INTRODUCTION

- Active Server Pages (ASP)
- ASP.NET is a web development framework that provides the developer with an advanced, object oriented interface to build enterprise- class web applications.
- Syntactically compatible with ASP.
- Part of . NET framework. Gets all the benefits (Type Safety, Inheritance, language interoperability and versioning) of the CLR.
- Basically used for creating dynamic web pages.
- Developers can embed scripting language (VBScript or JavaScript) code directly in their web pages which is difficult to handle.

FEATURES

- Compiled Code, not Interpreted.
- Separation of code from presentation.
- No more “DLL hell”
- Side by Side installation
- Support of Number of languages
- Runtime debugging
- Easy of Deployment
- Event Based Programming model
- Web services, ADO.NET
- Increase dev. Productivity (automatic code generation & Intelli sense features).
- OO based classes for pages, applications, user controls, server controls etc… which are easy to handle.
- Deployment support

Features

- Easier use of ASP.NET with existing devices like web browsers, PDA, cell phones and so on..
- Increase in performance and scalability
- 100 % backward compatibility
- Master pages
- Site Navigation
- Web parts
- Configuration tools
- Personalization
- Themes and Skins
- User Management
- Provides security (has features like user authentication using cookies, windows authentication, MS passport.)
- Mobile devices support
- Site Counters
-

FEATURES

- Maintains session state between multiple requests for a page. (state management)
- Configuration files

Dynamic Web Technologies

Different technologies for dynamically generating Web content.

- ✓ ASP.NET
- ✓ ASP
- ✓ CGI
- ✓ ColdFusion
- ✓ JSP
- ✓ PHP
- ✓ Ruby on Rails

All of these technologies share one thing in common:

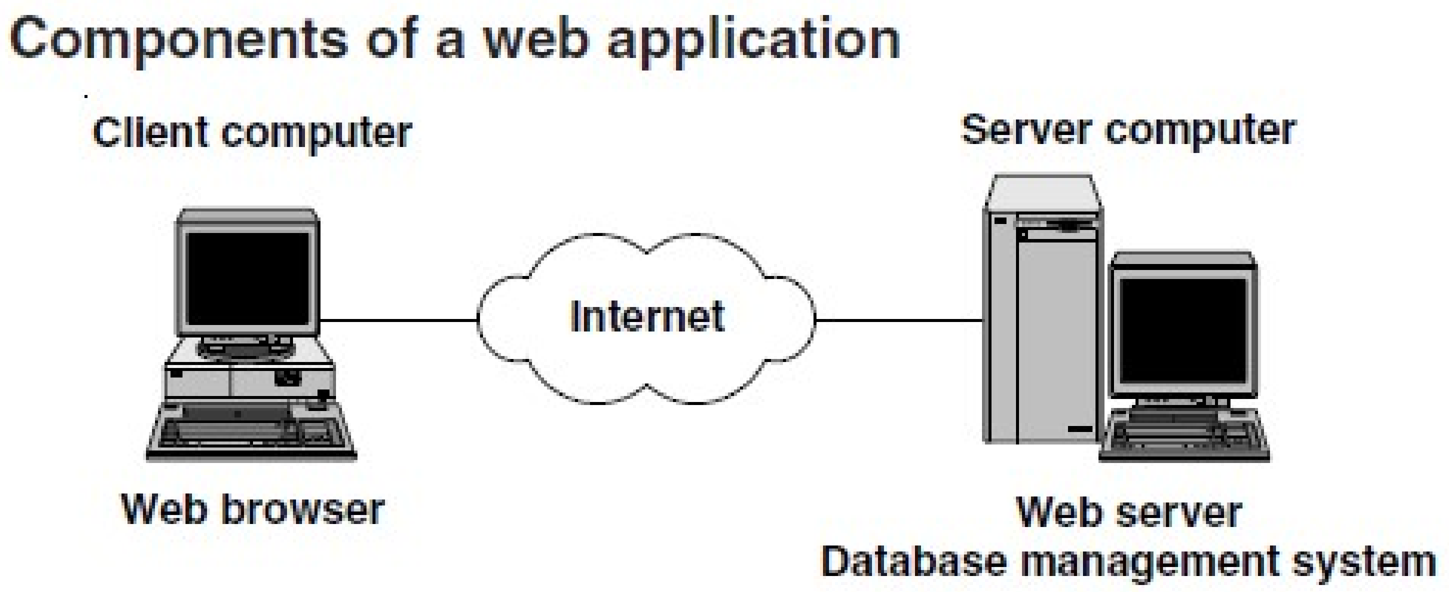
Using programming logic, they generate HTML on the server and send it back to the requesting browser.

desktop vs web application

A desktop application is a computer program that runs locally on a computer.

- ✓ A web application, is an application which runs on the remote server and can be accessed from any machine.
- ✓ Security & Safety is a problem with web appl.
- ✓ Desktop applications are more secure and safe.
- ✓ Web applications availability depends on the internet connection & speed.
- ✓ Maintenance of Web applications is easy compared with desktop applications.
- ✓ Testing of Web applications is difficult than desktop applications.

Web Application

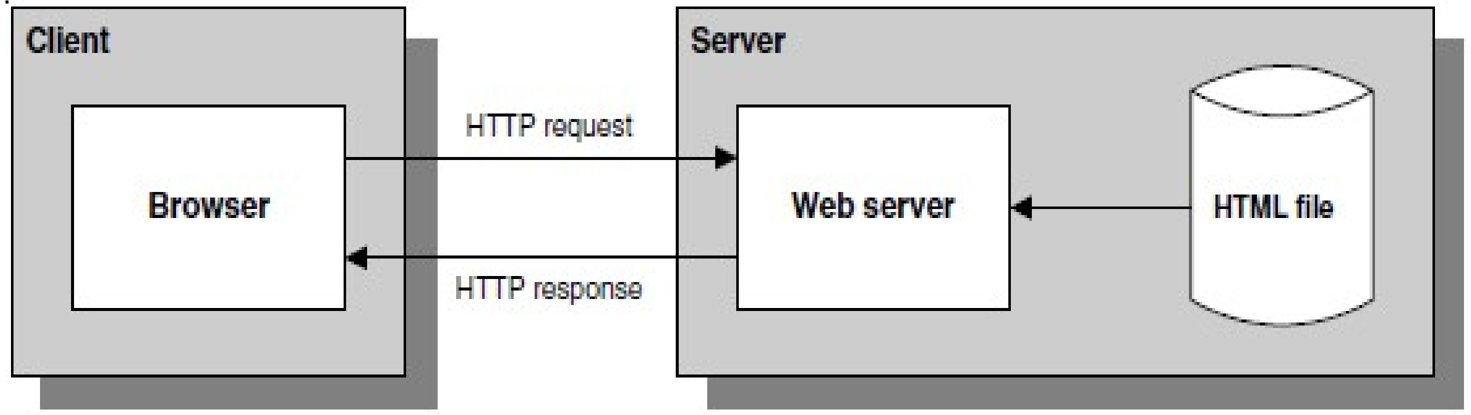


Static Pages & Dynamic Pages

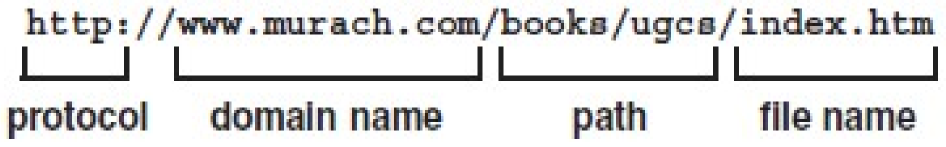
Static pages : Static pages don't change in response to user input. Stored as .html files in the disk.

How static web pages work :

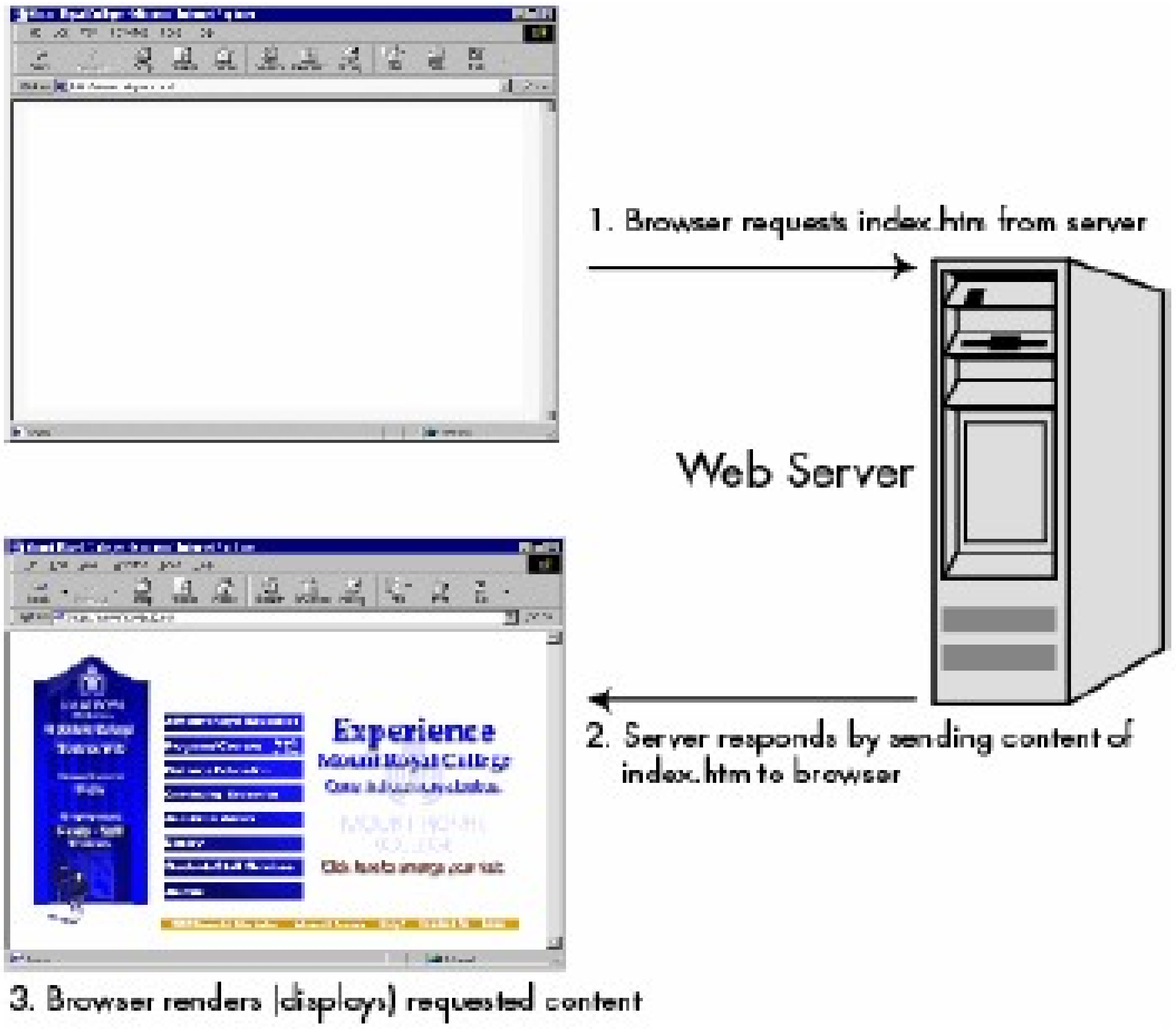
How a web server processes static web pages



The components of an HTTP URL



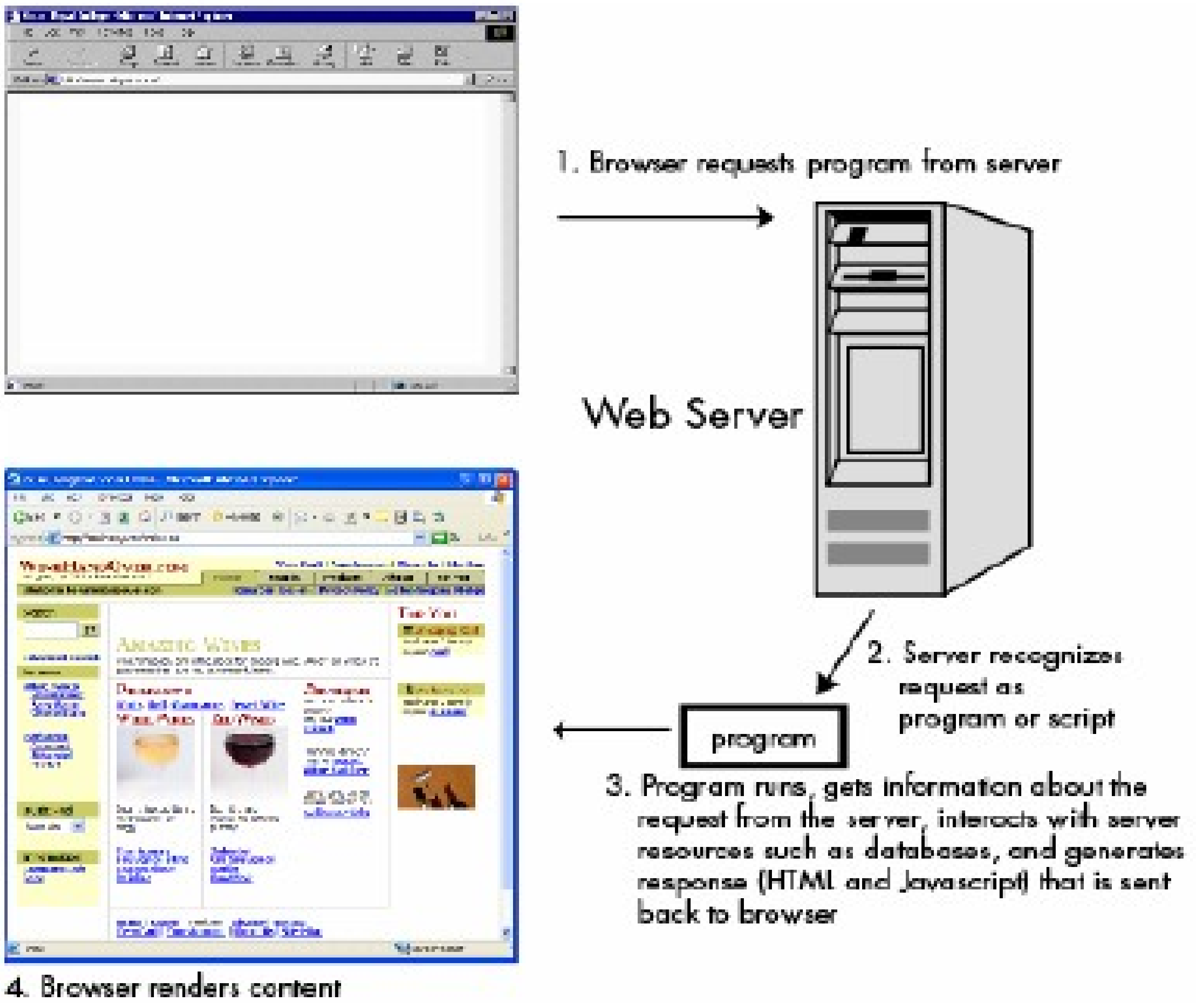
STATIC WEB CONTENT


1. Browser requests index.htm from server
Web Server
2. Server responds by sending content of index.htm to browser
Experience
Mount Royal College
3. Browser renders (displays) requested content

