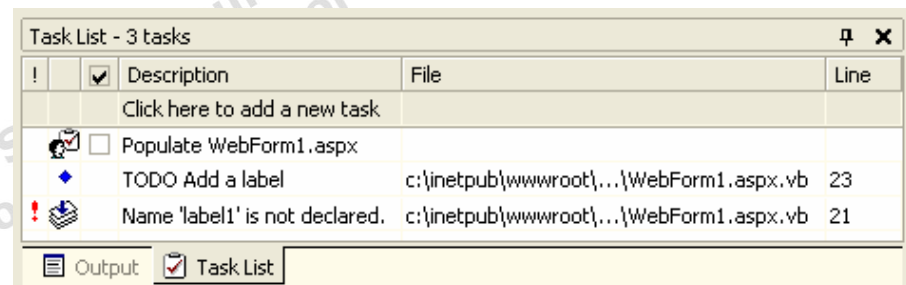
The second task in the following screen shot was automatically added by Visual Studio .NET because of a 'TODO' comment in the code. To access this section of code, click the item in the Task List and Visual Studio .NET will open the preferred page at that comment line.

There are a number of preset tokens that you can use in your code and they will automatically add a task to the Task List.

► To view and add to the list of tokens

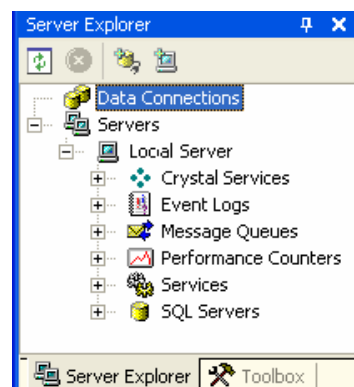
- a. On the **Tools** menu, click **Options**.
 - b. In the **Options** dialog box, on the **Environment** folder, click **Task List**.
- Visual Studio .NET automatically adds build errors to the Task List.

The bottom task in the following screen shot was added automatically when the page was built. To access this section of code, click the item in the task list and Visual Studio .NET will then open the preferred page at the line containing the error.



Server Explorer

Server Explorer allows you to view local data connections, servers, and windows services. Server Explorer supports the integration of external services into your Web site.



Toolbox

The Toolbox allows you to use a drag-and-drop operation on the controls in your application.

The available tools are grouped by category, in the following menus:

- **Data**

This category contains objects that allow your application to connect and access the data in a Microsoft SQL Server™ and other databases.

- **Web Forms**

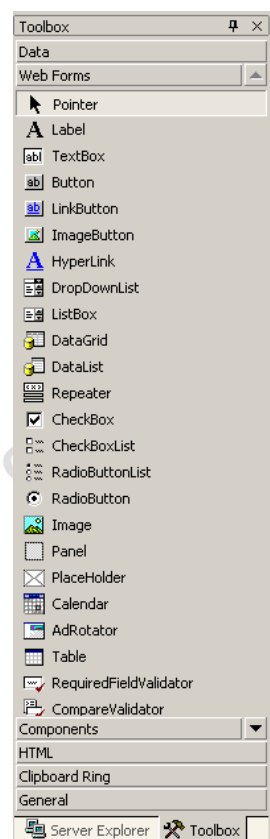
This category contains a set of server controls that you can add to Web pages.

- **Components**

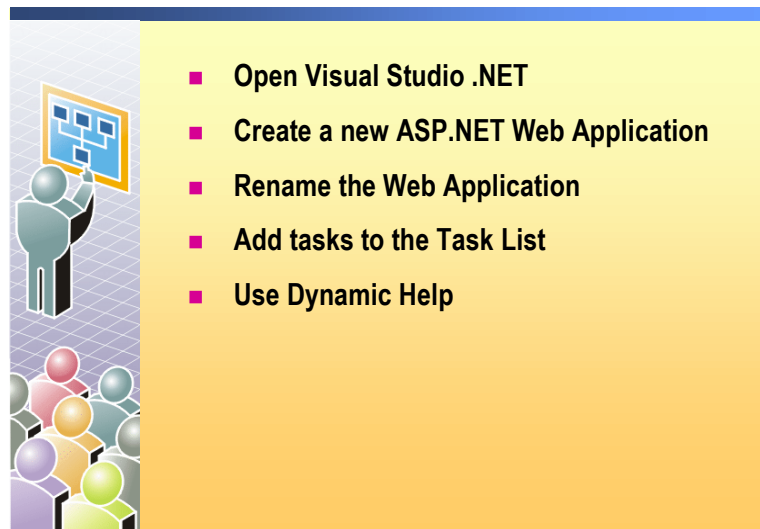
This category contains components that support the infrastructure of your application.

