

TRAINING MATERIALS FOR IT PROFESSIONALS

ASP.NET Using C#



EVALUATION COPY
Unauthorized reproduction or distribution is prohibited

ASP.NET Using C#

Student Guide

Revision 4.8

ASP.NET Using C#

Rev. 4.8

Student Guide

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Object Innovations.

Product and company names mentioned herein are the trademarks or registered trademarks of their respective owners.



TM is a trademark of Object Innovations.

Authors: Robert J. Oberg, Arun Ganesh, Srinivas Manickam

Special Thanks: Mark Bueno, Peter Thorsteinson, Sharman Staples, Ernani Cecon, Dave Solum, Daniel Berglund

Copyright ©2018 Object Innovations Enterprises, LLC All rights reserved.

Object Innovations
877-558-7246
www.objectinnovations.com

Printed in the United States of America.

Table of Contents (Overview)

Chapter 1	Introduction to ASP.NET
Chapter 2	Web Forms Architecture
Chapter 3	ASP.NET and HTTP
Chapter 4	Web Applications Using Visual Studio
Chapter 5	State Management and Web Applications
Chapter 6	Server Controls
Chapter 7	Caching in ASP.NET
Chapter 8	ASP.NET Configuration and Security Fundamentals
Chapter 9	Debugging, Diagnostics and Error Handling
Chapter 10	More Server Controls
Chapter 11	ADO.NET and LINQ
Chapter 12	Data Controls and Data Binding
Chapter 13	ASP.NET AJAX
Chapter 14	ASP.NET MVC
Chapter 15	ASP.NET Web API
Chapter 16	ASP.NET and Azure
Appendix A	Learning Resources
Appendix B	Hosting in IIS 7.5

Directory Structure

- **The course software installs to the root directory C:\OIC\AspCs.**
 - Example programs for each chapter are in named subdirectories of chapter directories **Chap01**, **Chap02**, and so on.
 - The **Labs** directory contains one subdirectory for each lab, named after the lab number. Starter code is frequently supplied, and answers are provided in the chapter directories.
 - The **CaseStudy** directory contains case studies in multiple steps.
 - The **Demos** directory is provided for doing in-class demonstrations led by the instructor.
- **Data files install to the directory C:\OIC\Data.**
- **Log files are written to the directory C:\OIC\Logs.**

Table of Contents (Detailed)

Chapter 1 Introduction to ASP.NET	1
Web Application Fundamentals.....	3
Setting up the Web Examples	4
Benefits of ASP.NET	8
ASP.NET Example Program	9
An Echo Program.....	10
ASP.NET Features	13
Compiled Code	15
Server Controls	16
Browser Independence.....	17
Separation of Code and Content	18
State Management.....	19
ASP.NET Core	20
Lab 1	21
Summary	22
Chapter 2 Web Forms Architecture	25
Web Forms Architecture.....	27
Code-Behind Version of Echo Example.....	28
HelloCodebehind.aspx	29
HelloCodebehind.aspx.cs.....	30
Page Class	31
Code-Behind Inheritance Model.....	32
Web Forms Page Life Cycle	33
View State.....	36
Enabling View State for Controls	37
Web Forms Event Model	38
Page Processing	40
Page Events.....	41
Page Properties	42
Sample Program.....	43
Page Directive	47
Tracing	49
Lab 2	51
Summary	52
Chapter 3 ASP.NET and HTTP	55
Classical Web Programming.....	57
Active Server Pages Object Model	58
Request and Response Objects	59
Request/Response in ASP.NET	60
HttpRequest Class	61
Properties of HttpRequest	62

Using HttpRequest Class	63
HTTP Collections	64
HttpResponse Class	66
Redirect	67
HttpUtility	68
Echo Program Example	69
Echo.aspx	70
EchoBack.aspx	71
GET and POST Compared	72
QueryString and Form Collections	73
Lab 3	74
Summary	75
Chapter 4 Web Applications Using Visual Studio.....	79
Using Visual Studio	81
Visual Studio Web Demo	82
Starter Web Site	83
ASP.NET Empty Web Site	85
Adding a Web Form	86
Default.aspx	87
Adding Controls	88
Setting Properties	89
Using Components in ASP.NET	92
Running Under IIS	93
Shadow Copying	97
Shadow Copy Demonstration	98
Temporary Copy of the Component	100
ASP.NET Applications	102
Global.asax	103
Web Application Life Cycle	104
Application Life Cycle Example	105
Sample Log File	106
Code in Global.asax	107
Log Class	108
Adding Global.asax File	109
ListBox	110
Data Binding	111
Data Binding Code Example	112
Items Collection	113
Flow Positioning	114
Lab 4	115
Summary	116
Chapter 5 State Management and Web Applications	123
Session and Application State	126
Example Program	127

Session Object.....	128
Page_Load.....	129
Session Variable Issues.....	131
Session State and Cookies	132
Session State Timeout.....	133
Session State Store.....	134
Application State.....	135
Implementing Application State	136
Global.asax	137
Users.aspx.cs.....	138
Multithreading Issues.....	139
Bouncing the Web Server	140
Lab 5A	141
Cookies	142
Cookies and ASP.NET.....	143
HttpCookie Properties.....	144
Example – Exposing Cookies	145
Acme Travel Agency Case Study.....	151
State Management Techniques	153
Lab 5B.....	154
Summary	155
Chapter 6 Server Controls.....	169
Server Controls in ASP.NET	171
HTML Server Controls	172
Using HTML Server Controls	173
HTML vs. Web Forms Server Controls.....	174
Server Control Examples	175
HTML Controls Example	176
Code for Login.....	177
HTML Controls in Visual Studio	178
Using HTML Controls	179
Web Controls	180
Validation Controls.....	181
Required Field Validation.....	183
Regular Expression Validation	184
Rich Controls	185
Copying a Web Site	186
Lab 6	187
User Controls	188
Using a User Control	189
Copyright.ascx	190
Copyright.ascx.cs	191
User Control Example.....	192
Summary	193

Chapter 7 Caching in ASP.NET	197
Introduction.....	199
What is Caching?	200
Need for Caching (Why Cache?).....	201
Data to be Cached – Time Frame	202
ASP vs. ASP.NET Response Model.....	203
Caching in ASP.NET	204
Three Types of Caching in ASP.NET.....	205
Output Caching	206
@ OutputCache Directive	207
Simple Output Caching Example.....	208
@ OutputCache – Attributes in Detail.....	211
VaryByParam in Detail.....	212
HttpCachePolicy Class	214
HttpCachePolicy Class – Example	215
Page Fragment Caching	217
Common Mistakes in Using Fragment Caching	218
Fragment Caching Example.....	219
Data Caching or Application Caching	220
Add an Item to the Cache Object.....	221
Insert and Add Methods.....	222
Application Caching – Example	223
Expiration.....	225
Problems in Caching	226
Lab 7	227
Summary	228
Chapter 8 ASP.NET Configuration and Security Fundamentals	233
One-minute Introduction to XML!	235
ASP.NET Configuration - Overview	236
Multi-level Configuration	237
Configuration Hierarchy	238
Example of Configuration Hierarchy.....	239
Web.Config File Structure	240
Web.Config Sections	241
Application Settings.....	242
ASP.NET Security – Overview	243
Role-Based Security and CAS	244
Types and Steps	245
Steps in Enabling Role-Based Security	246
Two Ways to Authenticate.....	247
Forms Authentication Example	248
Forms Authentication – Default.aspx	250
Forms Authentication – Web.Config	253
Features of Forms Authentication.....	254
Authentication Cookie	255

Forms Authentication Classes	256
Customizing Forms Authentication	257
Authentication Source.....	258
Forms Authentication – Analysis	259
Windows Authentication	260
Windows Authentication – Analysis	261
Authorization	262
Lab 8	263
Summary	264
Chapter 9 Debugging, Diagnostics and Error Handling.....	271
ASP.NET Diagnostics.....	273
Debugging Using Visual Studio	274
Calculator Example.....	275
Debugging Calculator	276
Application-Level Tracing.....	278
Tracing Calculator	279
Using the Page Cache	282
Single Page Example	283
Preparing to Debug	284
Trace Messages	285
Tracing the Calculator Page.....	286
Conditional Tracing	287
Trace Category.....	288
Trace Warning	289
Exceptions in Trace	290
Errors in ASP.NET	291
Uncaught Exception.....	292
Custom Error Pages	293
Lab 9	294
Summary	295
Chapter 10 More Server Controls.....	297
ASP.NET Control Improvements	299
Newer Controls in ASP.NET	300
Master Pages	301
Master Page Demonstration.....	302
HTML 5 and Modernizr	308
Creating Content Pages.....	309
TreeView Control	310
Master Page Application.....	312
Lab 10	313
Summary	314
Chapter 11 ADO.NET and LINQ.....	319
ADO.NET	321
ADO.NET Architecture	322