Dynamic Web Pages

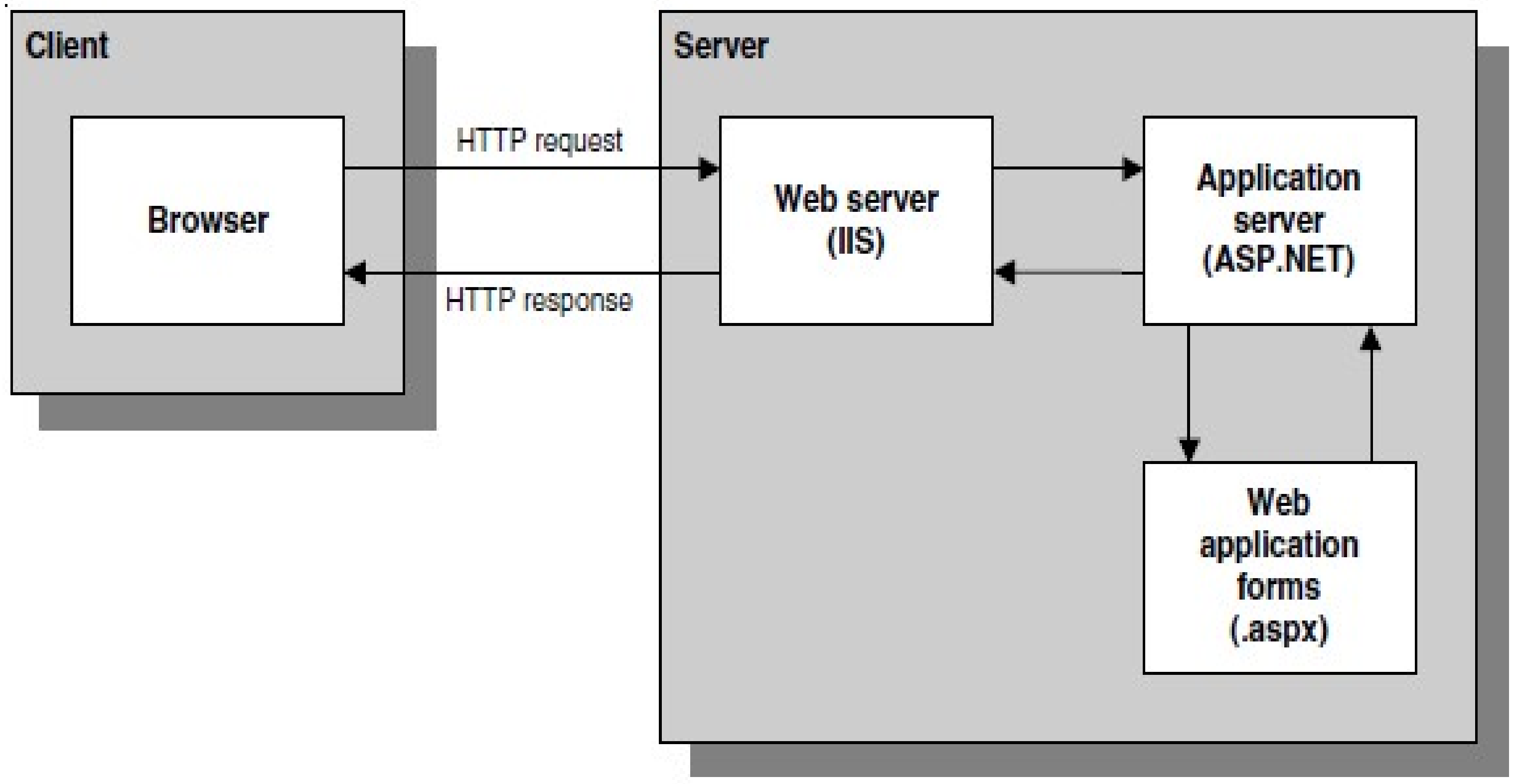
- Dynamic web pages are web forms that contain one or more server controls such as labels, text boxes and buttons.
- Brower sends a HTTP request to the web server.
- Web server (IIS) determines the type of page as dynamic page and passes the request to the application server (ASP.NET). ASP.NET manages the execution of the web form that is requested. ASP.NET creates html page and sends back to the browser as a ‘http reply’.
- Browser displays the page. The user resubmit the page by selecting controls. This is called ‘postback’.

Dynamic Web content


1. Browser requests program from server
Web Server
2. Server recognizes request as program or script
program
3. Program runs, gets information about the request from the server, interacts with server resources such as databases, and generates response (HTML and Javascript) that is sent back to browser
4. Browser renders content

Dynamic Web Pages

How a web server processes dynamic pages



The URL for an ASP.NET web page

```
http://msdn.microsoft.com/vs2005/default.aspx
```

ASP.NET WEB APPLICATION

An ASP.NET web application :

Consists of any number of web pages, controls, programming classes, web services, and other files Residing within a single web server application directory

The principal component of an ASP.NET web application are its web pages.

These are text files with an .aspx extension and are called **web forms.**

Consist of two parts:

- The declaratively-defined (i.e., by markup/tags) visual elements.
- The programming logic.

Server side vs Client side Programming

- In server side programming, all code runs on the server.

  When the ASP.NET code finishes running, the web server sends the user the final result—an ordinary HTML page that can be viewed in any browser.

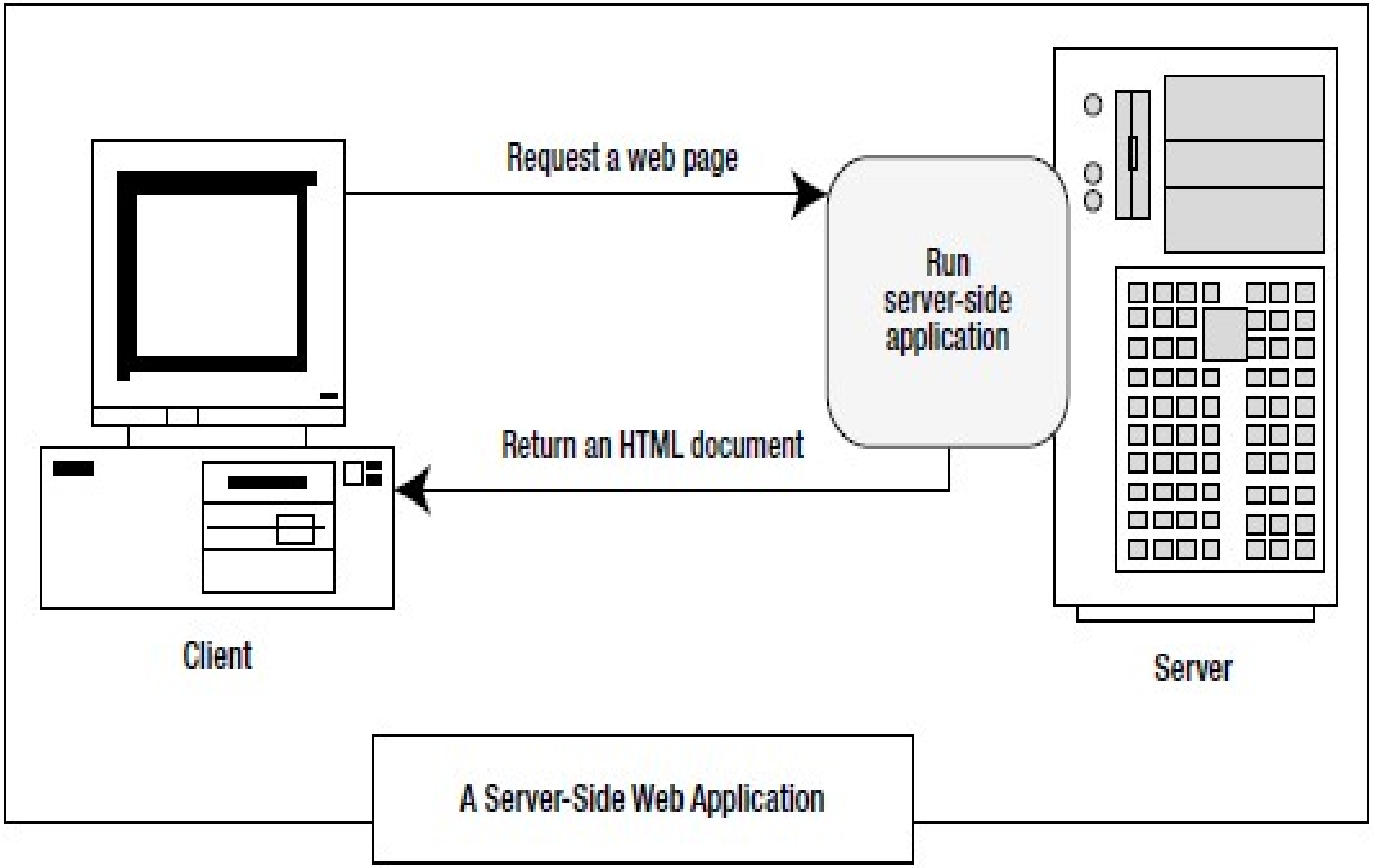
ASP.NET uses server side programming:

Reasons:

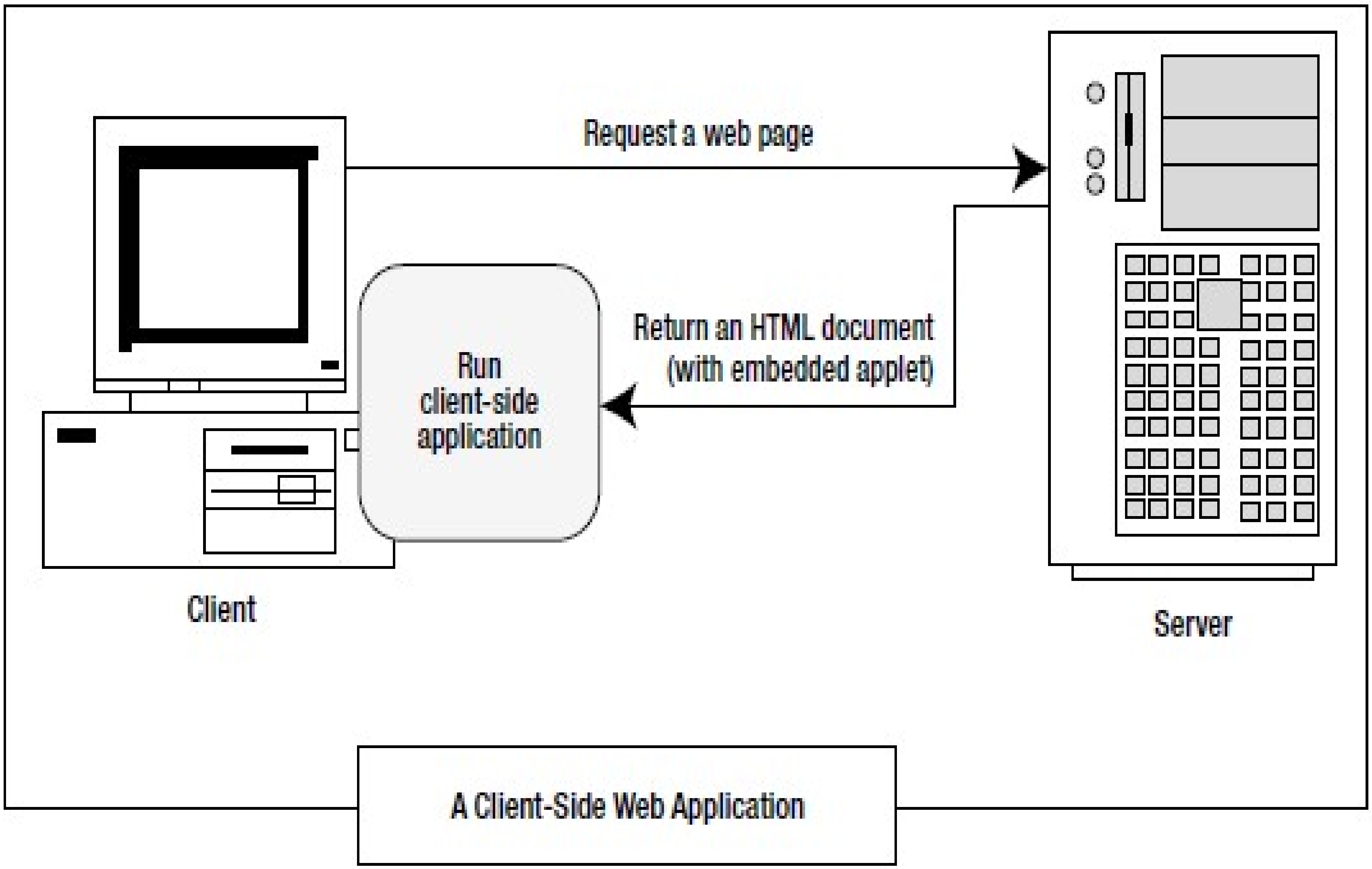
- Isolation: Client-side code can't access server-side resources.
- Security: end users can view the client-side code. He may tamper with it.
- Thin Clients: Smart phones, tablets may not support client side programming platforms such as silverlight, Flash.

- **In Client side programming**, a javascript code /any scripting language code is embedded in html page and sent to the browser.
  - browser downloads the code and executes it locally.