- **HTML**

This category contains a set of HTML controls that you can add to your Web page. These controls can run on either the server side or the client side.



Demonstration: Using the Visual Studio .NET IDE



In this demonstration, you will learn how to use the Visual Studio .NET IDE to create an ASP.NET Web application project, add tasks to the Task List, use Dynamic Help, and use Server Explorer.

► To run this demonstration

Create a Visual Studio .NET solution

1. Open Visual Studio .NET.
2. Show the features of the Start Page, such as **What's New**, **Search Online**, and **My Profile**.
3. Using Visual Studio .NET, create a new blank solution named **MyFirstSolution**:
 - a. On the **File** menu, point to **New**, and then click **Blank Solution**.
 - b. In the **New Project** dialog box, type **MyFirstSolution** in the **Name** text box, and then click **OK**.

The **MyFirstSolution** solution will contain several projects. The solution serves as a container to group these related projects.

Create an ASP.NET Web Application project within the solution

4. Create a new ASP.NET Web Application project named **MyFirstWebApplication** in the **MyFirstSolution** solution:
 - a. On the **File** menu, point to **New**, and then click **Project**.
 - b. In the **New Project** dialog box:

Visual Basic .NET

In the **Project Types** list, click **Visual Basic Projects**. In the **Templates** list, click **ASP.NET Web Application**, set the **Location** to **http://localhost/MyFirstWebApplicationVB**, click **Add to Solution**, and then click **OK**.

C#

In the **Project Types** list, click **Visual C# Projects**. In the **Templates** list, click **ASP.NET Web Application**, set the **Location** to **http://localhost/MyFirstWebApplicationCS**, click **Add to Solution**, and then click **OK**.

View where the new files are located

5. Using Solution Explorer, show the files that were created when the Web application was created.
6. Using Windows Explorer, show where the files were created in the file system.

The solution files are located in the

\My Documents\Visual Studio Projects\MyFirstSolution folder.

The Visual Basic .NET project files are located in the

\Inetpub\wwwroot\MyFirstWebApplicationVB folder and the C# project files are located in the **\Inetpub\wwwroot\MyFirstWebApplicationCS** folder.

View the properties of the Web Form

7. In Visual Studio .NET Solution Explorer, select **WebForm1.aspx** and show the properties that are listed in the Properties window.
8. Using the Toolbox, add a **Label** control and a **Button** control to the Web Form.

Notice that the Properties window now shows properties for the **Button** control.

Dynamic Help

9. On the **Help** menu, click **Dynamic Help**.

The Dynamic Help window opens with topics about the **Button** control.

10. Click the **Label** control.

The Dynamic Help window displays topics about the **Label** control.

11. Click the **Label Members (System.Web.UI.WebControls)** topic.

The topic is displayed in the main Visual Studio .NET window.

Show the **Text** and the **Visible** properties.

Server Explorer

12. In Server Explorer, expand **Servers**, expand *machinename*, expand **SQL Servers**, and then expand *machinename*.

The list of databases installed on the local computer running SQL Server is displayed.

13. For instance, open the **dentists** table of the **dentists** database.

Task List

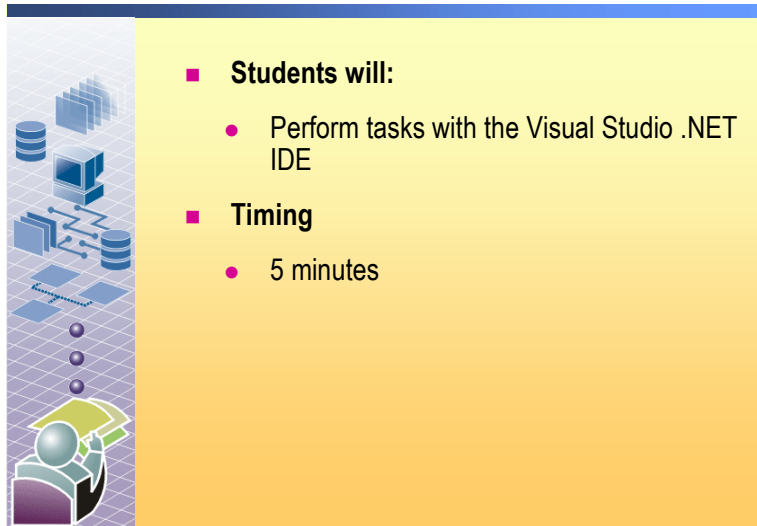
14. If the Task List is not visible, on the **View** menu, click **Show Tasks**, and then click **All** to display the Task List window.