.NET Data Providers	324
ADO.NET Interfaces	325
.NET Namespaces	326
Connected Data Access	327
SQL Express LocalDB	328
SqlLocalDB Utility	329
Visual Studio Server Explorer	330
Queries	332
ADO.NET with ASP.NET	333
Web Client Isolation	334
Web Client Database Code	335
Using Commands	337
Creating a Command Object	338
Using a Data Reader	339
Data Reader: Code Example	340
Use of Session State	341
Generic Collections	342
Executing Commands	343
Parameterized Queries	344
Parameterized Query Example	345
Lab 11A	346
DataSet	347
DataSet Architecture	348
Why DataSet?	349
DataSet Components	350
DataAdapter	351
DataSet Example Program	352
Data Access Class	353
Retrieving the Data	354
Filling a DataSet	355
Accessing a DataSet	356
Using a Standalone DataTable	357
DataTable Update Example	358
Adding a New Row	360
Searching and Updating a Row	361
Deleting a Row	362
Row Versions	363
Row State	364
Iterating Through DataRow	365
Command Builders	366
Updating a Database	367
Language Integrated Query (LINQ)	368
ADO.NET Entity Framework	369
LINQ and EDM Demo	370
IntelliSense	373

Basic LINQ Query Operators	374
Obtaining a Data Source	375
LINQ Query Example.....	376
Filtering.....	377
Ordering	378
Aggregation	379
Obtaining Lists and Arrays	380
Deferred Execution	381
Modifying a Data Source	382
LINQ to Entities Update Example.....	383
Entity Framework in a Class Library.....	384
Data Access Class Library	385
Client Code	386
Lab 11B.....	387
Summary	388
Chapter 12 Data Controls and Data Binding.....	401
Data Access in ASP.NET	403
Data Access Demonstration.....	404
Data Entry Demonstration.....	409
SQL Generation Options	410
Enable Edit and Delete	411
Editing Records.....	412
GridView Control	413
DetailsView Control	414
Storing the Connection String.....	415
Protecting the Configuration String.....	416
Lab 12A	417
FormView Control	418
Master/Detail Web Pages.....	419
Data Binding	422
Template Editing.....	423
Using XML Data.....	425
Example Program.....	426
XML Data Source Demo	427
Multiple-Tier Data Access.....	430
Object Data Source	432
Lab 12B.....	435
Summary	436
Chapter 13 ASP.NET AJAX.....	445
Desktop Applications.....	447
Web Applications.....	448
Plug-Ins	449
Client-Side Scripting.....	450
JavaScript Example.....	451

Script Code	452
JavaScript in ASP.NET	453
Dynamic Pages.....	454
Efficient Page Redraws.....	455
AJAX	456
Google Maps.....	457
ASP.NET AJAX	458
Partial Page Rendering.....	459
Partial Page Rendering Example	460
UpdatePanel Control.....	461
AJAX Extensions Controls	462
UpdatePanel Demo	463
AJAX Client Library	465
Using the Client Library	466
ScriptManager Control	467
Client Library Namespaces.....	468
Sys.Debug Tracing.....	469
Simple Client Library Example	470
Document Object Model.....	471
JavaScript for Simple Calculator.....	472
Using the Client Library	473
Lab 13	474
Summary	475
Chapter 14 ASP.NET MVC	481
Model-View-Controller (MVC)	483
What Is ASP.NET MVC?	484
Advantages of ASP.NET MVC	485
Advantages of Web Forms.....	486
Visual Studio ASP.NET MVC Demo.....	487
Starter Application	489
Simple App with Controller Only	491
Action Methods and Routing	497
Action Method Return Type	498
Rendering a View	499
Creating a View in Visual Studio	500
The View Web Page	501
Dynamic Output.....	502
Razor View Engine	503
Embedded Scripts	504
Embedded Script Example.....	505
Using a Model with ViewBag	506
Controller Using Model and ViewBag	507
View Using Model and ViewBag	508
Using Model Directly	509
A View Using Model in Visual Studio	510

View Created by Visual Studio	511
Using Forms.....	512
HTML Helper Functions	513
Handling Form Submission	514
Model Binding	515
Greet View	516
Input Validation	517
Nullable Type	518
Checking Model Validity.....	519
Validation Summary	520
Lab 14	521
Summary	522
Chapter 15: ASP.NET Web API.....	529
ASP.NET Web API	531
REST	532
Representation, State and Transfer	533
Collections and Elements.....	534
Web API Demo.....	535
Specifying a Start Page	539
Implementing PUT Verb.....	540
Using Fiddler	541
Composing a Request	543
ASP.NET MVC and Web API.....	545
String API Demo.....	546
Route Registration	548
Lab 15A	551
HTTP Response Codes	552
POST Response Code	553
HttpResponseException.....	554
Web API Clients	555
HttpClient.....	556
Initializing HttpClient	557
Issuing a GET Request	558
Issuing a POST Request	559
Lab 15B.....	560
Summary	561
Chapter 16: ASP.NET and Azure	571
What Is Windows Azure?	573
A Windows Azure Testbed.....	574
Windows Azure Demo.....	575
Publish to Azure.....	576
Web Deployment Completed.....	577
Modifying a Web Application	578
Deploy to Original Site	579

Lab 12	580
Summary	581
Appendix A: Learning Resources.....	585
Appendix B: Hosting in IIS 7.5.....	589
Internet Information Services	590
Installing IIS 7.5	591
ASP.NET with IIS 7.5	595
.NET Framework Version.....	596
Registering ASP.NET	597

Chapter 1

Introduction to ASP.NET

Introduction to ASP.NET

Objectives

After completing this unit you will be able to:

- Review the fundamentals of Web applications and set up a testbed using Internet Information Services and ASP.NET.
- Explain the benefits of ASP.NET.
- Describe the programming models provided by ASP.NET: Web Forms, Web services and MVC.
- Create a simple Web Forms application using the .NET Framework SDK.
- Outline the principal features of ASP.NET.

Web Application Fundamentals

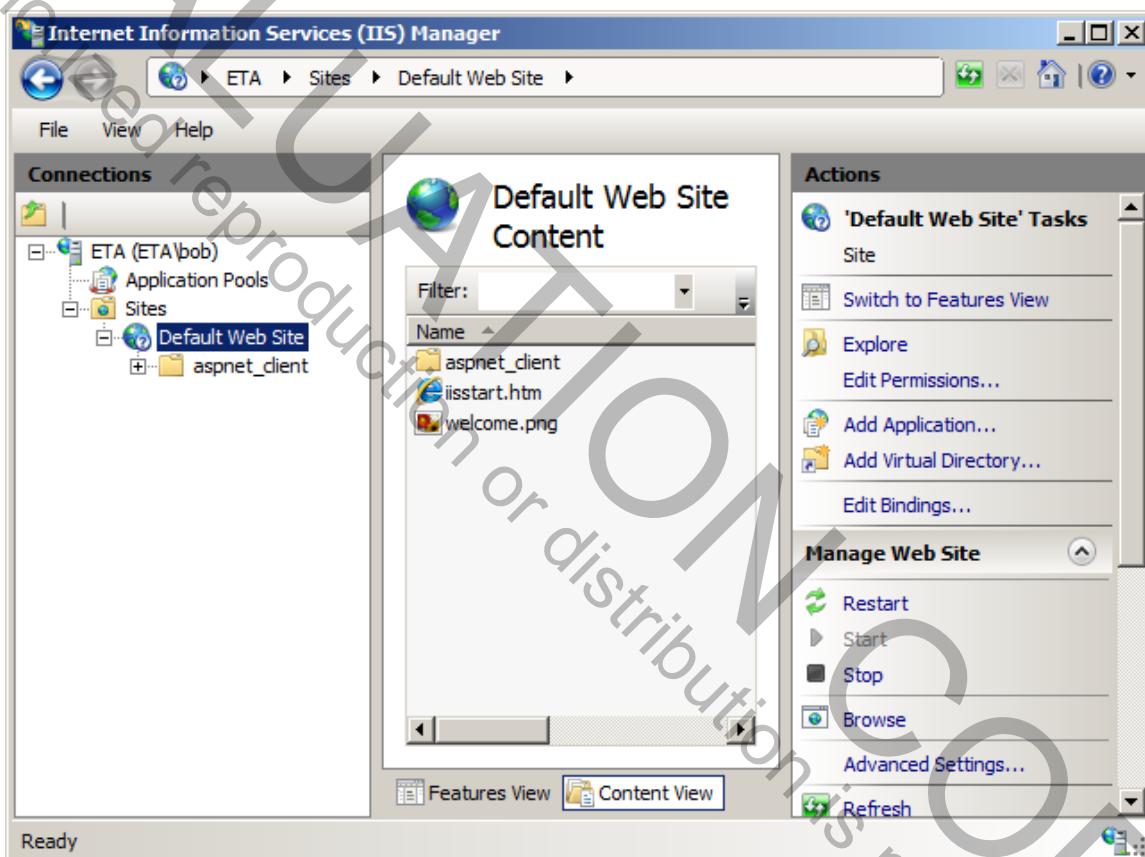
- A Web application consists of document and code pages in various formats.
- The simplest kind of document is a static HTML page, which contains information that will be formatted and displayed by a Web browser.
 - An HTML page may also contain hyperlinks to other HTML pages.
 - A hyperlink (or just “link”) contains an address, or a Uniform Resource Locator (URL), specifying where the target document is located.
- The resulting combination of content and links is sometimes called “hypertext” and provides easy navigation to a vast amount of information on the World Wide Web.