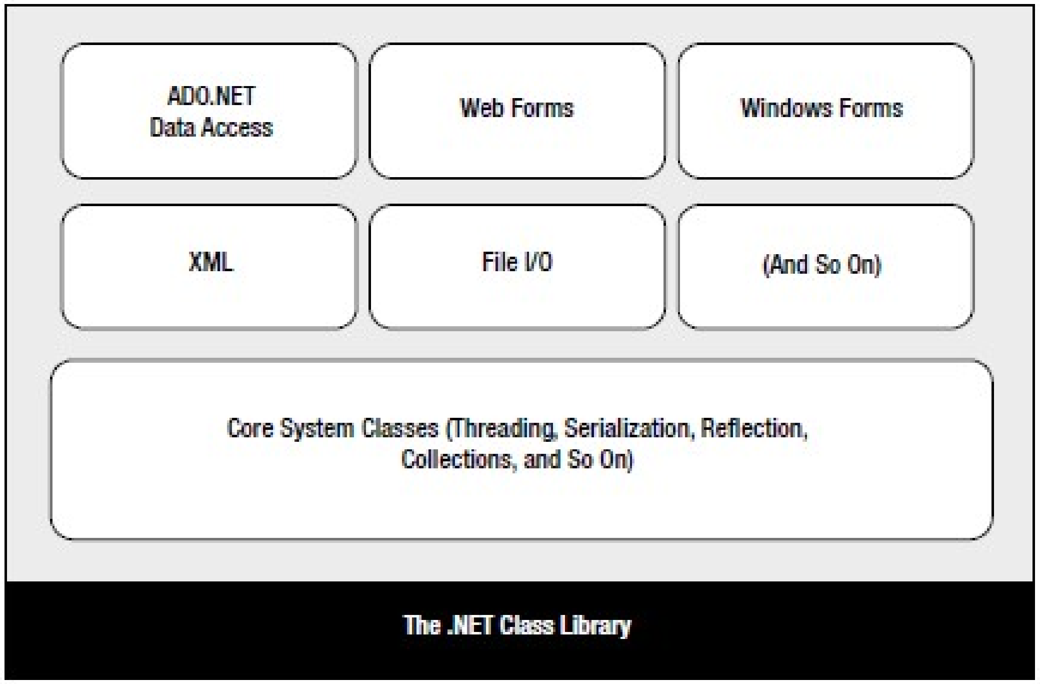
Server side Programming

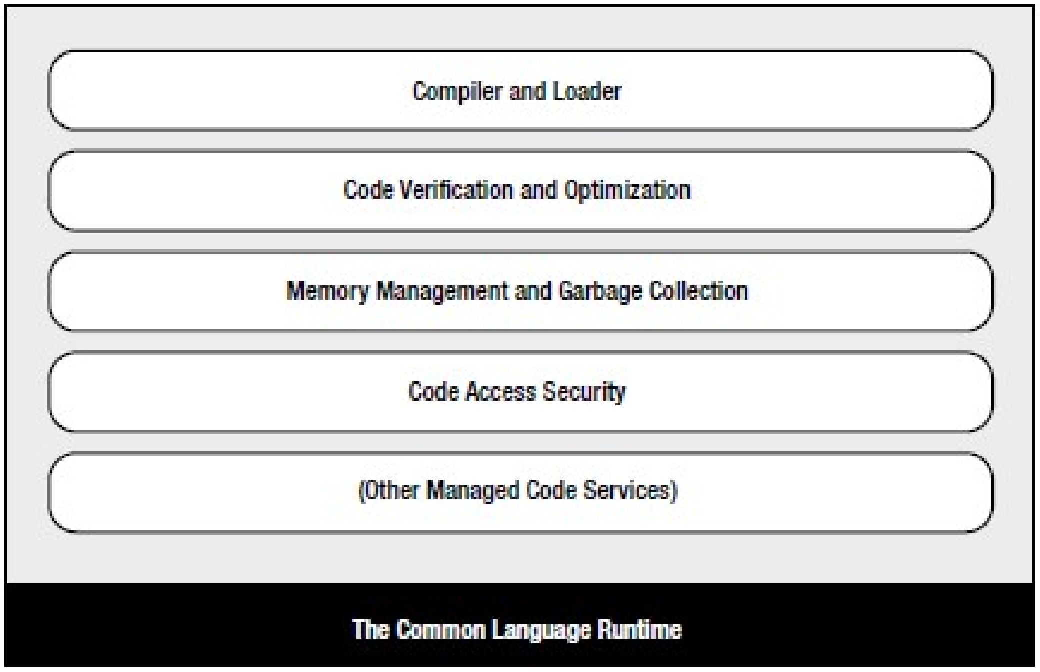


Client side Programming



Request a web page
Return an HTML document (with embedded applet)
Run client-side application
Client
Server
A Client-Side Web Application




Compiler and Loader
Code Verification and Optimization
Memory Management and Garbage Collection
Code Access Security
(Other Managed Code Services)
The Common Language Runtime

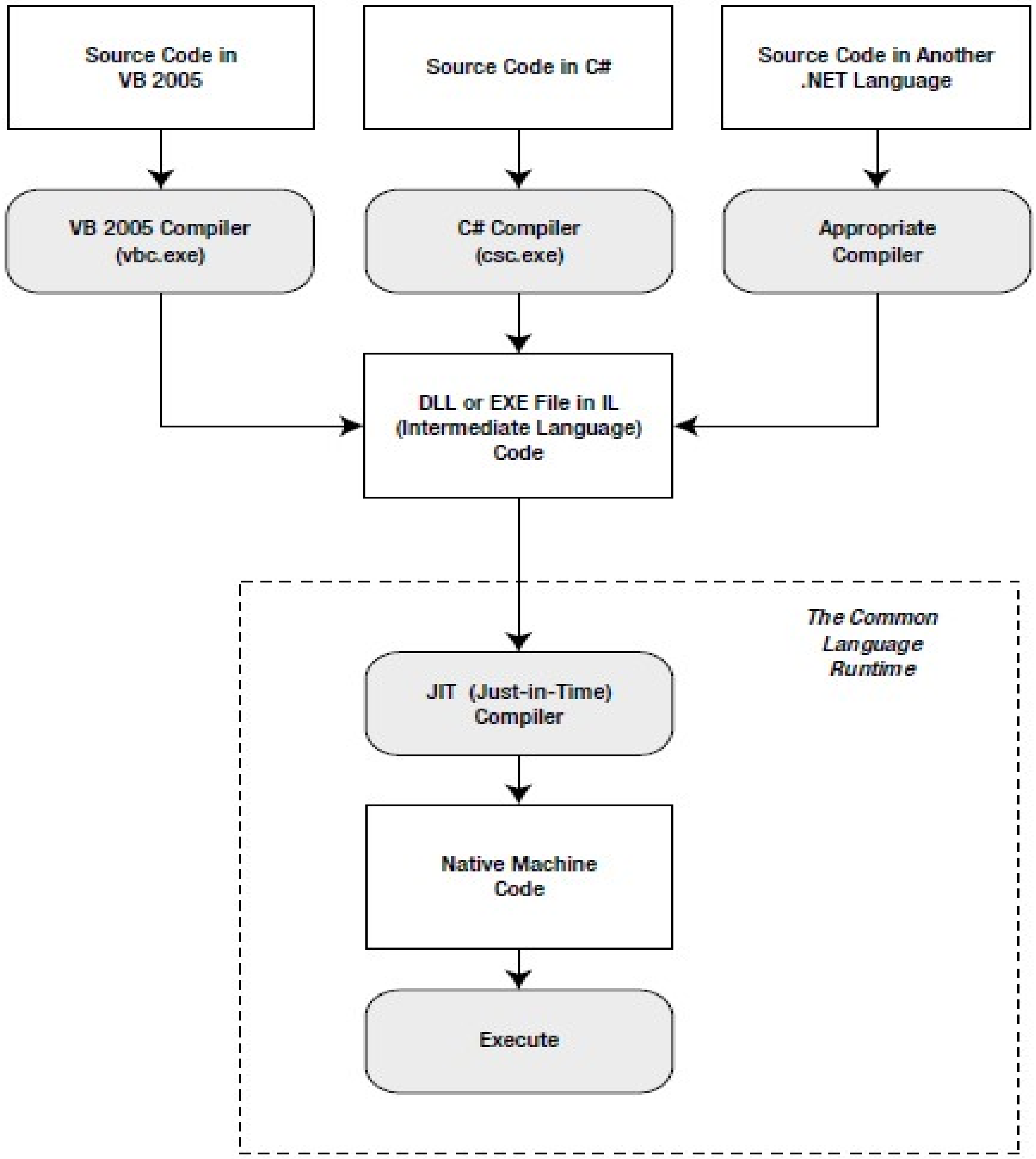
.NET framework

.NET FRAMEWORK

the .NET Framework is really a cluster of several technologies, which **contains:**

- *The .NET languages*
- *The Common Language Runtime (CLR)*
- *The .NET Framework class library*
- *ASP.NET*

LANGUAGE COMPILATION IN .NET


Source Code in VB 2005
Source Code in C#
Source Code in Another .NET Language
VB 2005 Compiler (vbc.exe)
C# Compiler (csc.exe)
Appropriate Compiler
DLL or EXE File in IL (Intermediate Language) Code
The Common Language Runtime
JIT (Just-in-Time) Compiler
Native Machine Code
Execute

CLR (COMMON LANGUAGE RUNTIME)

With CLR we can achieve:

- ✓ Deep language integration → for CLR there is no distinction between the C# code, VB etc..
- ✓ Side-by-side execution
- ✓ Fewer errors

Drawbacks:

- ✓ Performance → we can overcome performance bottle necks with compiled page code & caching them.
- ✓ Code tranparency → IL code is easy to disassemble

VISUAL STUDIO (CHAPTER 4)

Benefits:

- ✓ *Integrated error checking*
  - *Shows data type conversion errors, missing namespaces or classes, undefined variables. As you type, errors are displayed in error list window.*
- ✓ *The web form designer* → *You can design the web page by drag & dropping controls on to form and configure their properties. Code markup is added automatically in .aspx*
- ✓ *An integrated web server* → *IIS (Internet Information Services)*
- ✓ *Developer productivity enhancements* → *Collapsible code display, automatic stmt. Completion, color coded syntax, intelli sense etc..*
- ✓ *Fine-grained debugging* → *run to cursor, pause at any point, watch any variable value, single stepping, stepin ,stepout etc…*
- ✓ *Complete extensibility* → *creating custom controls, macros, project templates, using third party tools etc…*

VISUAL STUDIO

Features of Visual Studio:

- ✓ Page Design
- ✓ Automatic error detection
- ✓ Debugging Tools
- ✓ IntelliSense

Anatomy of ASP.NET applications –

ASP.NET file types, web folders

ASP.NET File types

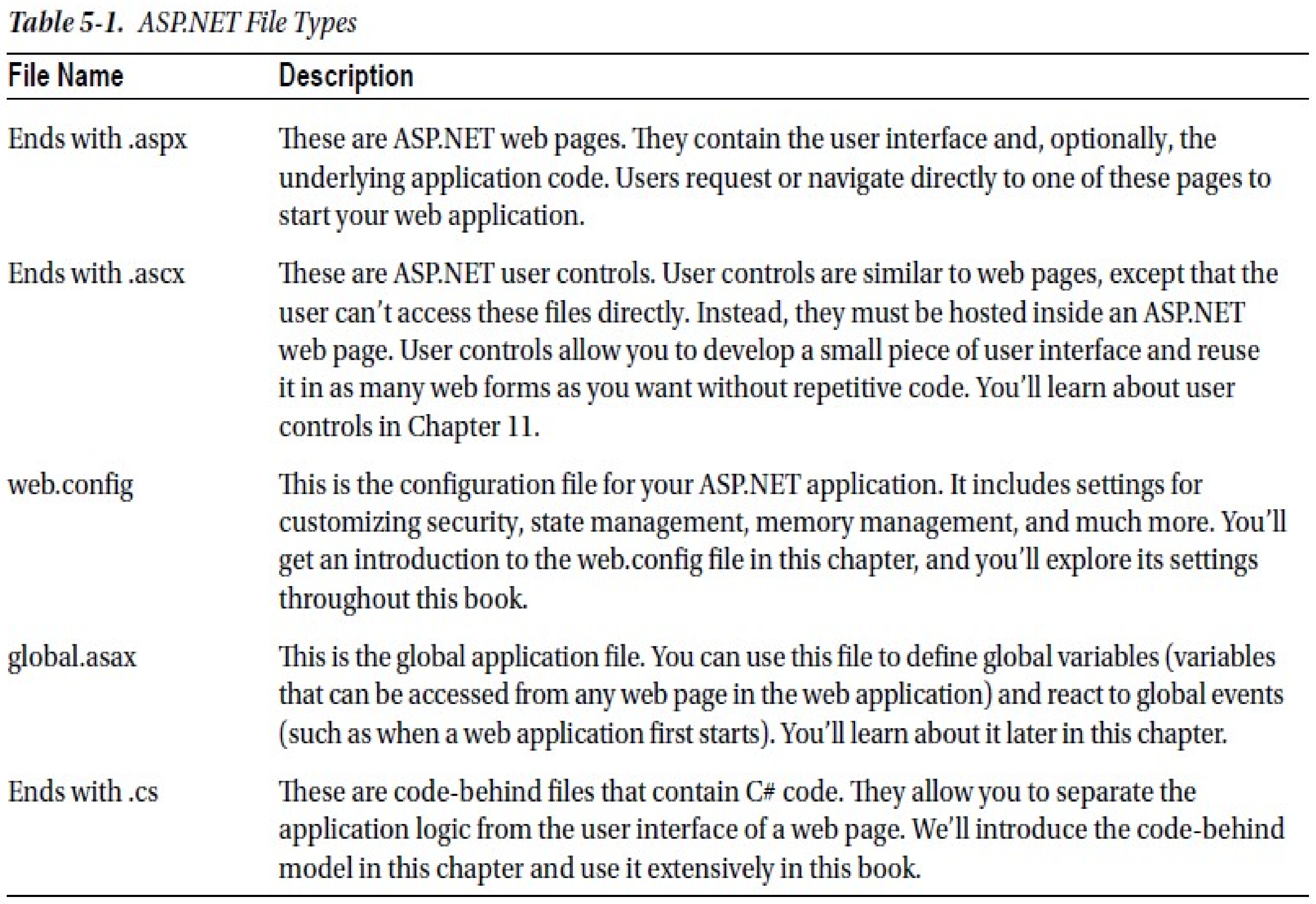


Table 5-1. ASP.NET File Types

File Name	Description
Ends with .aspx	These are ASP.NET web pages. They contain the user interface and, optionally, the underlying application code. Users request or navigate directly to one of these pages to start your web application.
Ends with .ascx	These are ASP.NET user controls. User controls are similar to web pages, except that the user can't access these files directly. Instead, they must be hosted inside an ASP.NET web page. User controls allow you to develop a small piece of user interface and reuse it in as many web forms as you want without repetitive code. You'll learn about user controls in Chapter 11.
web.config	This is the configuration file for your ASP.NET application. It includes settings for customizing security, state management, memory management, and much more. You'll get an introduction to the web.config file in this chapter, and you'll explore its settings throughout this book.
global.asax	This is the global application file. You can use this file to define global variables (variables that can be accessed from any web page in the web application) and react to global events (such as when a web application first starts). You'll learn about it later in this chapter.
Ends with .cs	These are code-behind files that contain C# code. They allow you to separate the application logic from the user interface of a web page. We'll introduce the code-behind model in this chapter and use it extensively in this book.