15. Insert the following tasks:

- Add a new ASP.NET Web Form
- Add a new project to the solution

16. On the **File** menu, click **Save All**.

Practice: Using the Visual Studio .NET IDE



- **Students will:**
 - Perform tasks with the Visual Studio .NET IDE
- **Timing**
 - 5 minutes

► **Complete the following tasks and list which window(s) you used**

Create a new Web application project.

Start Page

Add a **Button** control to the default Web Form.

Toolbox

Add a task to the project.

Task List

View the properties of the Web application project.

Properties

Determine what SQL Server databases are installed on your computer.

Server Explorer

Sample Courseware
Not for Commercial Use or Redistribution

Lesson: Creating an ASP.NET Web Application Project

- The Development Process
- Web Application Files
- Web Application File Structure
- Demonstration: Creating a Web Application Project

Introduction

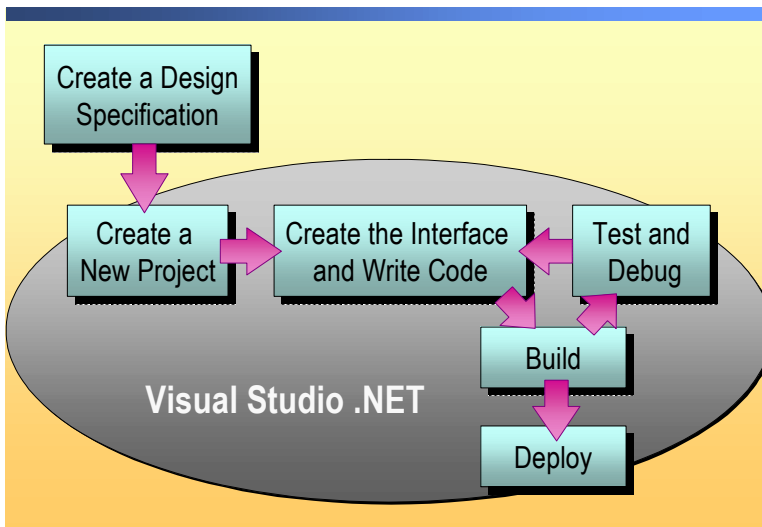
In this lesson, you will learn how to create, build, and view an ASP.NET Web application.

Lesson objectives

After completing this lesson, you will be able to:

- Explain how ASP.NET Web application pages are developed.
- Identify the function of the files that are used by Visual Studio .NET solutions.
- Explain the build process for ASP.NET Web application.
- Create, build, and view an ASP.NET Web application.

The Development Process



Introduction

Visual Studio .NET contains everything you need to build your own ASP.NET Web application from start to finish.

Creating an ASP.NET Web application with Visual Studio .NET involves the following basic steps:

1. Create a design specification

The design specification is the blueprint that you will use when you create a Web application. Take time before writing any code to design the application that you will be creating. Although Visual Studio .NET provides tools to help you quickly develop a solution, having a clear understanding of the user needs and initial feature set will help you to be more efficient in your development efforts. By first coming up with a design specification, it will also help you save time by minimizing the potential for rewriting code because of a poor or nonexistent design specification.

2. Create a new project

When you select a new project template, Visual Studio .NET automatically creates the files and the default code that are needed to support the project.

As part of this initial project creation, you should transfer the main coding tasks from your design specification into the Visual Studio .NET Task List. This transfer allows you to track your development against the specification.

3. Create the interface

To create the interface for your Web application, you will first need to place controls and objects on the Web pages by using the Editor/Browser window in Design mode.

As you add objects to a form, you can set their properties from the table in the Properties window or as code in the Editor window.

Note For more information on adding controls to an ASP.NET Web Form, see Module 4, “Creating a Microsoft ASP.NET Web Form,” in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*.

4. Write code

After you have set the initial properties for the ASP.NET Web Form and its objects, you can write the event procedures that will run when different actions are performed on a control or object.

You may also need to write code to add business logic and to access data.

Note For more information on writing code in ASP.NET Web Forms, see Module 5, “Adding Code to a Microsoft ASP.NET Web Form,” in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*.

5. Build

When building a project, you compile all of the code in the Web pages and other class files into a dynamic-link library (DLL) called an *assembly*.

Visual Studio .NET has two build options: debug and release. When you are first developing a project, you will build debug versions. When you are ready to release the project, you will create a release build of the project.