Setting up the Web Examples

- All the example programs for this chapter are in the chapter folder *Chap01* underneath the root folder *|OIC\AspCs*.
- One way to run the examples is to have Internet Information Services (IIS) installed on your system.
 - You will have to explicitly install it with Windows 7.
- Appendix B discusses installing and configuring IIS 7.5 in Windows 7.
- The management tool for IIS is a Microsoft Management Console (MMC) “snap-in,” the Internet Services Manager, which you can find under Administrative Tools in the Control Panel.
- You may run also run the ASP.NET examples using the built-in IIS Express.
 - This built-in Web server is started automatically from within Visual Studio 2017.
 - The use of this development Web server is discussed in Chapter 4

IIS Manager

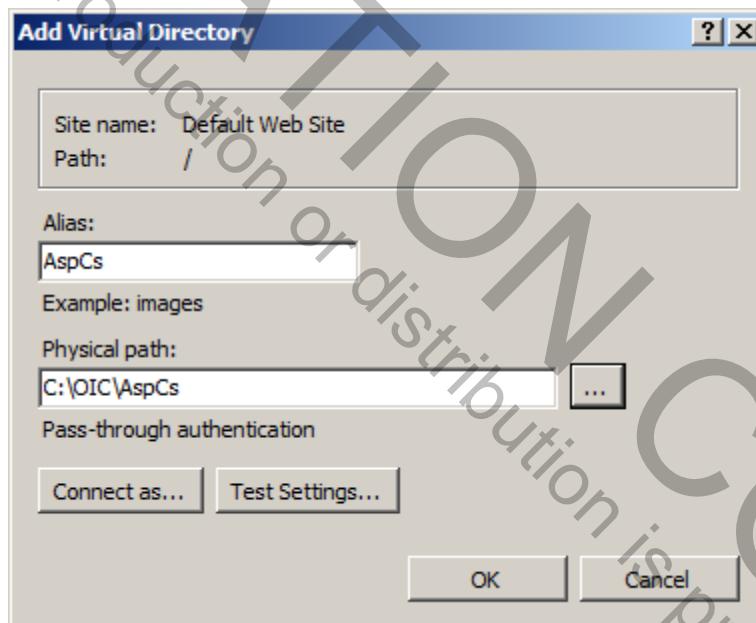
- The figure shows the main window of the Internet Services Manager for IIS 7.5, which comes with Windows 7.
 - We have selected the Content View tab at the bottom of the middle pane.



- You can Start and Stop the Web server and perform other tasks from the Manage Web Site group.
- With Advanced Settings you can change the physical path of the Default Web Site, which by default is located at **\inetpub\wwwroot** on the drive where Windows is installed.

Virtual Directory

- You can access Web pages stored at any location on your hard drive by creating a “virtual directory.”
 - Click Add Virtual Directory¹.
 - You can enter the desired alias, which will be the name of the virtual directory.
 - Browse to the desired physical path and click OK.
- The figure shows creating an alias *AspCs* for the folder *|OIC\AspCs*.

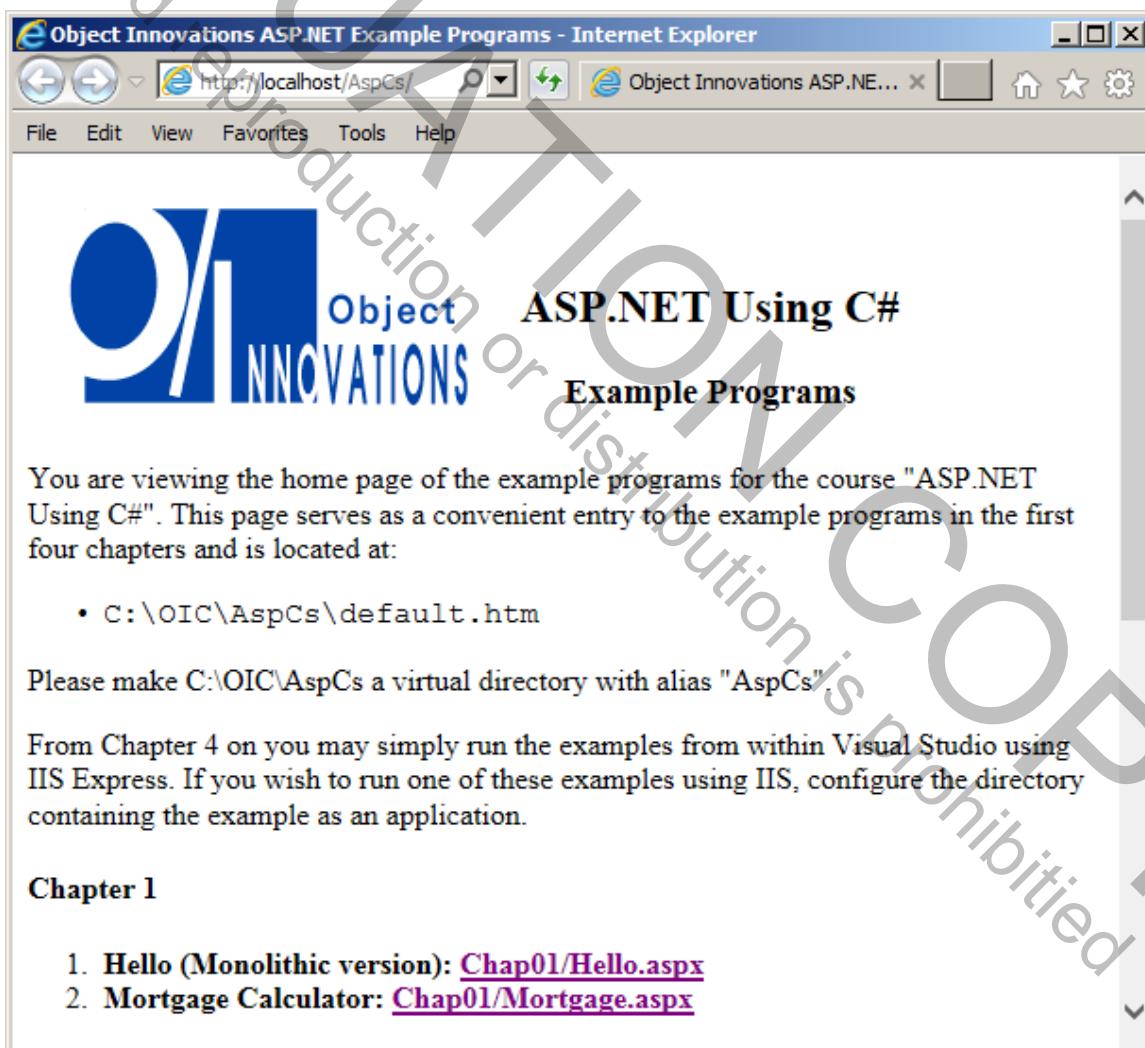


- You should perform this operation now on your own system so that you may follow along as the course examples are discussed.

¹ You can also add a virtual directory from the context menu brought up by a right-click over the Default Web Site.

Home Page for ASP.NET Examples

- Once a virtual directory has been created, you can access files in it by including the virtual directory in the path of the URL.
 - In particular, you can access the file **default.htm** using the URL <http://localhost/AspCs/>.
 - The file **default.htm** contains a home page for all the ASP.NET example programs for the course.



Benefits of ASP.NET

- You can use compiled, object-oriented languages with ASP.NET, including C# and Visual Basic.
 - All the power of the .NET Framework is available to you, including the extensive class library.
- Code and presentation elements can be cleanly separated.
 - Code can be provided in a separate section of a Web page from user interface elements.
 - The separation can be carried a step further by use of separate “code behind” files.
- ASP.NET comes with an extensive set of server controls that provide significant functionality out of the box.
- Server controls transparently handle browser compatibility issues.
- Configuration is handled by XML files without need of any registry settings, and deployment can be done simply by copying files.
- Visual Studio provides a very powerful and easy-to-use environment for developing and debugging Web applications.

ASP.NET Example Program

- We examine an example, *Hello.aspx*, in detail.
 - The example is complete in one file and contains embedded server code. ASP.NET pages have an **.aspx** extension.
- The source code consists of HTML along with some C# script code. There are also some special tags for “server controls,” recognized by ASP.NET.