ASP.NET File types

Table 5-2. ASP.NET Folders

Directory	Description
App_Browsers	Contains .browser files that ASP.NET uses to identify the browsers that are using your application and determine their capabilities. Usually, browser information is standardized across the entire web server, and you don't need to use this folder. For more information about ASP.NET's browser support—which is an advanced feature that most ordinary web developers can safely ignore—refer to *Pro ASP.NET 4.5 in C#* (Apress, 2012).
App_Code	Contains source code files that are dynamically compiled for use in your application.
App_GlobalResources	Stores global resources that are accessible to every page in the web application. This directory is used in localization scenarios, when you need to have a website in more than one language. Localization isn't covered in this book, and you can refer to *Pro ASP.NET 4.5 in C#* for more information.
App_LocalResources	Serves the same purpose as App_GlobalResources, except these resources are accessible to a specific page only.
App_WebReferences	Stores references to web services, which are remote code routines that a web application can call over a network or the Internet.
App_Data	Stores data, including SQL Server Express database files (as you'll see in Chapter 14). Of course, you're free to store data files in other directories.
App_Themes	Stores the themes that are used to standardize and reuse formatting in your web application. You'll learn about themes in Chapter 12.
Bin	Contains all the compiled .NET components (DLLs) that the ASP.NET web application uses. For example, if you develop a custom component for accessing a database (see Chapter 22), you'll place the component here. ASP.NET will automatically detect the assembly, and any page in the web application will be able to use it.

ASP.NET application folders

- ✓ \App_Code → is for storing class files, .wsdl files(web service), and typed datasets (xsd).
  - ✓ The files in this folder are automatically compiled by IDE and makes them available to all the web pages of application.
- ✓ \App_Data → contains data stores of your application. .mdf files, xml files etc … can be kept here.
- ✓ \App_Themes folder → this contains .skin, .css files and images. Used to provide a common look and feel to your site.
- ✓ \App_GlobalResources → contains.resx files containing different language string tables etc..
  - ✓ Files placed in this folder are available to all web pages as resources.
- ✓ \App_WebReferences → Allows to access remote web services automatically from your application.
- ✓ \Web_Browsers → contains .browser files, which are xml files to identity the browsers making requests to the application and their capabilities.

\App_LocalResources → these resources are accessible to a specific page only.

\Bin → Contains all the compiled .NET components (DLLs) that the ASP.NET web application uses.

Web form programming logic

A web form's programming logic can exist in either:

- ✓ The same file as the visual elements i.e., the .aspx file. This code is contained within a **code declaration block.**
- ✓ In a separate class file. The file is usually called a code-behind file.

By convention, its filename is same as .aspx file but with a language extension.

HelloWorld.aspx	--	web form
HelloWorld.aspx.cs	--	code-behind file

Sample web form

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="HelloWorldCodeBehind.aspx.cs"
    Inherits="HelloWorldCodeBehind" %>

<!DOCTYPE … >

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Hello World Code-</title>
</head>
<body>
    <form id="form1" runat="server" >
    <h1>Hello World</h1>
    The date is <em>
    <asp:Label ID="myDate" runat="server"></asp:Label>
    </em>
    </form>
</body>
</html>
```

Page directive

HelloWorldCodeBehind.aspx

HelloWorldCodeBehind.aspx.cs

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

public partial class HelloWorldCodeBehind : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        myDate.Text = DateTime.Now.Date.ToString();
    }
}
```

Why use code-behind?

- ✓ The real advantage of separating the code into its own file is that it may lead to more maintainable web forms.

- ✓ One of the main benefits of ASP.NET is that a page's programming logic can be **conceptually separated from the** Presentation by using a code-behind file a page's programming logic can also be **physically** separated from the presentation/ markup.

- ✓ By placing the programming code into its own file, it is also potentially easier to make use of a division of labor in the creation of the site.

Designing Web Page

- ✓ viewing of .aspx can be done in 3 ways:
  1. Design View → Graphical representation of what you see.
  2. Source View → view with html markup, ASP.NET controls tags.
  3. Split View → Page view with both Design view & source view at once.

Using Properties window

✓

Web Form

Web form contains:

1. Page Directive (line 1 & 2)

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

   a. Code Language → C#
   b. Code behind file name → .aspx.cs file name
   c. Name of the Page Class → page class Name

2. The Doc type (line 3) → The doctype indicates the type of mark up (for example, HTML5 or XHTML) that you're using to create your web page

   Most web pages use **HTML5** doctype

   ```
   <!DOCTYPE html>
   ```

HTML basics – Basic HTML elements

Tag	Name	Type	Description
<b>, <i>, <u>	Bold, Italic, Underline	Container	Used for making text bold, underlining, italic.
<p>	Paragraph	Container	Paragraphing a text
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	Heading	Container	Sets text bold & font size
<img>	Image	Stand-alone	To display image
<br>	Line Break	Stand-alone	Gives a single line break
<hr>	Horizontal Line	Stand-alone	Gets a single horizontal line
<a>	Anchor	Container	Gets a hyperlink for the text
<ul>, <li>	Unordered List, List Item	Container	Used for Bulleted List items
<ol>, <li>	Ordered List, List Item	Container	Used for Numbered List items.
<table>, <tr>, <td>, <th>	Table	Container	Table, table row, table cell & table header

HTML basics – Basic HTML elements

Tag	Name	Type	Description
<div>	Division	Container	Container for other elements
<form>	Form	Container	Used to hold all the controls on a web page
<span>	Span	Container	all-purpose container for bits of text content inside other elements. Used for formatting (changing color etc..) some portion of text.

A Complete Web Page

Basic Web Page:

```
<html>
    <head runat = "server">
      <title > Untitled Page</title>
    </head>
    <body>
         <form ID = "form1" runat = "server">
                <div>
                </div>
          </form>
    </body>
</html>
```

Writing Code

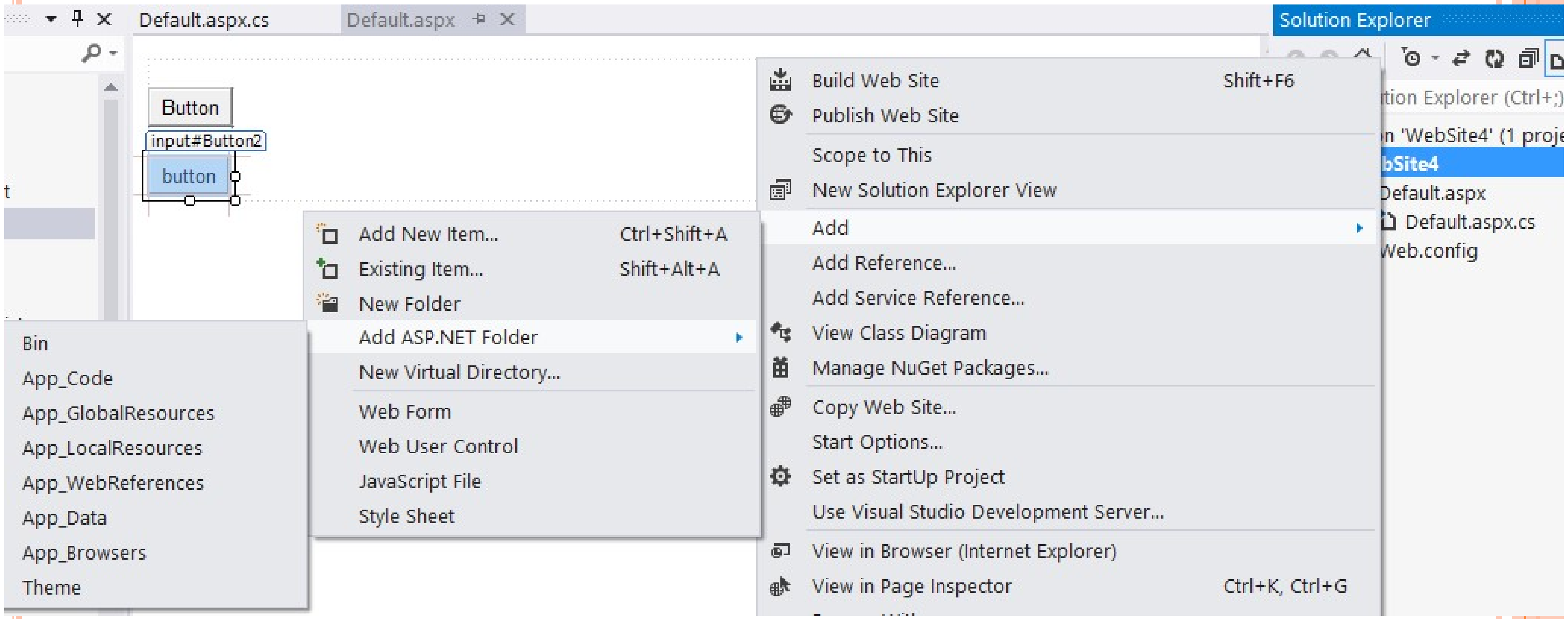
1. Using Code behind class
2. Adding Event Handlers
3. Outlining
4. IntelliSense
   - ✓ Using Member List
   - ✓ Caching Errors in code
   - ✓ Caching Errors in Markup
   - ✓ Automatically importing namespaces
   - ✓ Formatting and Coloring code

Adding Event Handlers

Three Ways of Adding Event Handlers:

1. Type event handler manually in Page Class.
2. Double click a control in Design View
3. *Choose the event from the Properties window.*

Sample web form



Server Controls

HTML Server Controls & ASP.NET Server Controls

Server Controls

- *Two sets of server-side controls that you can incorporate into your web*

forms. These two types of controls play subtly different roles:

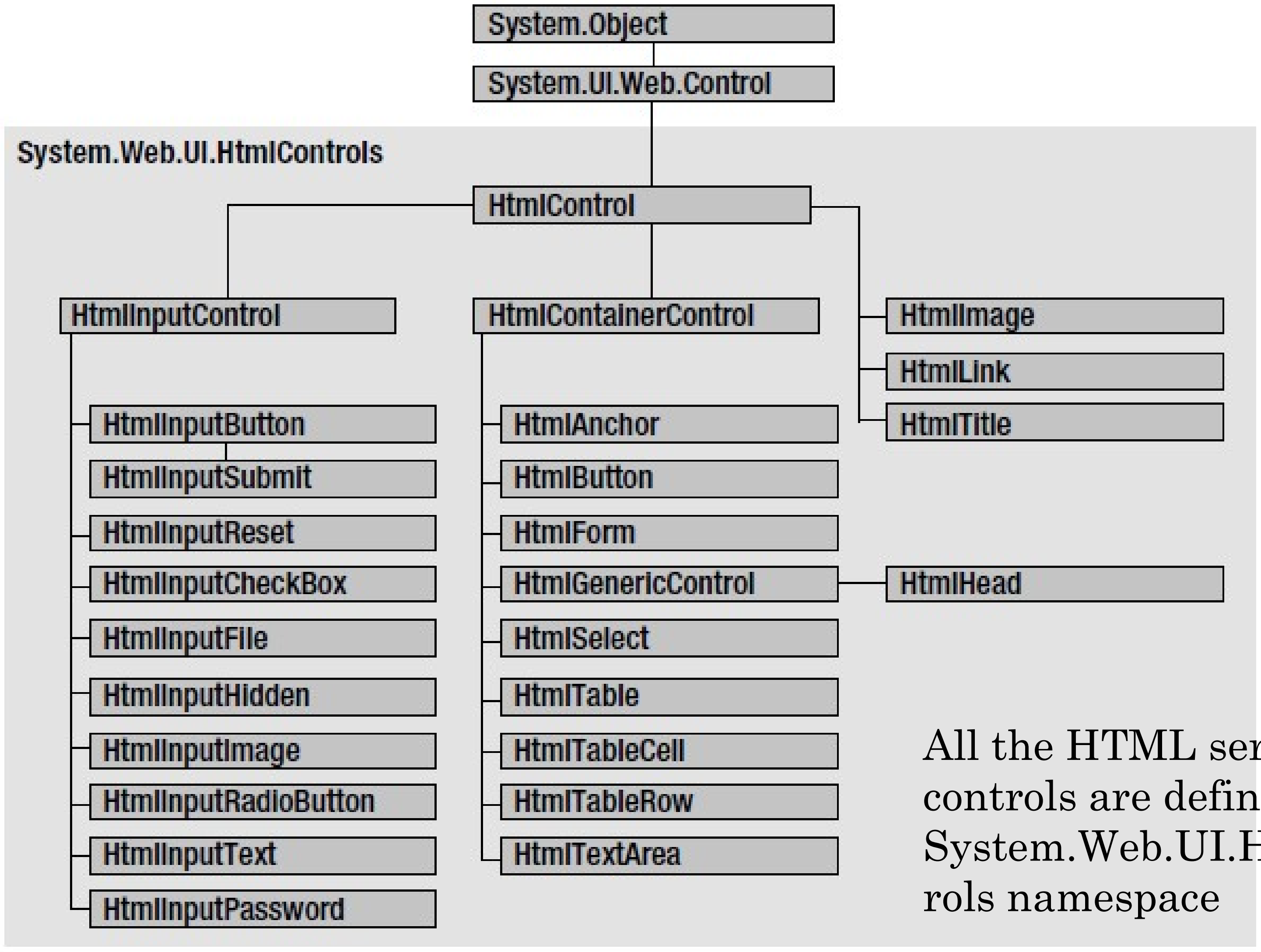
- ***HTML server controls:*** *These are server-based equivalents for standard HTML elements.* These controls are ideal if you're a seasoned web programmer who prefers to work with familiar HTML tags (at least at first). They are also useful when migrating ordinary HTML pages or classic ASP pages to ASP.NET, because they require the fewest changes.
- ***Web controls:*** *These are similar to the HTML server controls, but they provide a* richer object model with a variety of properties for style and formatting details. They also provide more events and more closely resemble the controls used for Windows development. Web controls also feature some user interface elements that have no direct HTML equivalent, such as the GridView, Calendar, and validation controls.