6. Test and debug

Testing and debugging is not a one-time step, but rather something that you do iteratively throughout the development process. Each time you make a major change, you will need to run a debug build of the application to ensure that it is working as expected.

Visual Studio .NET offers numerous debugging tools that you can use to find and fix errors in your application.

Note For more information on debugging, see Module 6, “Tracing in Microsoft ASP.NET Web Applications,” in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*.

7. Deploy

When a project is fully debugged and a release build has been built, you can deploy the necessary files to a production Web server.

Note For more information on deploying an ASP.NET Web application, see Module 15, “Configuring, Optimizing, and Deploying a Microsoft ASP.NET Web Application,” in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*.

Web Application Files

- **Solution files (.sln, .suo)**
- **Project files (.vbproj, .csproj)**
- **Web application files**
 - ASP.NET Web Forms (.aspx)
 - ASP.NET Web services (.asmx)
 - Classes, code-behind pages (.vb or .cs)
 - Global application classes (.asax)
 - Web.config file
- **Project assembly (.dll)**

Introduction

When you create a new project or work with existing projects, Visual Studio .NET creates a number of files that support your development.

Solution files

When you create a new project, a solution is also created, even if you only have one project in the solution. A folder is created for each solution in the **\My Documents\Visual Studio Projects** folder that contains the .sln and .suo files.

■ Solution files (.sln)

The *SolutionName.sln* file extension is used for solution files that link one or more projects together, and it stores certain global information. .sln files are similar to Visual Basic group (.vbg) files, which appear in previous versions of Visual Basic.

■ Solution User Options (.suo)

The *SolutionName.suo* file extension is used for Solution User Options files that accompany any solution records and customizations that you add to your solution. This file saves your settings, such as breakpoints and task items, so that they are retrieved each time you open the solution.

Project files

Each project is a single Web application stored in its own folder. Inside the project folder are the project configuration file and the actual files that make up the project. The Project Configuration file is an Extensible Markup Language (XML) document that contains references to all of the project items, such as forms and classes, in addition to project references and compilation options.

Visual Basic .NET project files use a .vbproj extension, while C# uses .csproj. These extensions enable you to differentiate between files that are written in other .NET-compatible languages and make it easy to include multiple projects that are based on different languages within the same solution.

Web application projects are created in a new folder in the **\Inetpub\wwwroot** folder. In addition, a virtual directory that points to the project folder is created in IIS.

Web application files

Visual Studio .NET supports a number of application file types and extensions:

- ASP.NET Web Forms (.aspx)

ASP.NET Web Forms are used when you need to build dynamic Web sites that will be accessed directly by users.

ASP.NET Web Forms may be supported by a code-behind page that is designated by the extension *WebForm.aspx.vb* or *WebForm.aspx.cs*.

- ASP.NET Web services (.asmx)

Web services are used when you want to create dynamic Web sites that will only be accessed by other programs.

ASP.NET Web services may be supported by a code-behind page that is designated by the extension *WebService.asmx.vb* or *WebService.asmx.cs*.

- Classes and code-behind pages (.vb or .cs)

Previous versions of Visual Basic used different file extensions to distinguish between classes (.cls), forms (.frm), modules (.bas), and user controls (.ctl). Visual Basic .NET allows you to mix multiple types within a single .vb file.

Code-behind pages carry two extensions, the page type (.aspx or .asmx) and the Visual Basic extension (.vb) or the C# extension (.cs). For example, the full file name for the code-behind page for a default ASP.NET Web Form is *WebForm1.aspx.vb* for a Visual Basic .NET project, and for a C# project it is *WebForm1.aspx.cs*.

Note You will learn more about code-behind pages in Module 5, “Adding Code to a Microsoft ASP.NET Form,” in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*.

- Discovery files (.disco and .vsdisco)

Discovery files are XML-based files that contain links (URLs) to resources that provide discovery information for an XML Web service. These files enable programmatic discovery of XML Web services.

- Global application classes (global.asax)

The Global.asax file, also known as the ASP.NET application file, is an optional file that contains code for responding to application-level events that are raised by ASP.NET or raised by HttpModules. At runtime, Global.asax is parsed and compiled into a dynamically generated .NET Framework class that is derived from the **HttpApplication** base class.

- Resource files (.resx)

A resource is any non-executable data that is logically deployed with an application. A resource might be displayed in an application as error messages or as part of the UI. Resources can contain data in a number of forms, including strings, images, and persisted objects. Storing your data in a resource file allows you to change the data without recompiling your entire application.