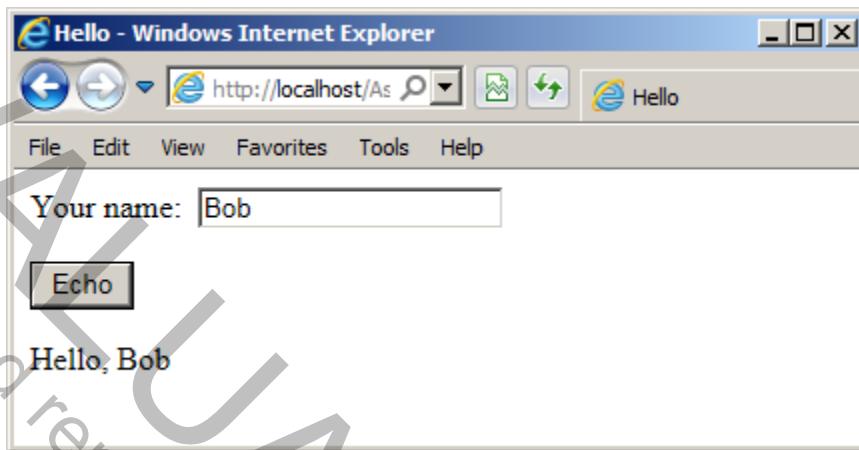
```
<!-- Hello.aspx -->
<%@ Page Language="C#" %>
<html>
<head>
    <title>Hello</title>
    <script runat="server">
        void cmdEcho_Click(object source,
                           EventArgs e)
        {
            lblGreeting.Text = "Hello, " + txtName.Text;
        }
    </script>
</head>
<body>
<form runat="server">Your name:&nbsp;
<asp:textbox ID="txtName" runat="server">
</asp:textbox>
<p><asp:button ID="cmdEcho" OnClick="cmdEcho_Click"
               Text="Echo" runat="server"
               tooltip="Click to echo your name">
</asp:button></p>
<asp:label ID="lblGreeting"
           runat="server"></asp:label>
</form>
</body>
</html>
```

An Echo Program

- You can run the program using the URL
<http://localhost/AspCs/Chap01>Hello.aspx> or by clicking on the link *Chap01/Hello.aspx* in the home page of the examples programs.
 - The page shows a text box where you can type in your name, and there is an “Echo” button.
 - Clicking the button will echo your name back, with a “Hello” greeting.
 - The simple form is again displayed, so you could try out other names.
 - If you slide the browser’s mouse cursor over the button, you will see the tool tip “Click to echo your name” displayed in a yellow box.

An Echo Program (Cont'd)

- The figure illustrates a run of this example.



- This little program would not be completely trivial to implement with other Web application tools, including ASP.
- The key user interface feature of such an application is its thoroughly forms-based nature.
 - The user is presented with a form and interacts with the form.
 - The server does some processing, and the user continues to see the same form.
 - This UI model is second nature in desktop applications but is not so common in Web applications.
 - Typically, the Web server will send back a different page.

An Echo Program (Cont'd)

- This kind of application could certainly be implemented using a technology like ASP, but the code would be a little ugly.
- The server would need to synthesize a new page that looked like the old page, creating the HTML tags for the original page, plus extra information sent back (such as the greeting shown at the bottom in our echo example).
 - A mechanism is needed to remember the current data that is displayed in the controls in the form.
- Another feature of this Web application is that it does some client-side processing too—the Echo button's tooltip displayed in a yellow box is performed by the browser.
 - Such rich client-side processing can be performed by modern browsers, such as Internet Explorer and Firefox.
- As can be seen by the example code, with ASP.NET it is very easy to implement this kind of Web application.
- We will study the code in detail later.
 - For now, just observe how easy it is!

ASP.NET Features

- ASP.NET provides a programming model and infrastructure that facilitates developing new classes of Web applications.
- Part of this infrastructure is the .NET runtime and framework.
- Server-side code is written in .NET compiled languages.
- Three main Web programming models are supported by ASP.NET.
- Web Forms helps you build form-based Web pages. A WYSIWYG development environment enables you to drag controls onto Web pages.
 - Special “server-side” controls present the programmer with an event model similar to what is provided by controls in ordinary Windows programming.
- ASP.NET MVC is a newer framework that provides an alternative to Web Forms for creating Web applications.
 - It is based on the Model-View-Controller design pattern.
- ASP.NET Web API supports the creation of HTTP services.
- ASP.NET MVC and ASP.NET Web API are introduced in the last two chapter of this course.

ASP.NET Features (Cont'd)

- ASP.NET also supports Web Services, which make it possible for a Web site to expose functionality via an API that can be called remotely by other applications.
 - Data is exchanged using standard Web protocols and formats such as HTTP and XML, which will cross firewalls.
 - A newer technology, Windows Communication Foundation, is now preferred for implementing Web services, along with Web API for HTTP services.
- Web Forms, ASP.NET MVC, ASP.NET Web API and ASP.NET Web services can all take advantage of the facilities provided by .NET, such as the compiled code and .NET runtime.
- In addition, ASP.NET itself provides a number of infrastructure services, including:
 - state management
 - security
 - configuration
 - caching
 - tracing

Compiled Code

- **Web Forms can be written in any .NET language that is compatible with the common language runtime, including C# and Visual Basic.**
 - This code is compiled, and thus offers better performance than ASP pages with code written in an interpreted scripting language such as VBScript.
- **Compilation normally occurs at HTTP request time, and subsequent accesses to the page do not require compilation.**
 - The ASP.NET compilation model is described in the next chapter.
- **All of the benefits, such as a managed execution environment, are available to this code, and of course the entire .NET Framework Class Library is available.**
 - Legacy unmanaged code can be called through the .NET interoperability services.

Server Controls

- ASP.NET provides a significant innovation known as “server controls.” These controls have special tags such as `<asp:textbox>`.
- Server-side code interacts with these controls, and the ASP.NET runtime generates straight HTML that is sent to the Web browser.
 - The result is a programming model that is easy to use and yet produces standard HTML that can run in any browser.

Browser Independence

- **Although the World Wide Web is built on standards, the unfortunate fact of life is that browsers are not compatible and have special features.**
 - A Web page designer then has the unattractive options of either writing to a lowest common denominator of browser, or else writing special code for different browsers.
 - Server controls help remove some of this pain.
- **ASP.NET takes care of browser compatibility issues when it generates code for a server control.**
 - If the requesting browser is upscale, the generated HTML can take advantage of these features, otherwise the generated code will be vanilla HTML.
 - ASP.NET takes care of detecting the type of browser.

Separation of Code and Content

- Typical ASP pages have a mixture of scripting code interspersed with HTML elements.
- In ASP.NET there can be a clean separation between code and presentation content.
 - The server code can be isolated within a single `<script runat="server"> ... / script>` block or, even better, placed within a “code behind” page.
- We will discuss "code-behind" pages in the next chapter.

State Management

- **HTTP is a stateless protocol.**
- **Thus, if a user enters information in various controls on a form, and sends this filled-out form to the server, the information will be lost if the form is displayed again, unless the Web application provides special code to preserve this state.**
 - ASP.NET makes this kind of state preservation totally transparent.
 - There are also convenient facilities for managing other types of session and application state.

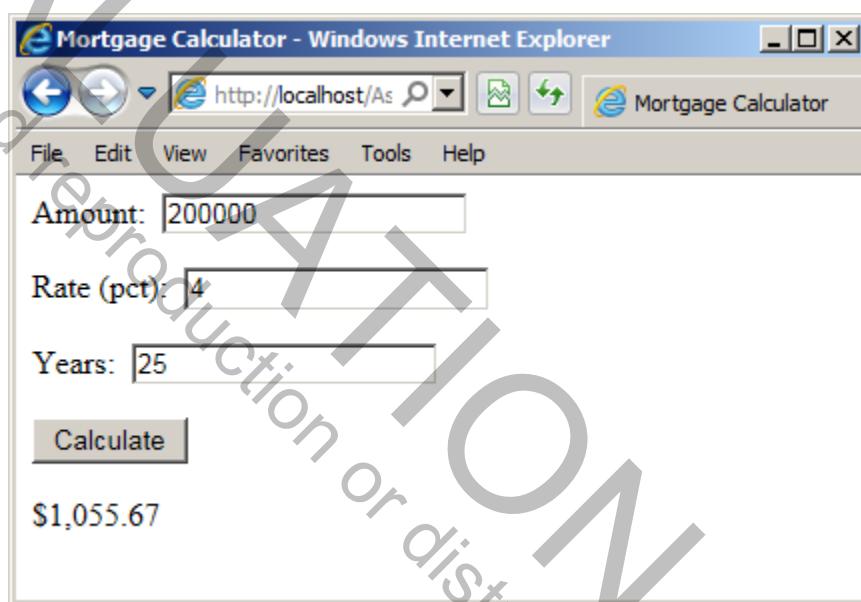
ASP.NET Core

- **ASP.NET Core is a completely new version of ASP.NET that is open-source and cross-platform.**
 - It runs on Windows, Mac and Linux.
- **It runs on either .NET Core or the full .NET Framework.**
- **ASP.NET Core is very modular providing flexibility in using what is needed for an application with minimal overhead.**
- **ASP.NET Core supports both MVC and Web API in an integrated manner, but it does not support Web Forms.**
- **This course focuses on classical ASP.NET, with an emphasis on Web Forms.**
 - ASP.NET Core is introduced in the Object Innovations course 4043, ASP.NET Core MVC.