HTML Server Controls

- HTML server controls provide an object interface for standard HTML elements. They provide three key features:

  They generate their own interface: *You set properties in code, and the underlying HTML* tag is created automatically when the page is rendered and sent to the client.
- ***They retain their state:*** *Because the Web is stateless, ordinary web pages need to do* a lot of work to store information between requests. HTML server controls handle this task automatically. For example, if the user selects an item in a list box, that item remains selected the next time the page is generated. Or if your code changes the text in a button, the new text sticks the next time the page is posted back to the web server.
- ***They fire server-side events:*** *For example, buttons fire an event when clicked, text boxes* fire an event when the text they contain is modified, and so on. Your code can respond to these events, just like ordinary controls in a Windows application. If a given event doesn't occur, the event handler won't be executed.

HTML SERVER CONTROL CLASSES



All the HTML server controls are defined in the System.Web.UI.HtmlControls namespace

HTML Server Controls

Class Name	HTML Element
HtmlForm	<form>
HtmlAnchor	<a>
HtmlImage	<img>
HtmlTable, HtmlTableRow, and HtmlTableCell	<table>, <tr>, <th>, and <td>
HtmlInputButton, HtmlInputSubmit, and HtmlInputReset	<input type="button">, <input type="submit">, and <input type="reset">
HtmlButton	<button>
HtmlInputCheckBox	<input type="checkbox">
HtmlInputRadioButton	<input type="radio">
HtmlInputText and HtmlInputPassword	<input type="text"> and <input type="password">
HtmlTextArea	<textarea>
HtmlInputImage	<input type="image">
HtmlInputFile	<input type="file">

HTML Server Controls

Class Name	HTML Element
HtmlInputHidden	<input type="hidden">
HtmlSelect	<select>
HtmlHead and HtmlTitle	<head> and <title>
HtmlGenericControl	Any other HTML Element.

HtmlControl class

Properties:

Properties	Description
Attributes	Provides a collection of all the attributes that are set in the control tag, and their values
Controls	Provides a collection of all the controls contained inside the current control
Disabled	Disables the control when set to true.
EnableViewState	If this is set to true (the default), the control stores its state in a hidden input field on the page, thereby ensuring that any changes you make in code are remembered.
Page	Provides a reference to the web page that contains this control as a System.Web.UI.Page object.
Parent	Provides a reference to the control that contains this control.
Style	Provides a collection of CSS style properties that can be used to format the control.
TagName	Indicates the name of the underlying HTML element.
Visible	Hides the control when set to false and will not be rendered to the final HTML page that is sent to the client.

properties

Control	Properties
HtmlAnchor	HRef, Target
HtmlImage	Src, Alt, Width, Height
HtmlInputCheckBox and HtmlInputRadioButton	Checked
HtmlInputText	Value
HtmlTextArea	Value
HtmlInputImage	Src, Alt
HtmlSelect	Items (collection)
HtmlGenericControl	InnerText and InnerHtml

HTML CONTROLS

HTML Controls:

Input (Button)

Input (Reset)

Input (Submit)

Input (Text)

Input (File)

Input (Password)

Input (Checkbox)

Input (Radio)

TextArea

Table

Image

Select

HTML Control events

HTML server controls also provide one of two possible events:

1. **ServerClick** → A Click event processed on the server

   Ex controls: HtmlAnchor, HtmlButton, HtmlInputButton, HtmlInputImage, HtmlInputReset

2. **ServerChange** → Fires when a change has been made to a text or selection control.

   Ex controls: HtmlInputText, HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputHidden, HtmlSelect, HtmlTextArea

HtmlContainerControl class

InnerHtml → The HTML content between the opening and closing tags of the control. Nested tags can also be applied.

InnerText → The text content between the opening and closing tags of the control.

HtmlInputControl *class*

Properties:

Type → Provides the type of input control.

Value → Returns the contents of the control as a string.

Page class

Properties:

Property	Description
IsPostBack	This Boolean property indicates whether this is the first time the page is being run (false) or whether the page is being resubmitted in response to a control event
EnableViewState	When set to false, this overrides the enableViewState property of the contained controls.
Application	This collection holds information that’s shared between all users in your website.
Session	This collection holds information for a single user, so it can be used in different pages.
Cache	This collection allows you to store objects that are time-consuming to create so they can be reused in other pages or for other clients.
Request	This refers to an HttpRequest object that contains information about the current web request

Page class

Properties:

Property	Description
Response	This refers to an HttpResponse object that represents the response ASP.NET will send to the user’s browser
Server	This refers to an HttpServerUtility object.
User	If the user has been authenticated, this property will be initialized with user information.

Application events

Basic Application Events:

Method	Description
Application_Start()	Occurs when the application starts, which is the first time it receives a request from any user
Application_End()	Occurs when the application is shutting down, generally because the web server is being restarted
Application_BeginRequest()	Occurs with each request the application receives, just before the page code is executed.
Application_EndRequest()	Occurs with each request the application receives, just after the page code is executed.
Session_Start()	Occurs whenever a new user request is received and a session is started.
Session_End()	Occurs when a session times out or is programmatically ended. This event is raised only if you are using in-process session-state storage
Application_Error()	Occurs in response to an unhandled error.

Application events – global.asax file

- You can trace out some application level events such as request for a page.
- **global.asax** file contains event handlers for handling application level events.
- Each application can have only one global.asax file

Configuring ASP.NET Application

- web.config file setting are
  - ✓ easily accessible & replicable
  - ✓ settings are easy to edit & understand

web.config file structure:

```
<?xml version = "1.0" ?>
<configuration>
    <appSettings > . . .</appSettings>
    <connectionStrings > . . .</connectionStrings>
    <system.web > . . .</system.web>
</configuration>
```

web.config file

Once you build your application with debugging enabled, the web.config file contents will be: (initially debug=false).

```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application, please
    visit http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.6.1"/>
    <httpRuntime targetFramework="4.6.1"/>
  </system.web>
</configuration>
```

ASP.NET website administration tool

- To open the tool, select Website → ASP.NET Configuration.
- To make changes to the web.config, you can use WAT.

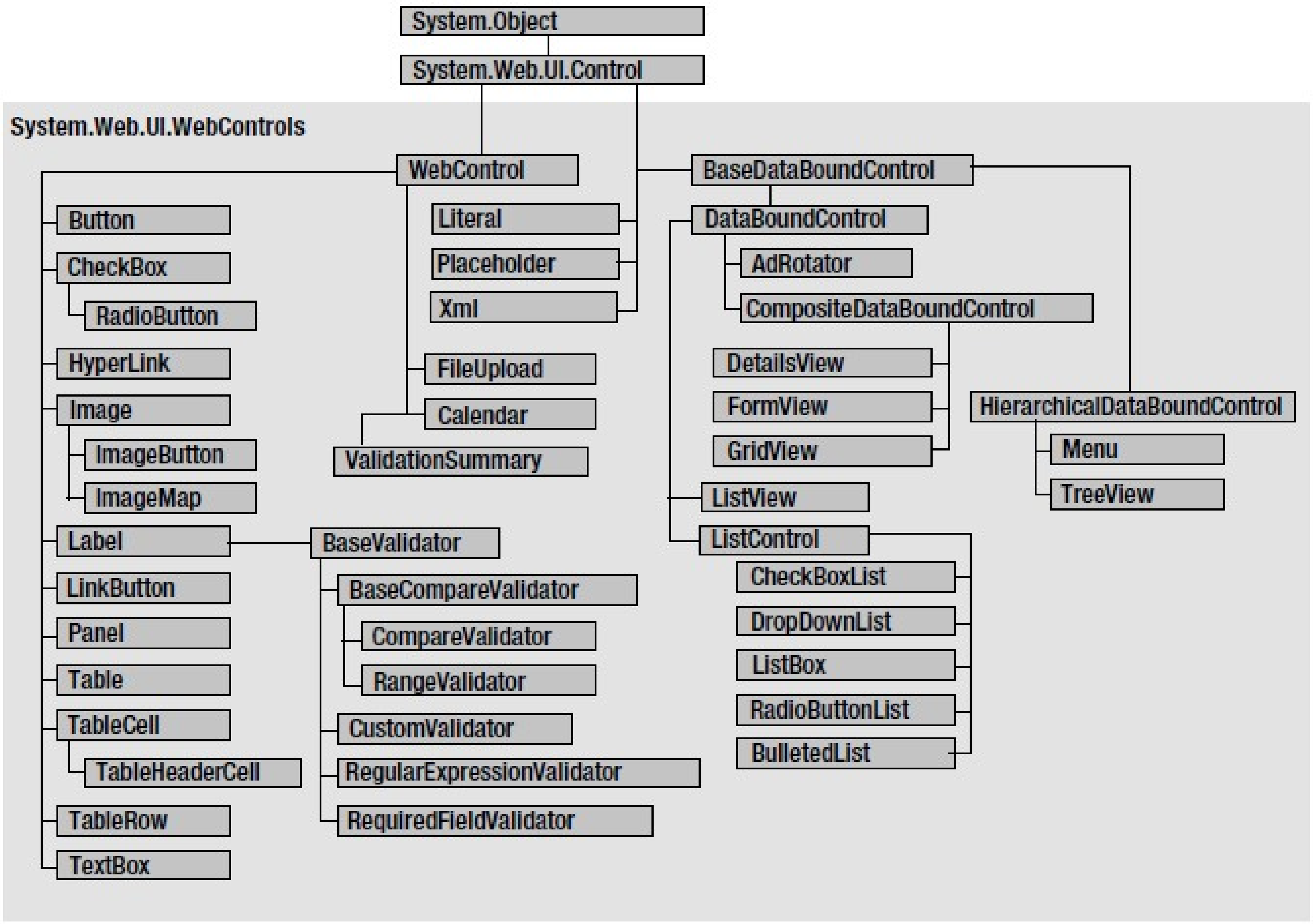
Server Controls

ASP.NET Web Server Controls

Web controls

- **They provide rich user interface**
  - A web control is programmed as an object but doesn’t necessarily correspond to a single element in the final HTML page. For example, you might create a single Calendar or GridView control, which will be rendered as dozens of HTML elements in the final page.
- **They provide a consistent object model**
  - HTML is full of quirks and idiosyncrasies. For example, a simple text box can appear as one of three elements, including <textarea>, <input type="text">, and <input type="password">. With web controls, these three elements are consolidated as a single TextBox control. Depending on the properties you set, the underlying HTML element that ASP.NET renders may differ.
- **They tailor their output automatically**
  - ASP.NET server controls can detect the type of browser and automatically adjust the HTML code they write to take advantage of features such as support for JavaScript.
- **They provide high-level features**
  - You’ll see that web controls allow you to access additional events, properties, and methods that don’t correspond directly to typical HTML controls.

Web controls



Web control classes

Control Class	Underlying HTML Element
Label	<span>
Button	<input type="submit"> or <input type="button">
TextBox	<input type="text">, <input type="password">, or <textarea>
CheckBox	<input type="checkbox">
RadioButton	<input type="radio">
Hyperlink	<a>
LinkButton	<a> with a contained <img> tag
ImageButton	<input type="image">
Image	<img>
ListBox	<select size="X"> where X is the number of rows that are visible at once
DropDownList	<select>
CheckBoxList	A list or <table> with multiple <input type="checkbox"> tags
RadioButtonList	A list or <table> with multiple <input type="radio"> tags
BulletedList	An <ol> ordered list (numbered) or <ul> unordered list (bulleted)
Panel	<div>
Table, TableRow, and TableCell	<table>, <tr>, and <td> or <th>

WEB CONTROLS

```
<asp:TextBox ID="txt" BackColor="Yellow" Text="Hello World"
ReadOnly="True" TextMode="MultiLine" Rows="5" runat="server" />
```

will be converted as the following HTML control:

```
<textarea name="txt" rows="5" cols="20" readonly="readonly"
   ID="txt“ style="background-color:Yellow;">Hello World</textarea>
```

WebControl class

Properties:

Property	Description
AccessKey	Specifies the keyboard shortcut as one letter.
BackColor, ForeColor, and BorderColor	Sets the colors used for the background, foreground, and border of the control
BorderWidth	Specifies the size of the control border
BorderStyle	Gets or sets the border style of the Web server control. Can be Dashed, Dotted, Double, Groove, Ridge, Inset, Outset, Solid, and None.
Enabled	When set to false, the control will be visible, but it will not be able to receive user input or focus.
EnableViewState	If this is set to true (the default), the control uses the hidden input field to store information about its properties, ensuring that any changes you make in code are remembered.
Font	Specifies the font used to render any text in the control.
Height and Width	Specifies the width and height of the control.

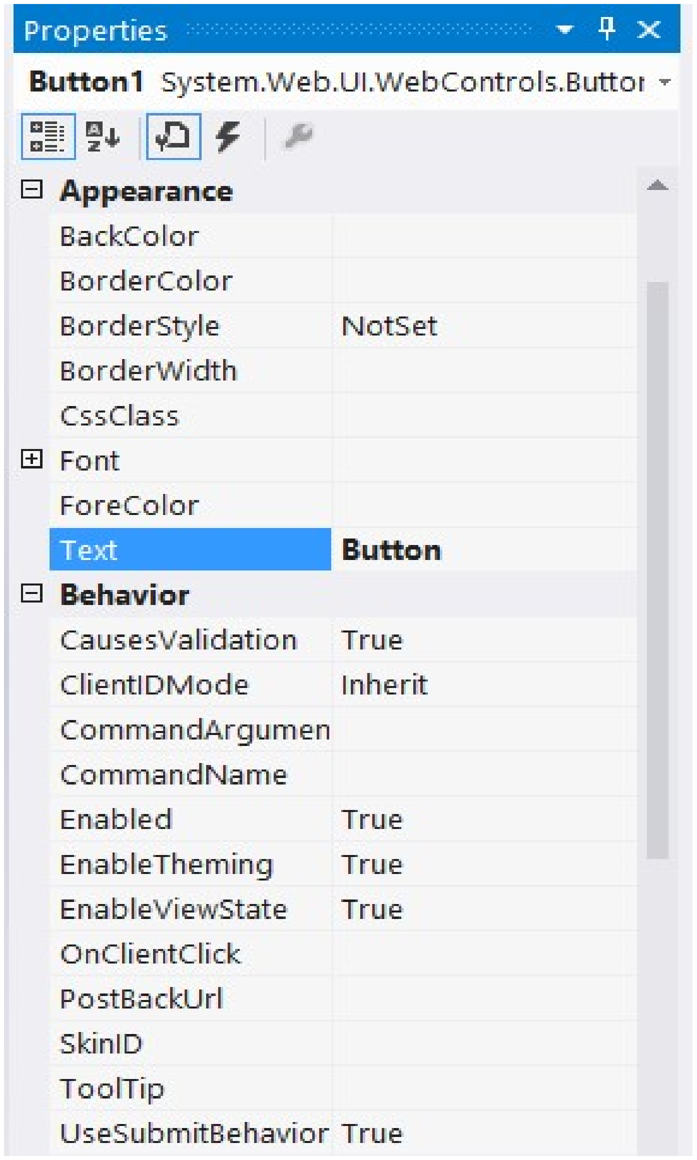
WebControl class

Properties:

	Description
ID	Specifies the name that you use to interact with the control in your code.
Page	Provides a reference to the web page that contains this control as a System.Web. UI.Page object
Parent	Provides a reference to the control that contains this control
TabIndex	A number that allows you to control the tab order. The control with a TabIndex of 0 has the focus when the page first loads.
ToolTip	Displays a text message when the user hovers the mouse above the control
Visible	When set to false, the control will be hidden and will not be rendered to the final HTML page that is sent to the client.