- Styles.css

Styles.css is the default stylesheet file for the Web application.

- Web.config file

This Web.config file contains configuration settings that the common language runtime reads, such as assembly binding policy, remoting objects, and so on, and settings that the application can read. Web.config files also contain the global application classes that support a project.

Other files

Any files that are not based on a programming language will have their own extensions. For example, a Crystal Report file uses the .rpt extension, and a text files uses .txt.

Project assembly

When a Web project is compiled, two additional types of files are created:

- Project Assembly files (.dll)

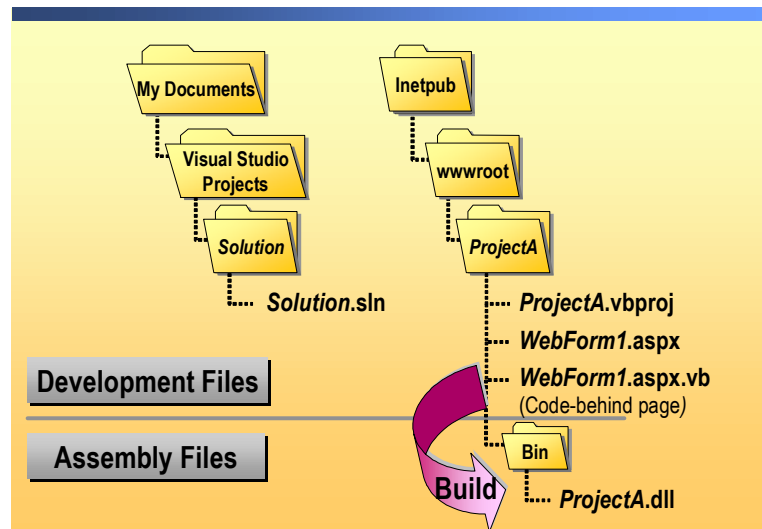
All of the code-behind pages (.aspx.vb and .aspx.cs) in a project are compiled into a single assembly file that is stored as *ProjectName.dll*. This project assembly file is placed in the /bin directory of the Web site.

- AssemblyInfo.vb or AssemblyInfo.cs

The AssemblyInfo file is used to write the general information, specifically assembly version and assembly attributes, about the assembly.

Note For more information on files that support ASP.NET Web applications, see the Visual Studio .NET documentation.

Web Application File Structure



Introduction

When you create an ASP.NET Web application, Visual Studio .NET creates two folders to store the files that support that application. When you compile a project, a third folder is created to store the resulting .dll file.

My Documents

Visual Studio .NET creates a folder for the solution, *ProjectA*, containing the file *ProjectA.sln*. This file is a map of all of the various files that support the project.

You can also create a blank solution and then add projects to it. By creating a blank solution, you will have a solution that has a different name than the project.

Inetpub

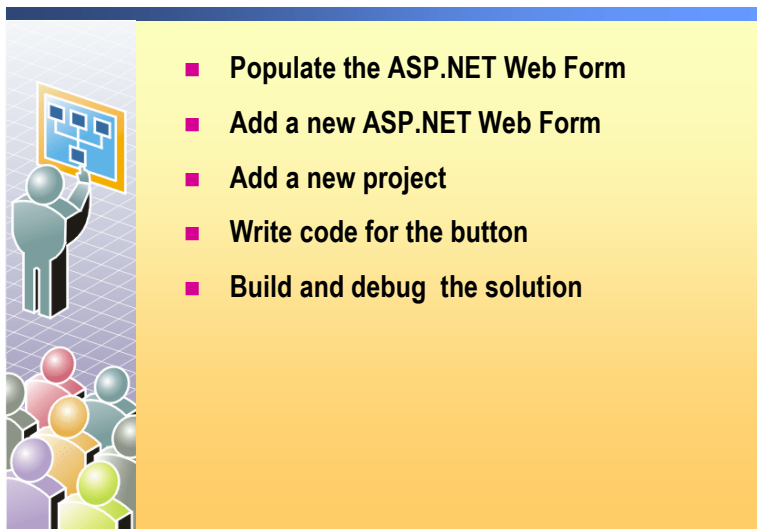
Visual Studio .NET also creates a folder named *ProjectA*, in the *Inetpub\wwwroot* folder, that contains the files that are required by the Web application. These files include:

- The project file, *ProjectA.vbproj* or *ProjectA.csproj*, which is an XML document that contains references to all project items, such as forms and classes, in addition to project references and compilation options.
- ASP.NET Web Forms, *WebForm1.aspx*, or XML Web services, *WebService1.aspx*.
- Code-behind pages, *WebForm1.aspx.vb*, *WebService1.aspx.vb*, *WebForm1.aspx.cs* or *WebService1.aspx.cs*.
- A Web.config file, which contains the configuration setting for the Web application.
- A Global.asax file that handles events that are fired while the Web application is running.