Lab 1

A Mortgage Calculator Web Page

You have received a consulting contract to add useful functionality to a realtor's Web site. In this lab you will implement the first feature, which is a Web page that can be used to calculate a mortgage payment.



Detailed instructions are contained in the Lab 1 write-up at the end of the chapter.

Suggested time: 30 minutes

Summary

- **ASP.NET is a unified Web development platform that greatly simplifies the implementation of sophisticated Web applications.**
- **ASP.NET supports three Web programming models, Web Forms, ASP.NET MVC and ASP.NET Web API.**
- **Server controls present the programmer with an event model similar to what is provided by controls in ordinary Windows programming.**
- **Other features of ASP.NET include:**
 - Compiled code
 - Browser independence
 - Separation of code and content
 - State management
 - Security
 - Configuration
 - Tracing
 - Caching

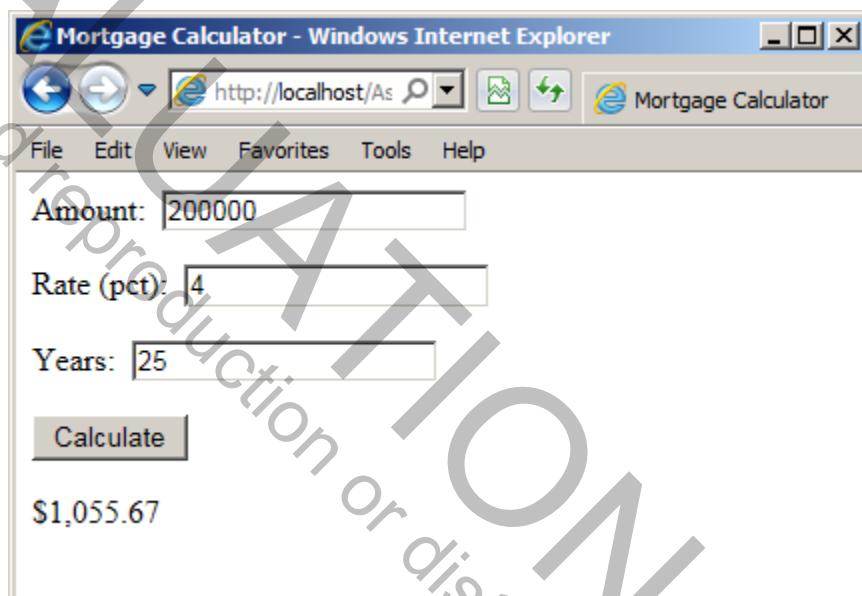
Lab 1

A Mortgage Calculator Web Page

Introduction

You have received a consulting contract to add useful functionality to a realtor's web site. In this lab you will implement the first feature, which is a Web page that can be used to calculate a mortgage payment.

The screen capture shows the completed Web page with a sample calculation.



Suggested Time: 30 minutes

Root Directory: OIC\AspCs

Directories: Labs\Lab1
Chap01\Hello.aspx
Chap01\MortgageCalculator

(do your work here)
(starter file)
(console mortgage calculator)

Files: Chap01\Mortgage.aspx

(answer)

1. As a starter file, copy the file **Hello.aspx** into **Labs\Lab1** and rename as **Mortgage.aspx**. You should now be able to access the “echo” page through the URL <http://localhost/AspCs/Labs/Lab1/Mortgage.aspx>.
2. Change the title to “Mortgage Calculator”.

3. Create the user interface for your Web mortgage calculator by making two copies of the textbox for the name in Hello. Rename the three textboxes, one button and one label appropriately. Rename the button handler to match the new name of the button.
4. Examine the code for the C# console program in the **MortgageCalculator** folder. Run this program a couple of times to generate some test data for use in testing your Web mortgage calculator.
5. Implement the event handler for the Calculate button by copying in code from the **MortgageCalculator** project. For the Web version you will obtain a numerical value by using the **Convert** class. For example,

```
decimal amount; // amount of mortgage  
...  
amount = Convert.ToDecimal(txtAmount.Text);
```

6. After you have calculated **calcPymt**, display it formatted as currency in the label **lblPayment**. You can do the formatting by the **String.Format** method.

```
lblPayment.Text = String.Format("{0, 8:C}", calcPymt);
```

7. Save your file and test it in the browser. Try out the test data obtained from running the console version of the program.

Chapter 6

Server Controls

EVALUATION COPY
Unauthorized reproduction or distribution is prohibited

Server Controls

Objectives

After completing this unit you will be able to:

- **Discuss the role of server controls in ASP.NET.**
- **Use HTML server controls to wrap standard HTML controls.**
- **Use Web Forms server controls as the preferred controls for new work.**
- **Describe rich controls and illustrate with the calendar control.**
- **Use validation controls to validate user input data.**
- **Describe how user controls can facilitate code reuse in ASP.NET.**

Server Controls in ASP.NET

- **An important innovation in ASP.NET is server controls.**
 - They provide an event model that is startlingly similar to Windows GUI programming, and they encapsulate browser dependencies.
 - They integrate seamlessly into the Visual Studio development environment.
 - The end result is an extremely powerful tool for Web development.
- **We have been using server controls from the very beginning of the course, where we presented our “Hello” program.**
- **In this chapter we will look at server controls more systematically, and we will see a number of examples of interesting controls.**
 - The controls discussed in this chapter have been available since .NET 1.1 or earlier.
 - Later, we look at a number of powerful new controls introduced in .NET 2.0, .NET 3.5 and .NET 4.0.

HTML Server Controls

- **HTML server controls provide equivalent functionality to standard HTML controls, except that they run on the server, not on the client.**
 - You distinguish an HTML server control from an ordinary HTML control on a Web page by using the **runat="server"** attribute.
- **Here are two controls. Both are INPUT controls.**
 - The first is a server control.
 - The second is of type password and is a regular HTML control.

```
<INPUT id=txtUserId  
       style="WIDTH: 135px; HEIGHT: 22px"  
       type=text size=17 runat="server"></P>  
<INPUT id= ""  
       style="WIDTH: 138px; HEIGHT: 22px" type=password  
       size=17 name=txtPassword>
```

Using HTML Server Controls

- Working with HTML server controls is much like working with the Web Forms server controls we've used already.
 - In server-side code you access the control through a control variable that has the same name as the **id** attribute.
- However, we are dealing with HTML controls, so there are some differences.
 - You access the string value of the control not through the **Text** property but through the **Value** property.
- Here is some code that uses the value entered by the user for the *txtUserId* HTML server control.

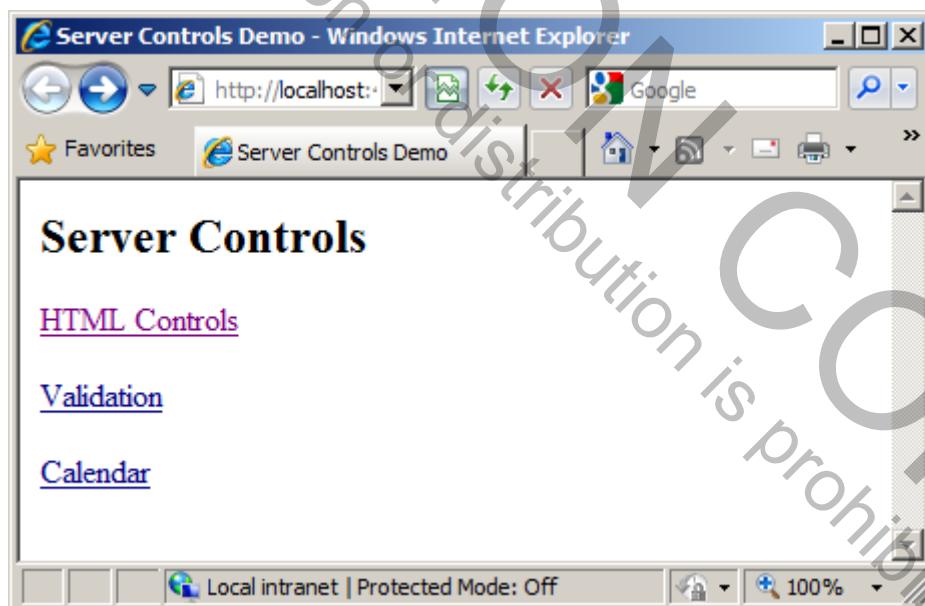
```
lblMessage.Text = "Welcome, " + txtUserId.Value;
```

HTML vs. Web Forms Server Controls

- The advantage of HTML server controls for the experienced Web programmer is that they match ordinary HTML controls exactly.
 - Thus your knowledge of the details of HTML control properties and behavior carries over to the ASP.NET world.
 - The main reason for HTML server controls is that it makes it easy to convert HTML to ASPX pages.
- However, this similarity means they carry over all the quirks and inconsistencies of HTML.
- For example, there are not two different controls for the somewhat different behaviors of a textbox and a password control.
 - HTML uses in both cases the INPUT control, distinguishing between the two by the `type=password` attribute.
- Web Forms controls, in contrast, are a fresh design and have an internal consistency.
- Also, as we shall soon see, there is a much greater variety to Web Forms controls.