WEB CONTROL PROPERTIES

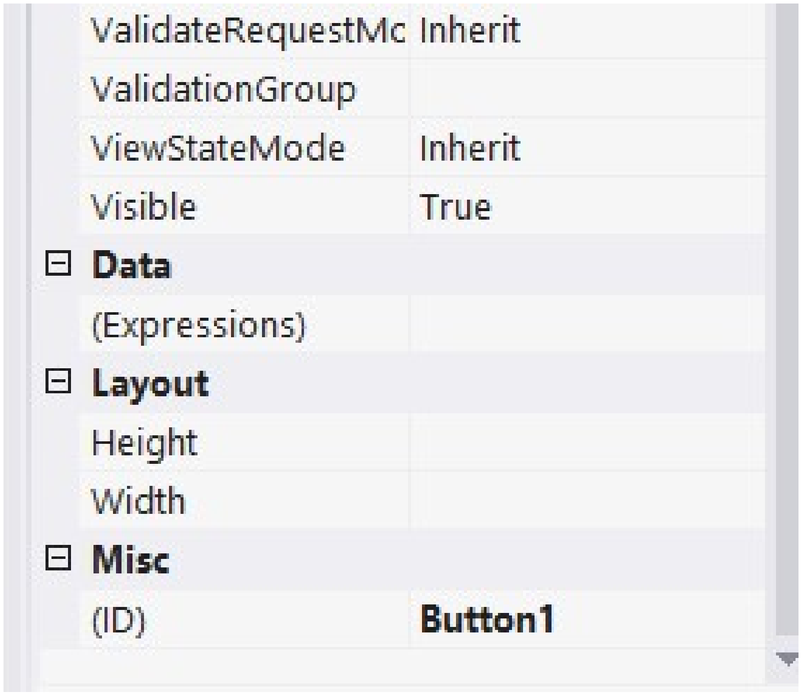
Button control properties:



Properties

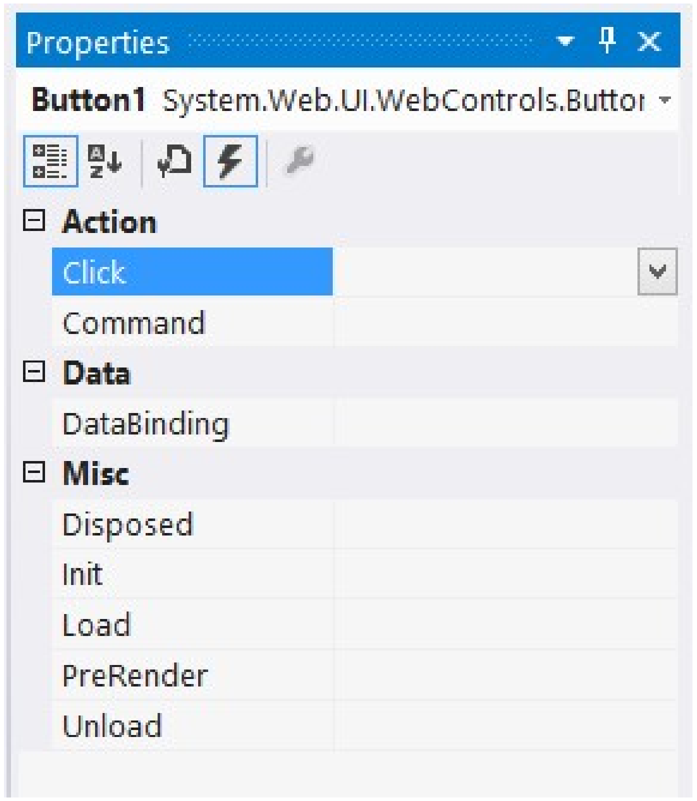
Button1 System.Web.UI.WebControls.Buttor

⊟ **Appearance**	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
CssClass	
⊞ Font	
ForeColor	
Text	**Button**
⊟ **Behavior**	
CausesValidation	True
ClientIDMode	Inherit
CommandArgumen	
CommandName	
Enabled	True
EnableTheming	True
EnableViewState	True
OnClientClick	
PostBackUrl	
SkinID	
ToolTip	
UseSubmitBehavior	True



ValidateRequestMc	Inherit
ValidationGroup	
ViewStateMode	Inherit
Visible	True
⊟ **Data**	
(Expressions)	
⊟ **Layout**	
Height	
Width	
⊟ **Misc**	
(ID)	**Button1**

Button control events



Properties

Button1 System.Web.UI.WebControls.Buttor

⊟ **Action**

Click	
Command	

⊟ **Data**

DataBinding	

⊟ **Misc**

Disposed	
Init	
Load	
PreRender	
Unload	

Web Server Controls

- **Html Server controls →** These are server-based equivalents for standard HTML elements.
- **Web Server Controls →** ASP.NET controls that provide a richer object model with a variety of properties for style and formatting details.
- **Custom and User Controls**

Standard Controls :

Label	Table
TextBox	BulletedList
Button	AdRotator
ImageButton	Xml
CheckBox	Calendar
RadioButton	
RadioButtonList	
Image	

ASP.NET CONTROLS

Data Controls :

GridView

DataList

DetailsView

SqlDataSource

AccessDataSource

XmlDataSource

ReportViewer

ASP.NET CONTROLS

Navigation Controls :

Menu

TreeView

Validation Conrols:

RequiredFieldValidator

RangeValidator

RegularExpressionValidator

CompareValidator

CustomValidator

ValidationSummary

ASP.NET CONTROLS

Authentication Controls:

Login

PasswordRecovery

LoginStatus

LoginName

CreateUserWizard

ChangePassword

ASP.NET CONTROLS

ASP.NET Web Server Controls :

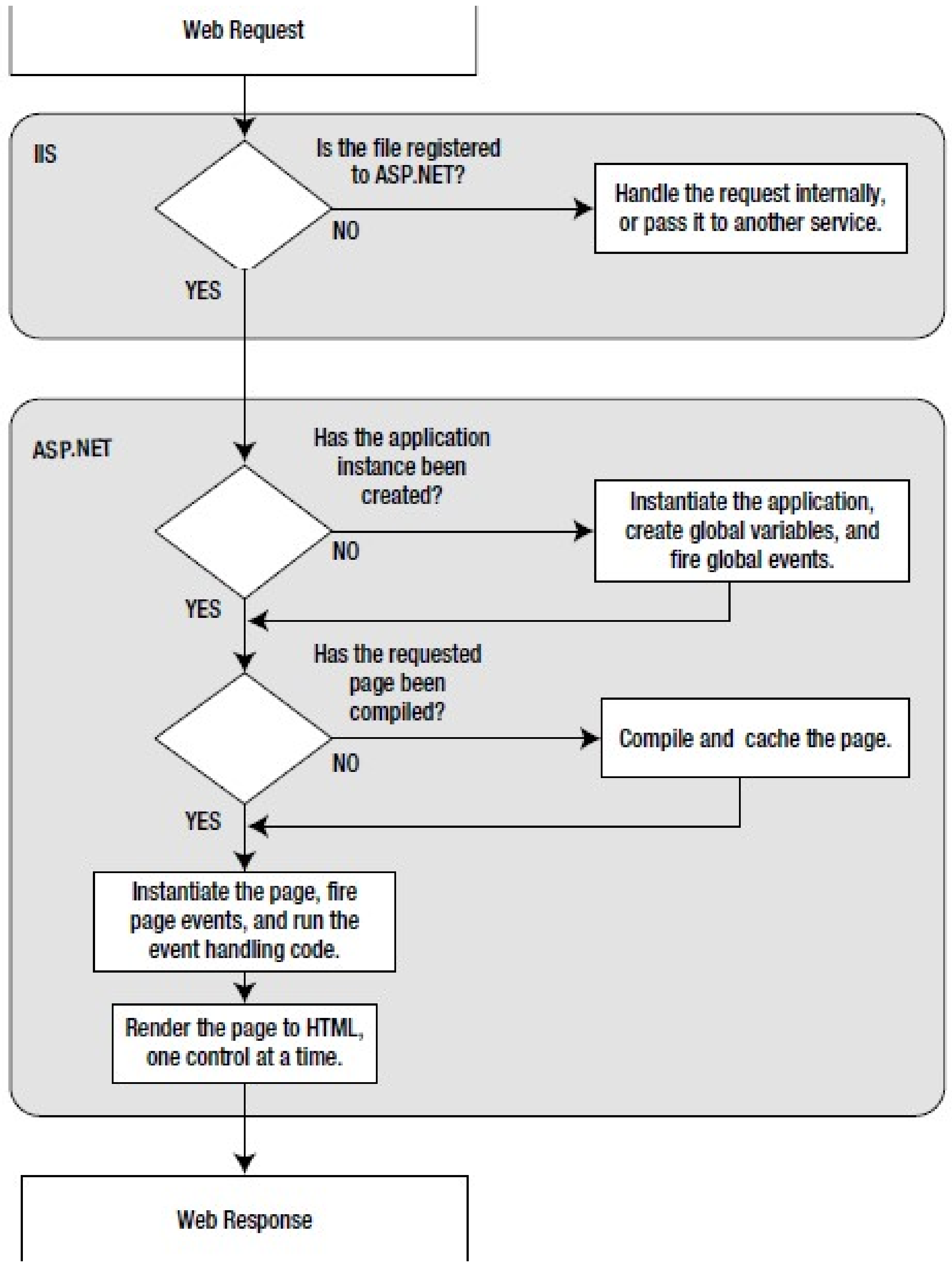
```
<asp:X runat="server" attribute="value">
Content
</asp:X>
```

Ex :

```
<asp:Label ID="Label1" runat="server"
  Text="Label"></asp:Label>
```

Text Box :

```
<asp:TextBox ID="TextBox1"
  runat="server"></asp:TextBox>
```


Web Request
IIS
Is the file registered to ASP.NET?
NO
Handle the request internally, or pass it to another service.
YES
ASP.NET
Has the application instance been created?
NO
Instantiate the application, create global variables, and fire global events.
YES
Has the requested page been compiled?
NO
Compile and cache the page.
YES
Instantiate the page, fire page events, and run the event handling code.
Render the page to HTML, one control at a time.
Web Response

The stages in an ASP.NET request

default.aspx FILE

```
<%@ Page Language="C#" AutoEventWireup="true"
   CodeFile="Default.aspx.cs" Inherits="_Default" %>
…
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
         <div>
         </div>
    </form>
</body>
</html>
```

default.aspx.cs FILE

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

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

CONTROL UNITS

- Unit structure to set Width, Height, BorderWidth etc..
- Units can be specified in no of pixels (px) or in percentage of browser window(%).
- <asp:Panel Height="300px" Width="50%" ID="pnl" runat="server" />
  - Sets Height of panel to 300 pixels and Width to 50% of browser window.
- Programmatically setting values:
  - pnl.Height = Unit.Pixel(300);
  - pnl.Width = Unit.Percentage(50);

OR

```
Unit myUnit = new Unit(300, UnitType.Pixel);
 pnl.Height = myUnit;
 pnl.Width = myUnit;
```

Colors

- The Color property refers to a Color object from the System.Drawing namespace. You can create color objects in several ways:
- ***Using an ARGB (alpha, red, green, blue) color value:*** *You specify each value as an* integer from 0 to 255. The alpha component represents the transparency of a color, and usually you'll use 255 to make the color completely opaque.
- ***Using a predefined .NET color name:*** *You choose the correspondingly named readonly* property from the Color structure. These properties include the 140 HTML color names.
- ***Using an HTML color name:*** *You specify this value as a string by using the* ColorTranslator class.
- **Example:**

using System.Drawing;

The following code shows several ways to specify a color in code:

```
// Create a color from an ARGB value
int alpha = 255, red = 0, green = 255, blue = 0;
ctrl.ForeColor = Color.FromArgb(alpha, red, green, blue);
// Create a color using a .NET name
ctrl.ForeColor = Color.Crimson;
// Create a color from an HTML code
ctrl.ForeColor = ColorTranslator.FromHtml("Blue");
```

When defining a color in the .aspx file, you can use any one of the known color names:

```
<asp:TextBox ForeColor="Red" Text="Test" ID="txt" runat="server" />
```

Colors

Focus, Default controls, Access Keys, Default Button

Focus: Unlike HTML server controls, every web control provides a Focus() method.

Button1.Focus() → sets input focus to control 'Button1'

Default controls:

- You can set a control that should always be focused by setting the DefaultFocus property of the <form> tag:

  <form **DefaultFocus**="TextBox2" runat="server">

- You can override the default focus by calling the Focus() method in your code.

Access Keys:

- If you set the AccessKey property of a TextBox to A, pressing Alt+A will switch focus to the TextBox.

<asp:Label AccessKey="2" AssociatedControlID="TextBox2" runat="server"

Text="TextBox2:" /> <asp:TextBox runat="server" ID="TextBox2" />

Default Button: The default button is the button that is "clicked" when the user presses the Enter key.

<form **DefaultButton**="cmdSubmit" runat="server">

Units, Enumerations, Colors, Fonts

Fonts: The Font property actually references a full FontInfo object, which is defined in the System.Web.UI.WebControls namespace.