The Assembly

When you build a Web application project, Visual Studio .NET creates an assembly in the *Inetpub\wwwroot\ProjectA\bin* folder. An assembly is one .dll file that is created from all of the code-behind pages that make up a Web application.

Demonstration: Creating a Web Application Project



In this demonstration, you will learn how to add Web Forms to a Web Application project, add a new project to a solution, and build and run a Web Application project.

► To run this demonstration

1. Open the MyFirstSolution solution file.
2. Open the WebForm1.aspx file in Design view.
3. There are already two controls, a **Button** control and a **Label** control, that you placed in the previous demonstration.
4. Select the label and change the **ID** property to **lblMessage** in the Properties window.
5. Double-click the **Button** control to open the code-behind page for the Web Form, and add the following comment to the event procedure:

Populate an ASP.NET Web Form

Visual Basic .NET

```
'TODO: Write Hello World
```

C#

```
//TODO: Write Hello World
```

A new task is automatically added to the Task List because of the TODO token.

Add a new ASP.NET Web Form

6. On the **Project** menu, click **Add Web Form**.
7. In the **Add New Item** dialog box, change the default name to login.aspx, and click **Open**.

Note You can also add an ASP.NET Web Form to a project by right-clicking the project in Solution Explorer, clicking **Add**, and then clicking **Add Web Form**.

8. In the Task List, select the **Add a new ASP.NET Web Form** task check.

- Add a new project**
9. In Solution Explorer, right-click the solution, click **Add**, and then click **New Project**.
10. In the **Add New Project** dialog box:
- Visual Basic .NET** Click **Visual Basic**, click **ASP.NET Web Service**, set the location to **http://localhost/MyFirstWebServiceVB**, and then click **OK**.
- C#** Click **Visual C#**, click **ASP.NET Web Service**, set the location to **http://localhost/MyFirstWebServiceCS**, and then click **OK**.
- The new project also contains a XML Web service.
11. In the Task List, check the **Add a new project to the solution** task.
- Write code for a button control**
12. In the Task List, double-click the **TODO: Write Hello World** task.
- The correct file is opened and the cursor is placed at the correct spot in the code.
13. Write the following code:
- Visual Basic .NET** `lblMessage.Text = "Hello World!"`
- C#** `lblMessage.Text = "Hello World!";`
-
- Note** This is a syntax error because the code sets the **Txt** property instead of the **Text** property. This error is there to show you what happens when the build fails.
-
14. Remove the **TODO** from the **TODO: Write Hello World** comment.
- The **TODO** task disappears automatically from the Task List.
- Build and run the solution**
15. Verify that **MyFirstWebApplicationVB** or **MyFirstWebApplicationCS** is the startup project. In Solution Explorer, **MyFirstWebApplicationVB** or **MyFirstWebApplicationCS** should appear in bold text.
-
- Note** To set **MyFirstWebApplicationVB** or **MyFirstWebApplicationCS** as the startup project, right-click the **MyFirstWebApplicationVB** or **MyFirstWebApplicationCS** project in Solution Explorer and click **Set as StartUp Project**.
-
16. On the **Build** menu, click **Build Solution** to build the solution.
- Both **MyFirstWebApplicationVB** or **MyFirstWebApplicationCS** and **MyFirstWebServiceVB** or **MyFirstWebServiceCS** are built, but the Output window displays the following error condition:
- Build: 1 succeeded, 1 failed, 0 skipped
- And an error message is added to the Task List because **Txt** is not a member of **System.Web.UI.WebControls.Label**.
17. To view all of the tasks from both projects, on the **View** menu, click **Show Tasks**, and then click **All**.

18. Double-click the error message in the Task List. The cursor jumps to the correct spot in the code.

19. Correct the syntax error.

The correct code should look like the following:

Visual Basic .NET

```
lblMessage.Text = "Hello World!"
```

C#

```
lblMessage.Text = "Hello World!";
```

For Visual Basic .NET, when you move the cursor off of the corrected line of code, the error is removed from the Task List.