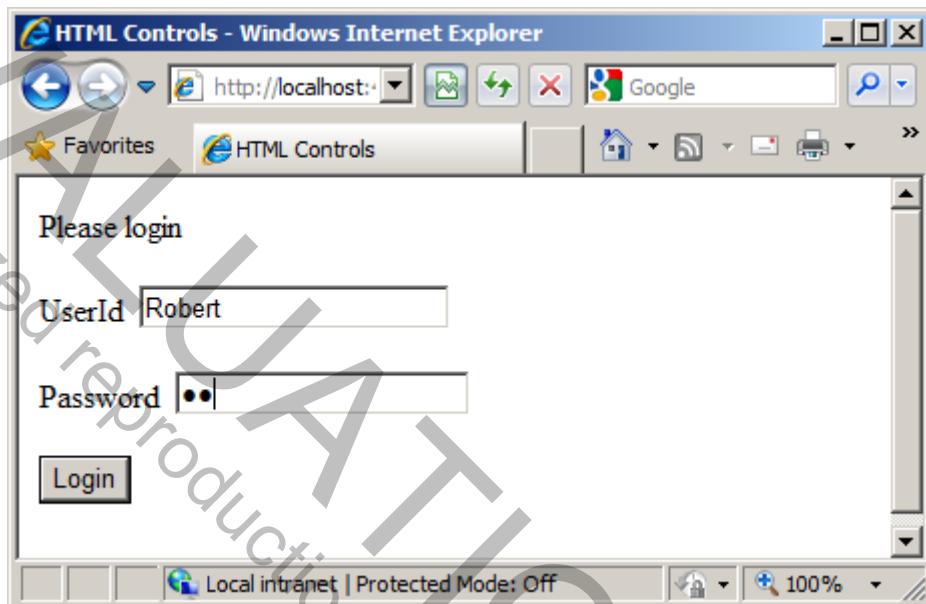
Server Control Examples

- All of our server control examples can be accessed from *Chap06\ServerControls\Default.aspx*.
 - As usual, you should use IIS to configure the folder **ServerControls** as an application if you run under IIS.
 - Or, you can simply run from within Visual Studio.
- The top-level page gives you a choice of three examples,
 - HTML Controls
 - Validation
 - Calendar



HTML Controls Example

- Follow the link to **HTML Controls**, and you will come to a login page.



- There is a **textbox** for entering a user ID and a **password control** for entering a password.
 - Both of these controls are HTML INPUT controls, as shown previously.
 - The textbox runs at the server, and the password is an ordinary HTML control.

Code for Login

- Clicking the Login button (implemented as a Web Forms Button control) results in very simple action.
 - There is one legal password (valid for all users in this toy example!), hardcoded at “77”.
 - The button event handler checks for this password. If legal, it displays a welcome message; otherwise, an error message.

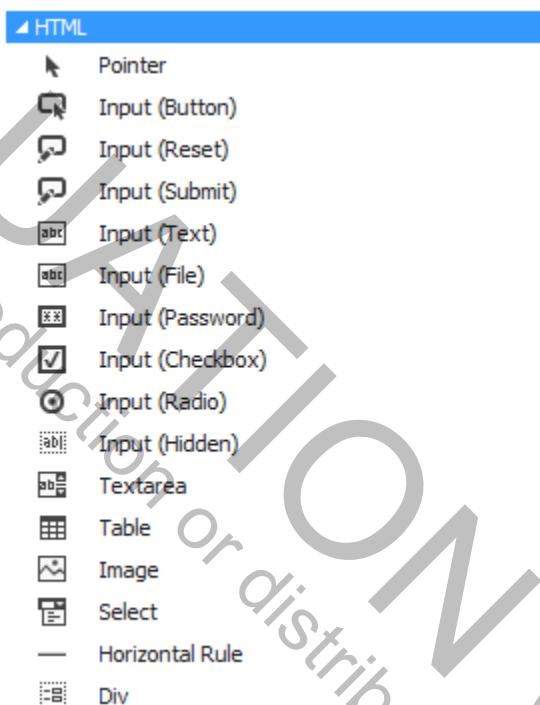
```
private void Login_Click(object sender,
EventArgs e)
{
    if (Request.Params["txtPassword"] == "77")
        lblMessage.Text = "Welcome, "
            + txtUserId.Value;
    else
        lblMessage.Text = "Illegal password";
}
```

- Since the password control is not a server control, no server control variable is available for accessing the value.
 - Instead, we must rely on a more fundamental technique, such as using the **Params** collection of the **Request** class.
 - Note that for indexing in the **Params** collection we should identify our HTML control with the **name** property rather than **id**.

```
<input name="txtPassword" type="password" /><br />
```

HTML Controls in Visual Studio

- It is easy to work with HTML controls in Visual Studio.
- The Toolbox has a palette of HTML controls, which you can access through the HTML tab.



- There is only one palette for HTML controls.
 - You distinguish between classical HTML controls and server HTML controls by the **runat="server"** attribute.

Using HTML Controls

- You can drag HTML controls onto a form, just as we have done with Web Forms controls.
- The default choice for HTML controls is not to run at the server.
- To make an HTML control into a server control, edit the source file to specify `runat="server"`.
 - You'll find IntelliSense there to help you.

```
Please login<br />
<br />
UserId ;
<input id="txtUserId" type="text"
      runat="server"/><br />
<br />
Password ;
<input name="txtPassword" type="password" /><br />
<br />
<asp:Button ID="btnLogin" runat="server"
            Text="Login" />
<br />
<br />
<asp:Label ID="lblMessage"
            runat="server"></asp:Label>
```

- You can inspect the **runat** property in the Properties panel, but you cannot change it there.

Web Controls

- The most important kind of control in ASP.NET is the *Web Forms server control* or just *Web control*.
- These are new controls provided by the .NET Framework, with special tags such as `<asp:textbox>`.
 - These controls run at the server, and they generate HTML code that is sent back to the browser.
 - They are easy to work with, because they behave consistently.
 - For example, you can determine the value returned by a control by using simple property notation.

```
string name = txtName.Text;
```

- All of our previous examples of server controls have been Web controls.
- In this section, we will look at several additional kinds of Web controls, including validation controls, list controls, and rich controls such as the Calendar control.

Validation Controls

- The rest of our discussion of server controls will focus on Web controls.
- A very convenient category of control is the group of validation controls.
 - The basic idea of a validation control is very simple.
 - You associate a validation control with a server control whose input you want to validate.
 - Various kinds of validations can be performed by different kinds of validation controls.
 - The validation control can display an error message if the validation is not passed.
 - Alternatively, you can check the **IsValid** property of the validation control. If one of the standard validation controls does not do the job for you, you can implement a custom validation control.

Validation Controls (Cont'd)

- **The following validation controls are available:**
 - RequiredFieldValidator
 - RangeValidator
 - CompareValidator
 - RegularExpressionValidator
 - CustomValidator
 - There is also a ValidationSummaryControl that can give a summary of all the validation results in one place.
- **An interesting feature of validation controls is that they can run on either the client or the server, depending on the capabilities of the browser.**
 - With an upscale browser such as Internet Explorer, ASP.NET will emit HTML code containing JavaScript to do validation on the client.
 - If the browser does not support client-side validation, the validation will be done only on the server.

Required Field Validation

- A very simple and useful kind of validation is to check that the user has entered information in required fields.
 - Our second server control demonstration page provides an illustration.
 - Back on the top-level **ServerControls\Default.aspx** page, follow the link to “Validation”.
- You will be brought to *RegisterNewUser.aspx*.

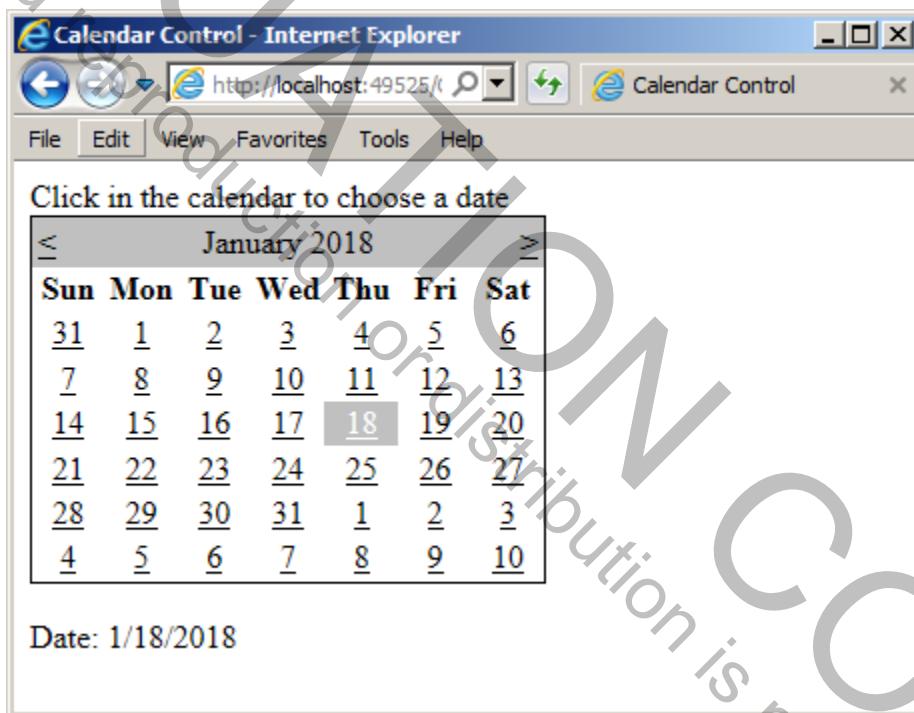
The screenshot shows a Windows Internet Explorer window titled "Register New User - Windows Internet Explorer". The URL in the address bar is "http://localhost:...". The page itself is titled "Register New User" and contains instructions: "Please choose a UserId and Password (letters and numbers)". It has two text input fields: "UserId" containing "bob66*" and "Password" containing "*****". Below these, it asks "Please enter some information about yourself" and has two more text input fields: "First Name" containing "Robert" and "Last Name" which is empty. Red validation messages are displayed next to the empty "Last Name" field ("Must not be blank") and the "UserId" field ("Must use letters and digits"). A "Register" button is at the bottom. The status bar at the bottom of the browser window shows "Local intranet | Protected Mode: Off" and "100%".