FontInfo properties:

Name → Font Name

Names → An array of strings with font names, in the order of preference.

Size → The size of the font as a FontUnit object. This can represent an absolute or relative size.

Bold, Italic, Strikeout, Underline, and Overline → boolean value

POSTBACK VS NON-POSTBACK EVENTS

Postback	**Non-postback**
Button	BulletdList
Calendar	CheckBox
DataGrid	CheckBoxList
GridView	DropDownList
ImageButton	ListBox
LinkButton	RadioButtonList
Menu	RadioButton
Repeater	TextBox

AutoPostBack property:

IsPostBack → indicates whether the page isbeing loaded first time (IsPostBack =false) or not.

ASP.NET PAGE LIFE CYCLE EVENTS

- Request for Page
- Page Start -- IsPostBack is set/reset
- Initialization -- controls are instantiated,theme and skin is applied
- Load -- controls properties are restored
- Validation - validation of controls
- Postback Event Handling
- Render
- Unload

Web page view state

```
<input type = "hidden" name = "__VIEWSTATE"
ID = "__VIEWSTATE“ value = "dDw3NDg2NTI5MDg7Oz4. . ." />
```

BulletedList Properties

Property	Description
BulletStyle	Determines the type of list. Values are NotSet, Numbered, LowerAlpha, UpperAlpha, LowerRoman, UpperRoman, Disc, CustomImage.
BulletImageUrl	If the BulletStyle is set to CustomImage, this points to the image that is placed to the left of each item as a bullet.
FirstBulletNumber	In an ordered list, this sets the first value.
DisplayMode	Determines whether the text of each item is rendered as text (use Text, the default) or a hyperlink (use LinkButton or HyperLink).

ImageMap control

ImageMap control is used to create an image that contains clickable hotspot region. When user click on the region, the user is either sent to a URL or a sub program is called. When it is rendered on the page, it is implemented through <img /> HTML tag.

Properties	Description
ImageUrl	Url of image location.
AlternetText	Appears if image not loaded properly
Tooltip	Appears when on mouse over the image
ImageAlign	Used to align the Text beside image.
HotSpotMode	PostBack/Navigate .... When Navigate, the user is navigated to a different URL. In case of PostBack, the page is posted back to the server.
OnClick	Attach a server side event that fires after clicking on image when HostSpotMode is PostBack.
PostBackValue	You can access it in the server side click event through ImageMapEventArgs. (eg. e.PostBackValue)

View state

- ✓ Microsoft ASP.NET view state is the technique used by an ASP.NET Web page to persist changes to the state of a Web Form across postbacks.
- ✓ If you right clicked on the page, you can see an option 'View Source' witch will display the rendered page source.
- ✓ Rendered page source contains 2 hidden fields in 2 div sections of the form tag.

```
 <form ID = "form1" method = "post" action ="CurrencyConverter.aspx">
<div class = "aspNetHidden">
<input type = "hidden" name = "__VIEWSTATE" ID = "__VIEWSTATE"
value = "dDw3NDg2NTI5MDg7Oz4..." />
</div>
<div class = "aspNetHidden">
<input type = "hidden" name = "__EVENTVALIDATION" ID =
   "__EVENTVALIDATION"
value = "/wEWAwLr3rrOBgLr797..." />
</div>
</form>
```

TOPICS

- Page class
- Page navigation
- Asp.Net basic controls units, Enumerations
- Fonts, Colors, Focus, Default button
- List controls,
- Panel, Table control.

PAGE NAVIGATION

- <a href="Page2.aspx" runat="server">**goto page2**</a>
- When you clicked on ‘goto page2’ hyperlink, you will be redirected to a new page Page2.aspx

Redirecting to another page:

Response.Redirect( ) method:

```
Response.Redirect("newpage.aspx");
```

- When Redirect( ) method is executed, ASP.NET sends a redirect message back to the browser. Browser sends the redirecting page request to server.

Redirecting to another site:

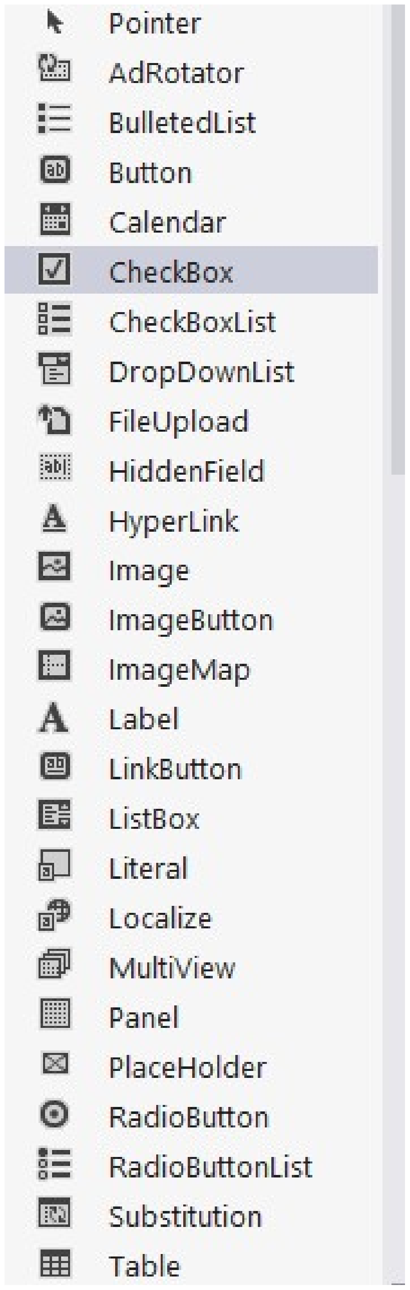
```
Response.Redirect("http://www.amazon.com");
```

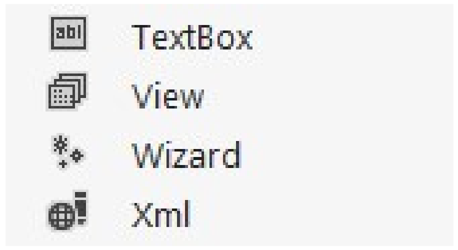
Redirecting to another page :

```
Server.Transfer("newpage.aspx");
```

- The advantage of using the Transfer() method is that it doesn’t involve the browser. Instead of sending a redirect message back to the browser, ASP.NET simply starts processing the new page as though the user had originally requested that page.
- This method wouldn’t allow to transer to another website or html page.

Basic Web server Controls


Pointer
AdRotator
BulletedList
Button
Calendar
CheckBox
CheckBoxList
DropDownList
FileUpload
HiddenField
HyperLink
Image
ImageButton
ImageMap
Label
LinkButton
ListBox
Literal
Localize
MultiView
Panel
PlaceHolder
RadioButton
RadioButtonList
Substitution
Table


TextBox
View
Wizard
Xml

ASP.NET SERVER CONTROLS

- Literal
- TextBox
- LinkButton
- ImageButton
- HyperLink
- DropDownList
- ListBox
- CheckBox
- CheckBoxList
- RadioButton
- RadioButtonList
- Image
- Table
- Calendar
- AdRotator
- Panel
- Xml
- PlaceHolder
- FileUpload
- ImageMap
- BulletedList
- View
- Wizard

ASP.NET PAGE PROCESSING EVENTS

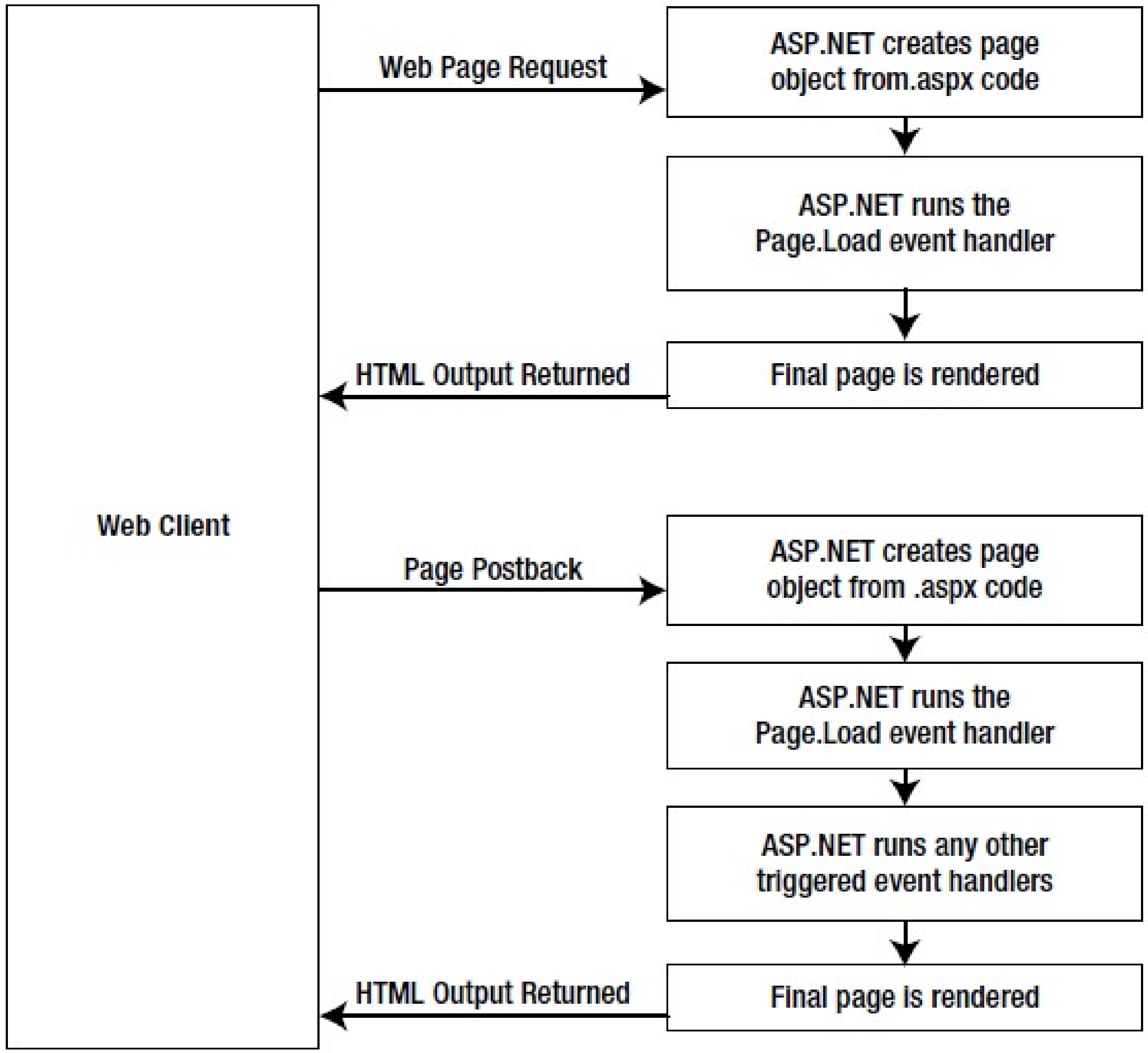
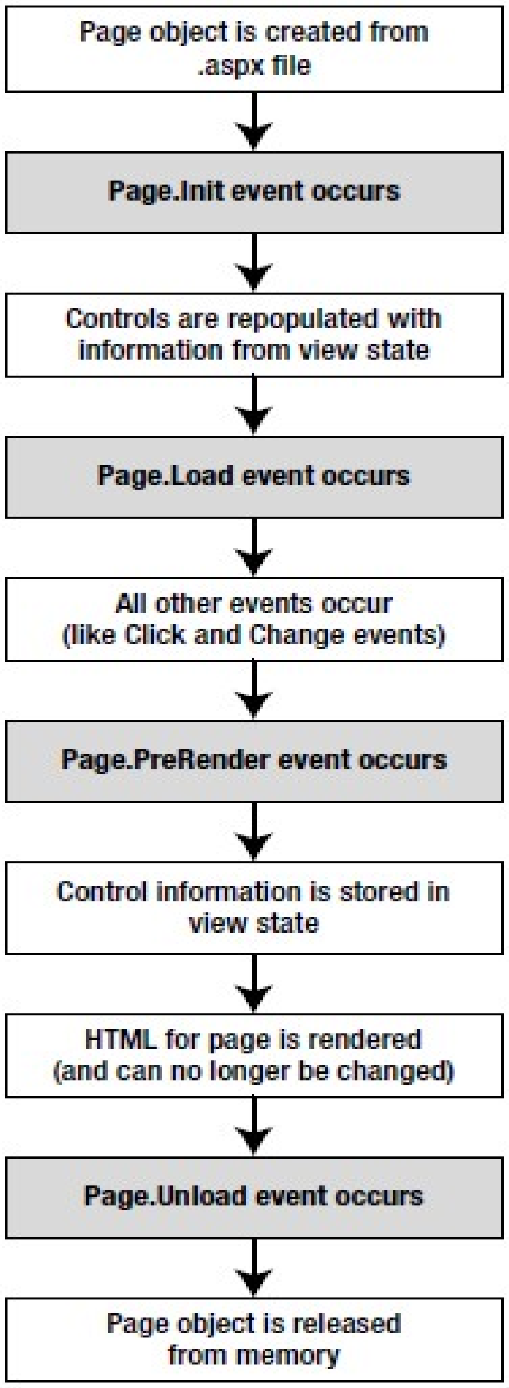

Web Page Request
ASP.NET creates page object from.aspx code
ASP.NET runs the Page.Load event handler
HTML Output Returned
Final page is rendered
Web Client
Page Postback
ASP.NET creates page object from .aspx code
ASP.NET runs the Page.Load event handler
ASP.NET runs any other triggered event handlers
HTML Output Returned
Final page is rendered

***Figure 6-11.** The page-processing sequence*



Page object is created from .aspx file
Page.Init event occurs
Controls are repopulated with information from view state
Page.Load event occurs
All other events occur (like Click and Change events)
Page.PreRender event occurs
Control information is stored in view state
HTML for page is rendered (and can no longer be changed)
Page.Unload event occurs
Page object is released from memory