20. On the **Build** menu, click **Rebuild Solution** to rebuild the solution.

Verify that you have the following message in the Output window:

```
Rebuild All: 2 succeeded, 0 failed, 0 skipped
```

View the result

21. To view the Web page in a browser, right-click **WebForm1.aspx** in the **MyFirstWebApplicationVB** or **MyFirstWebApplicationCS** project in Solution Explorer, and then click **View in Browser**.

Note You can build and browse a Web Form in one step by right-clicking the page in Solution Explorer and then clicking **Build and Browse**.

22. In the browser, click the button on the Web Form, and make sure that you see the message **Hello World**.

Sample Courseware
Not for Commercial Use or Redistribution

Review

- Overview of Visual Studio .NET
- Creating an ASP.NET Web Application Project

-
1. What is the difference between a Visual Studio .NET solution and a Visual Studio .NET project?

The solution is just a development tool for organizing projects. The project is the actual Web application.

2. What is the difference between Server Explorer and Solution Explorer in Visual Studio .NET?

Server Explorer lists server resources. Solution Explorer lists the projects in the open solution and the files in the projects.

3. How do you add a new Web Form to a project?

You can either click Add Web Form on the Project menu, or right-click the project in Solution Explorer, click Add, and then click Add Web Form.

4. When you create a Web application project, where are the project files stored?

In the \Inetpub\wwwroot\projectname folder.

5. When you build a project, what file(s) is created?

The assembly DLL for the project is created and stored in the \bin folder of the project.

6. How do you view a Web Form in the Visual Studio .NET browser?

Before you can view a Web Form, you must build the project. You can either right-click the Web Form and click Build and Browse, or click Build Project on the Build menu, then right-click the Web Form and click View in Browser.

Sample Courseware
Not for Commercial Use or Redistribution

The diagram illustrates the architecture of the Lab Web Application. It is divided into two main sections by a dashed blue line: the **Lab Web Application** (top) and the **XML Web Service** (bottom).

Lab Web Application Components:

- Logon Page** (Login.aspx) and **Registration** (Register.aspx) are connected to the **Coho Winery** database.
- Logon Page** and **Registration** both point to the **Benefits Home Page** (Default.aspx).
- Benefits Home Page** is connected to the **Menu Component** (Class1.vb or Class1.cs).
- Page Header** (Header.aspx) is connected to the **Menu Component**.
- Menu Component** is connected to the **Web.config** file.
- Benefits Home Page** points to four sub-modules: **Life Insurance** (Life.aspx), **Retirement** (Retirement.aspx), **Medical** (Medical.aspx), and **Dentists** (Dental.aspx).
- Life Insurance**, **Retirement**, **Medical**, and **Dentists** are all connected to the **XML Web Service**.
- Retirement** is connected to the **Prospectus** (Prospectus.aspx).
- Medical** is connected to the **Doctors** (Doctors.aspx).
- Dentists** is connected to the **User Control** (namedate.aspx).
- Prospectus**, **Doctors**, and **User Control** are all connected to the **XML Web Service**.
- XML Web Service** (dentalService1.asmx) is connected to the **Dentists** database.

Data Sources:

- Coho Winery** (Database)
- ASPState** (Session State)
- tempdb** (Temporary Database)
- XML Files** (Files)
- Doctors** (Database)
- Dentists** (Database)

After completing this lab, you will be able to:

- Create a Microsoft® Visual Studio® .NET solution.
- Create a Web Application project.
- Add files to a Web Application project.

Prerequisites

There are no prerequisites for this lab.

Coho Winery offers several benefits to its employees. In the labs for Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*, you will create a Web site that enables employees to select and set up their chosen benefits.

In this lab, you will create a Microsoft ASP.NET Web Application project for the Web site. You can work through the labs by using either C# or Microsoft Visual Basic® .NET.

Estimated time to complete this lab: 15 minutes

Exercise 1

Creating an ASP.NET Web Application Project

In this exercise, you will create the ASP.NET Web Application project that will be used for the Benefits Web site that is built throughout the labs that comprise Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*.

► **Create the ASP.NET Web application and add the starter files**

1. Using Visual Studio .NET, create a new blank solution named **2310LabApplication**:
 - a. On the **File** menu, point to **New**, and then click **Blank Solution**.
 - b. In the **New Project** dialog box, type **2310LabApplication** in the **Name** text box, and then click **OK**.

The 2310LabApplication solution contains several projects. Each project represents a different application or component. The solution serves as a container to group these related projects.