Regular Expression Validation

- The *RegularExpressionValidator* control provides a very flexible mechanism for validating string input.
- It checks whether the string is a legal match against a designated regular expression.
 - Our example illustrates performing a regular expression validation of UserId. The requirement is that the id consist only of letters and digits, which can be specified by the regular expression
- The following properties should normally be assigned for a RegularExpressionValidator control:
 - **ValidationExpression** (the regular expression, not surrounded by quotes)
 - **ControlToValidate**
 - **ErrorMessage**
- You can try this validation out on our Register New User page by entering a string for UserId that contains a non-alphanumeric character.

Rich Controls

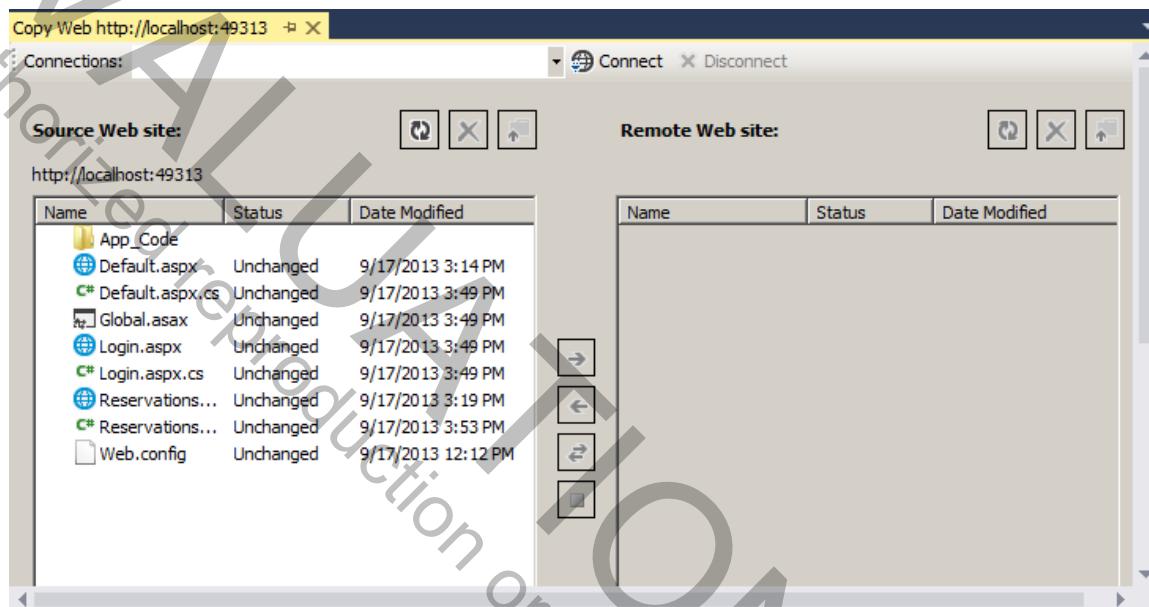
- Another category of Web Forms controls consists of “rich controls,” which can have quite elaborate functionality.
 - The Calendar control provides an easy-to-use mechanism for entering dates on a Web page.
- Our third sample server control page provides an illustration, as shown in the figure.



- The user can select a date on the Calendar control.
- The **SelectedDate** property then contains the selected date as an instance of the **DateTime** structure.

Copying a Web Site

- Visual Studio provides a convenient means for copying a whole Web site, using various protocols.
 - Open a Web site and select menu Website | Copy Web Site.



- Click Connect, choose the desired protocol, and navigate to the desired target location.
- You can then select all or some of the files and folders to be copied, and can click an arrow button to perform the copy operation.
- Of course, for the file system protocol you could also simply copy files using Windows Explorer.

Lab 6

Using a Calendar Control

In this lab you use the Calendar control in a real application. You will improve the user interface for booking hotel reservations in the Acme Web site by letting the customer pick a date by using a Calendar control.

Customer John Smith Number Days

Cities Manage My Reservations

Hotels Make Reservation

December 2013						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
<u>24</u>	<u>25</u>	<u>26</u>	<u>27</u>	<u>28</u>	<u>29</u>	<u>30</u>
<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>
<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>	<u>14</u>
<u>15</u>	<u>16</u>	<u>17</u>	<u>18</u>	<u>19</u>	<u>20</u>	<u>21</u>
<u>22</u>	<u>23</u>	<u>24</u>	<u>25</u>	<u>26</u>	<u>27</u>	<u>28</u>
<u>29</u>	<u>30</u>	<u>31</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>

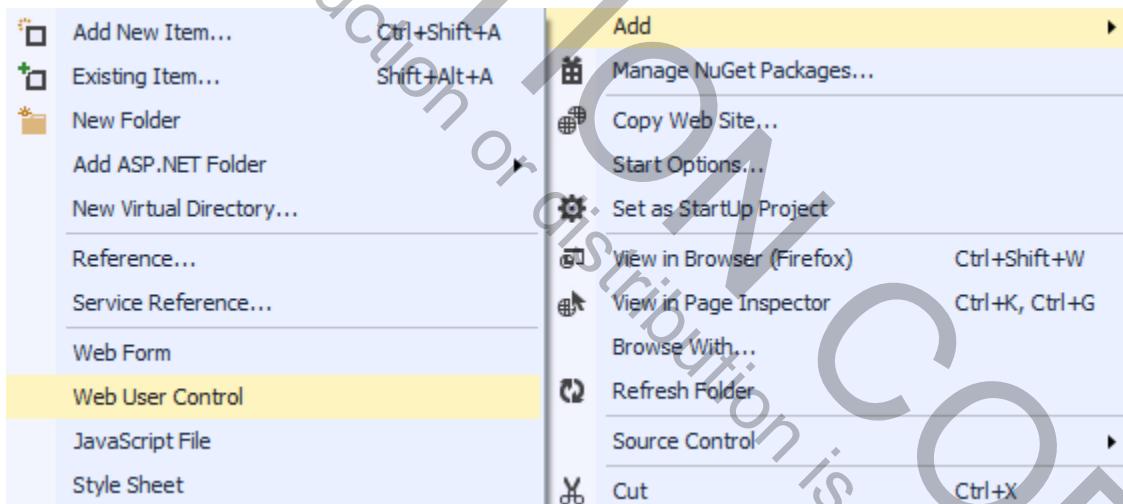
Reservation has been booked
ReservationId = 1, ReservationRate = 115.00
ReservationCost = 575.00
OK

Detailed instructions are contained in the Lab 6 write-up at the end of the chapter.

Suggested time: 30 minutes

User Controls

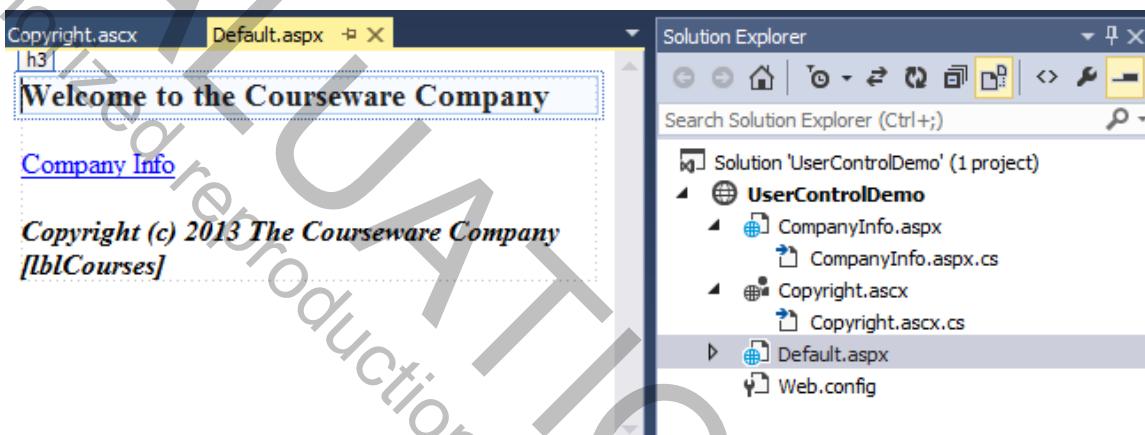
- **Web user controls enable you to create reusable code that can be used on multiple pages of a Web application.**
 - It can both provide a user interface and implement programming logic.
 - Physically, a user control consists of a file with a **.ascx** extension (and an associated **.ascx.cs** if code-behind is used).
- **You can create a new user control in Visual Studio by a right-click over the project and Add | Web User Control.**



- You can also find it in the Add New Item dialog.