2. Create a new ASP.NET Web Application project named **BenefitsVB** or **BenefitsCS** in the 2310LabApplication solution:
 - a. On the **File** menu, point to **New**, and then click **Project**.
 - b. In the **New Project** dialog box, in the **Project Types** list, click **Visual Basic Projects** or **Visual C# Projects**.
 - c. In the **Templates** list, click **ASP.NET Web Application**.
 - d. Set the **Location** to **http://localhost/BenefitsVB** for a **Visual Basic .NET** project, or to **http://localhost/BenefitsCS** for a C# project.
 - e. Click **Add to Solution** and then click **OK**.

Caution When adding projects to the solution, the capitalization of the project name is important. Because you may be using some prebuilt Web Forms in this and other labs in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*, you must verify that you have capitalized the Benefits project as shown.

Visual Studio .NET creates a virtual root named **BenefitsVB** or **BenefitsCS**. List the 6 files for a Visual Basic project or 5 files for a C# project that appear in Solution Explorer.

For a Visual Basic .NET project:

AssemblyInfo.vb, BenefitsVB.vsdico, Global.asax, Styles.css, Web.config, WebForm1.aspx

For a C# project:

AssemblyInfo.cs, BenefitsCS.vsdico, Global.asax, Web.config, WebForm1.aspx

The Benefits project will be the main Web application that you will build throughout the labs in Course 2310B, *Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET*. Two versions of the project can be created: **BenefitsVB** is a **Visual Basic .NET** solution and **BenefitsCS** is a **C#** solution.

3. Add the starter lab files to the Benefits project:
 - a. In Solution Explorer, right-click **BenefitsVB** or **BenefitsCS**, point to **Add**, and then click **Add Existing Item**.
 - b. For the Visual Basic .NET Project, browse to the *install folder*\Labfiles\Lab02\VB\Starter\BenefitsVB folder for the Visual Basic .NET files.
For the Visual C# project, browse to the *install folder*\Labfiles\Lab02\CS\Starter\BenefitsCS folder for the C# files.

Note The default installation folder for this class is
C:\Program Files\Msdntrain\2310.

- c. In the **Files of type** box of the **Add Existing Item – Benefits** dialog box, select **All Files (*.*)**.
 - d. Select all of the files in this folder, and then click **Open**.
 - e. Click **Yes** if asked if you want to create a new class file for the medical.aspx Web Form.

When you add files to the Benefits project, they become part of the Benefits Web application. The files that you copied in the preceding step provide a foundation upon which you will build your Web application.

4. Right-click the **WebForm1.aspx** page in Solution Explorer and then click **Build and Browse**.

When you build and browse a project, the code files are compiled and the default Web Form is displayed in the built-in browser in Visual Studio .NET.

At this stage, WebForm1.aspx is blank. The browser displays a blank Web page.

5. Each project in Visual Studio .NET has its own virtual directory in Internet Information Services (IIS). Using Windows Explorer, find where the files were created for the Benefits project. Fill in your answer below.

\Inetpub\wwwroot\BenefitsVB

-or-

\Inetpub\wwwroot\BenefitsCS

Exercise 2

Using Visual Studio .NET

In this exercise, you will set up a profile, configure the Benefits project, and then add tasks to the Task List.

► Set attributes for a profile

1. Click the **Start Page** tab to view the Start page of Visual Studio .NET.
If the **Start Page** tab is not available, on the **Help** menu, click **Show Start Page**.
2. Click **My Profile** to open the profile attribute section.
3. In the **Help Filter** drop-down list, click **.NET Framework SDK**.

Setting a profile allows Visual Studio .NET to customize the interface for the type of development you are doing.

► Set properties for the Visual Basic .NET project

1. In Solution Explorer, right-click the **BenefitsVB** project, and then click **Properties**.
2. In the **BenefitsVB Property Pages** dialog box, in the Common Properties folder, click **Build**.
3. In the **Option Strict** drop-down list, click **On**, and then click **OK** to apply the changes.

Enabling Option Strict causes Visual Basic .NET to enforce stronger data type conversion rules, and helps to ensure a well-running Web application.

► Set properties for the Visual C# project

- Option Strict does not need to be set for a Microsoft Visual C# project because C# enforces stronger type conversion rules by definition.

► Add tasks to the Task List

1. On the **View** menu, point to **Show Tasks**, and then click **All** to display the Task List.
2. On the Task List, click the area labeled **Click here to add a new task**. In the **Description** field, type **Build the BenefitsList component** and then press ENTER.
3. Create a second task named **Build the Benefits Web application** and then press ENTER.

The Task List is a convenient place to list the tasks that you have yet to accomplish.

4. On the **File** menu, click **Exit**.
5. Click **Yes** to save changes.