Using a User Control

- In Visual Studio you can simply drag a user control's .ascx file onto your form.
 - The example **UserControlDemo** illustrates a “copyright” user control that has been dragged to the bottom of a Web form.



- To code a user control on a Web Form, do two things:

- Register the control, giving a tag prefix and name and specifying the .ascx file.

```
<%@ Register Src="Copyright.ascx"  
TagName="Copyright" TagPrefix="oi"%>
```

- Use the control as a new tag, enhancing HTML and ASP.NET tags available to you.

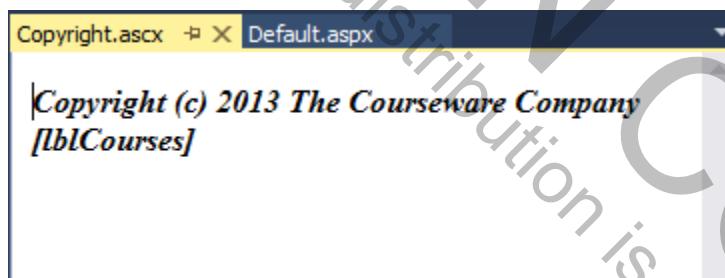
```
<oi:Copyright ID="Copyright1" runat="server"/>
```

Copyright.ascx

- The *Copyright.ascx* file is like an *.aspx* file with some differences.
 - There is no `<html>`, `<body>`, or `<form>` tag, because the page is not meant to be used standalone but to be included in a normal Web Form page.
 - Directive at top specifies **Control**.

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="Copyright.ascx.cs" Inherits="Copyright"
%>
<strong><em>Copyright (c) 2013 The Courseware
Company<br />
    <asp:Label ID="lblCourses"
runat="server"></asp:Label></em></strong>
```

- In Visual Studio you can create a user control in design mode.



Copyright.ascx.cs

- The code-behind file is similar to that for a Web Form, but the class derives from *UserControl* in the namespace *System.Web.UI*.

```
public partial class Copyright :  
System.Web.UI.UserControl  
{  
    protected void Page_Load(object sender,  
EventArgs e)  
    {  
        // Read number of courses from a file  
        StreamReader reader = File.OpenText(  
            @"c:\OIC\Data\courses.txt");  
        string strCount = reader.ReadLine();  
        lblCourses.Text = strCount + " courses";  
        reader.Close();  
    }  
}
```

User Control Example

- We provide a *Copyright* control that can be reused on multiple pages on the website of a courseware company.
 - The sample code is provided in **UserControlDemo**.
- The control provides two features:
 - Display of a copyright notice.
 - Display of the current number of courses offered by the company (read from a file in this example).



Summary

- Server controls provide an event model that is similar to Windows GUI programming, and they encapsulate browser dependencies.
- HTML server controls wrap standard HTML controls, along with all their idiosyncrasies.
- Web Forms server controls provide a consistent and easy to use programming model.
- ASP.NET comes with a variety of rich controls with powerful functionality. The calendar control is a good example.
- You can use validation controls to validate user input data.
- User controls enable you to encapsulate reusable user interface and logic elements for use in multiple pages of a website.

Lab 6

Using a Calendar Control

Introduction

In this lab you use the Calendar control in a real application. You will improve the user interface for booking hotel reservations in the Acme Web site by letting the customer pick a date by using a Calendar control.

Suggested Time: 30 minutes

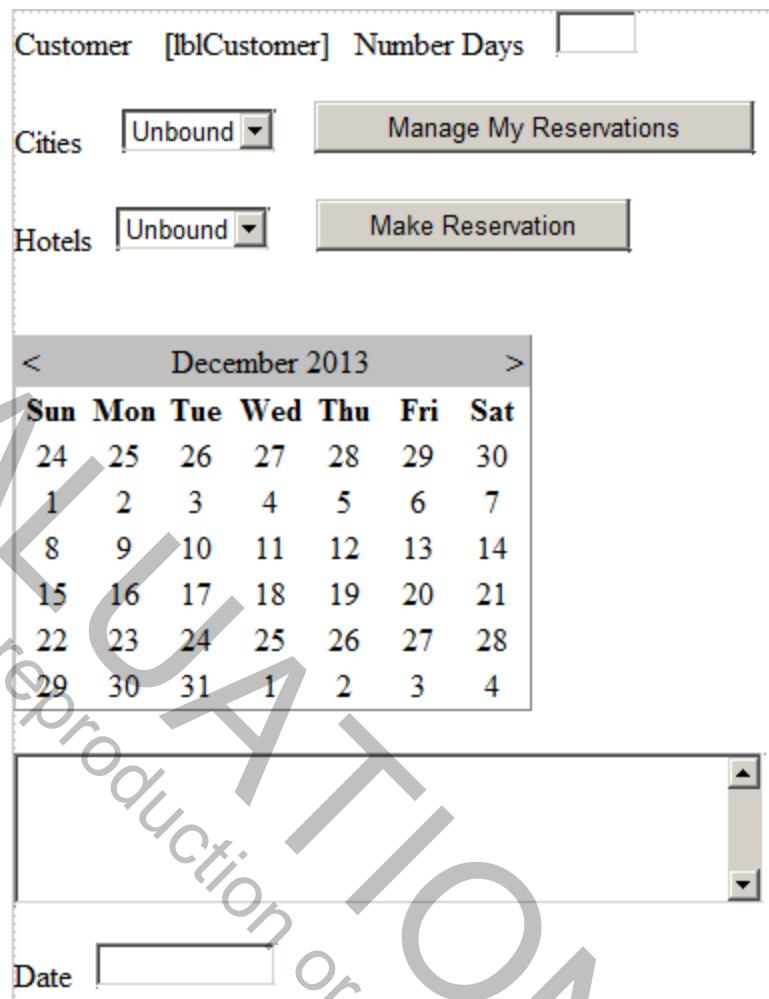
Root Directory: OIC\AspCs

Directories:	Labs\Lab6\AcmeWeb	(do your work here)
	Labs\Lab5B\AcmeWeb	(previous lab)
	CaseStudy\AcmeWeb\Step3	(starter code to copy)
	CaseStudy\AcmeWeb\Hidden	(quick solution)
	CaseStudy\AcmeWeb\Step4	(answer)

1. Use Website| Copy Web Site to deploy the Web site from either your own previous lab work or **CaseStudy\AcmeWeb\Step3** to the working directory. Verify that you can build and run the starter project from the new location.
2. Rearrange the controls as shown on the next page. Temporarily keep the textbox for entering a date. Drag a calendar control onto your form, first providing additional space for it. Don't worry about the form being pretty.
3. You can get a quick and dirty implementation of using the Calendar control by adding a handler for the calendar control's **SelectionChanged** event and imitating the code from the Calendar control example in the chapter. This code assigns a textbox representing the date selected by the user. You leave the rest of the program as it is. You use the date in making a reservation by parsing the string from the textbox. Verify that this solution works.

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    txtDate.Text = Calendar1.SelectedDate.ToShortDateString();
}
```

4. This solution is not very elegant, because you are converting a date to a string and then back again. It is also confusing to the user to have two controls representing the date. You can at least improve the solution's appearance by deleting the label for the Date textbox and hiding the textbox (set the **Visible** property to false). This solution is saved in the **CaseStudy\AcmeWeb\Hidden** folder.



5. As a first attempt at an alternative solution, add a **DateTime** instance variable **m_date**. You can assign this variable when the user selects the date, and use the variable when making a reservation. Build and run. It does not work! Why?
6. Try making **m_date** a **static** variable. Now it “works.” But this is not a valid solution. Why not?
7. Devise a valid solution and test it.

Lab 6 Answer

5. This solution does not work because HTTP is a stateless protocol, and so the instance variable **m_date** is not preserved. (The “date” will be 1/1/0001.)
6. Making **m_date** a **static** variable “works” because the value is now preserved as part of application state. But this is not a valid solution, because one user could wind up making a reservation using a date selected by another user. Moreover, the code is not safe for multithreading.
7. A valid solution is to use session state. Now the date is correct for each user, and there are no threading issues. See the code in the Step 4 answer folder.



7400 E. Orchard Road, Suite 1450 N
Greenwood Village, Colorado 80111
Ph: 303-302-5280
www.ITCourseware.com

EVALUATION COPY
Unauthorized reproduction or distribution is prohibited