The Postback processing Sequence

Automatic Postback

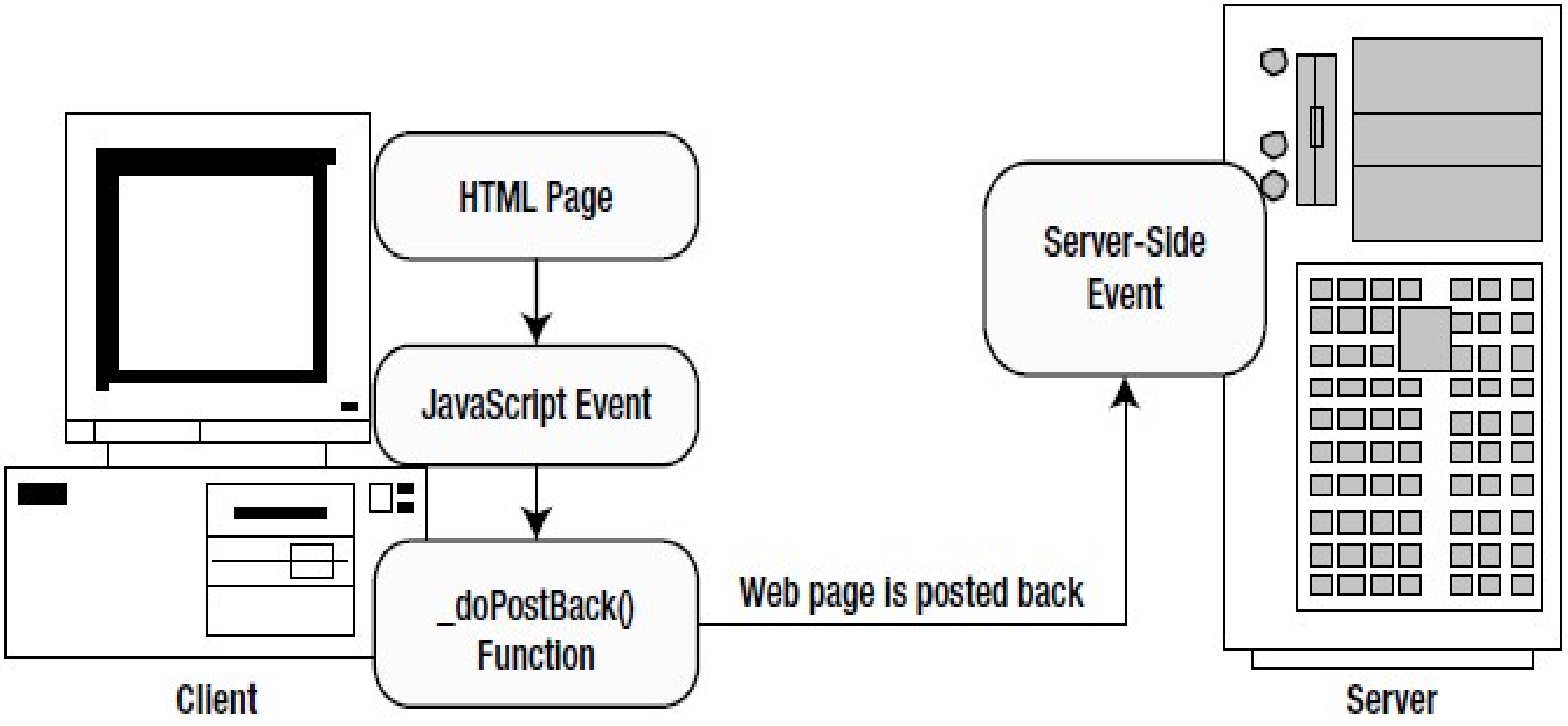


Figure 6-13. An automatic postback

Page Life Cycle Stages

Stage	Description
Request for page	The page request starts before a page has been instantiated. When a user requests a page from the web server. If cached version exists it is returned, else a new one.
Page Start	Page is instantiated. Request, Response , IsPostback and UICulture properties are set.
Initialization	During this stage, controls are instantiated and the page's control hierarchy is constructed. Theme and skin information is applied to the page during this stage.
Load	During this stage, control properties are restored from state if the request is a postback.
Validation	Validate() method of all validation controls are called to validate the state of controls on page.
Postback Event Handling	If the request is a postback, then postback events in response to such things as button clicks, selected index changes, and more are invoked.
Render	each control in the Page's control hierarchy is asked to contribute its own rendered output to the rendered output of the entire page. The Page's view state is also included.
Unload	child controls and the page dispose their resources. Response, Request data are cleaned

Page Life Cycle Events

Event	Description
Page_PreInit	Called at the beginning of the Initialize stage. Used to create dynamic controls, set master pages and themes dynamically, and to read/write user profile data.
Page_Init	Called during the Initialize stage to initialize control properties
Page_Load	Called during the Load stage to read control properties or update existing control properties. At this stage, control properties have been reconstituted from view state
(Control Events)	Processing of control events
Page_PreRender	This event is triggered just before the final version of the page is rendered. If you need to make final tweaks to properties of controls based on information that is only available immediately before rendering, this is the event you should use.
Page_Unload	This event is called immediately before the page is discarded by ASP.NET to dispose of costly resources such as database connections. Also commonly used to write final logging and tracing information.

Debugging

- ✓ Single Step Debugging
  1. Place a break point at any executable stmt and Press F5 (or Start Debugging Option from Debug menu).
  2. Program stops execution at the break point line. Now you can test variable values (through watch windows, or placing cursor over the variables).
  3. You can execute line by line by pressing F11 button. (single stepping).

- ✓ Variable Watches

By using variable watch windows, we can know the values of variables after execution of a statement.

Three types of windows:

1. Autos → include variables that are accessed or changed in the previous line.
2. Locals → Displays all the variables that are in the scope in the current method.
3. Watch → Displays variables you have added.

Topics

ViewState

Cross page posting

Query String

Cookies

Session State

HttpSessionState class

Member	Description
Count	Provides the number of items in the current session collection.
IsCookieless	Identifies whether the session is tracked with a cookie or modified URLs.
Keys	Gets a collection of all the session keys that are currently being used
Mode	Provides an enumerated value that explains how ASP.NET stores session-state information
SessionID	Provides a string with the unique session identifier for the current client.
Timeout	Determines the number of minutes that will elapse before the current session is abandoned, provided that no more requests are received from the client.
Abandon()	Cancels the current session immediately and releases all the memory it occupied.
Clear()	Removes all the session items.

Configuring session

```
<configuration>
...
<system.web>
...
<sessionState
cookieless="UseCookies"
cookieName="ASP.NET_SessionId"
regenerateExpiredSessionId="false"
timeout="20"
mode="InProc"
stateConnectionString="tcpip=127.0.0.1:42424"
stateNetworkTimeout="10"
sqlConnectionString="data source=127.0.0.1;Integrated Security=SSPI"
sqlCommandTimeout="30"
allowCustomSqlDatabase="false"
customProvider=""
compressionEnabled="false"
/>
</system.web>
</configuration>
```

Data controls

- **GridView** → Shows the tabular data in a Grid.
- The GridView control is used to display the values of a data source in a table. Each column represents a field where each row represents a record.
- The GridView control provides many built-in capabilities that allow the user to sort, update, delete, select and page through items in the control.
- The GridView control can be bound to a data source control.

GridView features:

- Improved data source binding capabilities
- Tabular rendering – displays data as a table
- Built-in sorting capability
- Built-in select, edit and delete capabilities
- Built-in paging capability
- Built-in row selection capability
- Multiple key fields
- Programmatic access to the GridView object model to dynamically set properties, handle events and so on
- Richer design-time capabilities
- Control over Alternate item, Header, Footer, Colors, font, borders, and so on.
- Slow performance as compared to Repeater and DataList control .

Populating columns of GridView

1. By setting AutoGenerateColumns property to true.

Disadv: it is not possible to explicity say, which properties should be displayed as columns, what the HeaderText or width of each column should be.

Ex:

```
<asp:GridView ID="gvUsers" runat="server"
    AutoGenerateColumns="true"></asp:GridView>
```

2. By using BoundField:

This allows you to create the columns allows to explicitly define, which columns should be displayed, how they look and in which order they are displayed.

- ✓ In order to specify the columns we need to set the **AutoGeneratedColumns** property to false.

Ex:

```
<asp:GridView ID="gvUsers" runat="server" AutoGenerateColumns="false">
  <Columns>
    <asp:BoundField HeaderText="ID" DataField="IDUser" ItemStyle-Width="50"/>
    <asp:BoundField HeaderText="Name" DataField="Name" ItemStyle-Width="200"/>
    <asp:BoundField HeaderText="Username" DataField="UserName" ItemStyle-
  Width="200"/>
  </Columns>
</asp:GridView>
```

Column types

Column type	Description
BoundField	This column displays text from a field in the data source.
ButtonField	This column displays a button in this grid column.
CheckBoxField	This column displays a check box in this grid column. It's used automatically for True / false fields.
CommandField	This column provides selection or editing buttons.
HyperLinkField	This column displays its contents (a field from the data source or static text) as a hyperlink.
ImageField	This column displays image data from a binary field.
TemplateField	This column allows you to specify multiple fields, custom controls, and arbitrary HTML using a custom template.

Populating columns of GridView

Ex:

```
<asp:BoundField DataField = "ProductID" HeaderText = "ID" />
```

Configuring columns of GridView using BoundField properties

BoundField properties:

Properties	Description
DataField	Identifies the field (by name) that you want to display in this column
DataFormatString	Formats the field. This is useful for getting the right representation of numbers and dates.
FooterText, HeaderText, and HeaderImageUrl	Sets the text in the header and footer region of the grid if this grid has a header (GridView.ShowHeader is true) and footer (GridView.ShowFooter is true).
ReadOnly	If true, it prevents the value for this column from being changed in edit mode. No edit control will be provided. Primary key fields are often read-only.
InsertVisible	If true, it prevents the value for this column from being set in insert mode

Configuring columns of GridView using BoundField properties

BoundField properties:

Properties	Description
Visible	If false, the column won’t be visible in the page.
SortExpression	Sorts your results based on one or more columns.
HtmlEncode	If true (the default), all text will be HTML encoded to prevent special characters from mangling the page.
NullDisplayText	Displays the text that will be shown for a null value.
ConvertEmptyStringToNull	If true, converts all empty strings to null values.
ControlStyle, HeaderStyle, FooterStyle, and ItemStyle	Configures the appearance for just this column, overriding the styles for the row.

Formatting GridView

- Each BoundField column provides a DataFormatString property you can use to configure the appearance of numbers and dates using a *format string*.
- Format strings generally consist of a placeholder and a format indicator, which are wrapped inside curly brackets.
- Ex: {0:C} → Currency format

Ex:

```
<asp:BoundField DataField = "UnitPrice" HeaderText = "Price"
DataFormatString = "{0:C}" />
```

Time & Date Format Strings:

Type	Format String	Syntax	Example
Short Date	{0:d}	M/d/yyyy	10/30/2012
Long Date	{0:D}	dddd, MMMM dd, yyyy	Monday, January 30, 2012
Long Date and Short Time	{0:f }	dddd, MMMM dd, yyyy HH:mm aa	Monday, January 30, 2012 10:00 AM
Long Date and Long Time	{0:F}	dddd, MMMM dd, yyyy HH:mm:ss aa	Monday, January 30, 2012 10:00:23 AM

GridView styles

Style	Description
HeaderStyle	Configures the appearance of the header row that contains column titles, if you've chosen to show it (if ShowHeader is true).
RowStyle	Configures the appearance of every data row.
AlternatingRowStyle	If set, applies additional formatting to every other row
SelectedRowStyle	Configures the appearance of the row that's currently selected.
EditRowStyle	Configures the appearance of the row that's in edit mode. This formatting acts in addition to the RowStyle formatting.
EmptyDataRowStyle	Configures the style that's used for the single empty row in the special case where the bound data object contains no rows.
FooterStyle	Configures the appearance of the footer row at the bottom of the GridView, if you've chosen to show it
PagerStyle	Configures the appearance of the row with the page links, if you've enabled paging (set AllowPaging to true).

Styles

EX:

```
<RowStyle BackColor = "#E7E7FF" ForeColor = "#4A3C8C" />
<HeaderStyle BackColor = "#4A3C8C" Font-Bold = "True" ForeColor = "#F7F7F7" />
```

EX:

```
    <asp:BoundField DataField = "ProductName" HeaderText = "Product Name">
        <ItemStyle BackColor = "#E7E7FF" ForeColor = "#4A3C8C" />
        <HeaderStyle BackColor = "#4A3C8C" Font-Bold = "True" ForeColor =
          "#F7F7F7" />
    </asp:BoundField>
```

Using a Data Field as a Select Button

- You don’t need to create a new column to support row selection. Instead, you can turn an existing column into a link.

- To use this technique use a add a ButtonField column. Then, set the DataTextField to the name of the field you want to use.

EX: `<asp:ButtonField ButtonType = "Button" DataTextField = "ProductID" />`

Sorting & Paging the GridView

- To enable sorting, you must set the GridView.AllowSorting property to true. Next, you need to define a SortExpression for each column that can be sorted.
- To use automatic paging, you need to set AllowPaging to true (which shows the page controls), and you need to set PageSize to determine how many rows are allowed on each page

Paging with the GridView

Ex:

```
<asp:GridView ID = "GridView1" runat = "server" DataSourceID = "sourceProducts"
PageSize = "10" AllowPaging = "True" . . .>
. . .
</asp:GridView>
```

- Set **GridView.EnablePersistedSelection** property to true to avoid the same row position from being selected as you move from one page to another.

GridView templates

- The TemplateField allows you to define a completely customized *template for a column. Inside the template* you can add control tags, arbitrary HTML elements, and data binding expressions.
- you want to create a column that combines the in-stock, on-order, and reorder level information for a product using ItemTemplate as shown below:

```
<asp:TemplateField HeaderText = "Status">
    <ItemTemplate>
        <b > In Stock:</b>
        <%# Eval("UnitsInStock") % > <br />
        <b > On Order:</b>
        <%# Eval("UnitsOnOrder") % > <br />
        <b > Reorder:</b>
        <%# Eval("ReorderLevel") %>
    </ItemTemplate>
</asp:TemplateField>
```

Paging member of the GridView

Property	Description
AllowPaging	Enables or disables the paging of the bound records. It is false by default
PageSize	Gets or sets the number of items to display on a single page of the grid. The default value is 10.
PageIndex	Gets or sets the zero-based index of the currently displayed page, if paging is enabled.
PagerSettings	Provides a PagerSettings object that wraps a variety of formatting options for the pager controls. These options determine where the paging controls are shown and what text or images they contain.
PagerStyle	Provides a style object you can use to configure fonts, colors, and text alignment for the paging controls.
PageIndexChanging and PageIndexChanged events	Occur when one of the page selection elements is clicked, just before the PageIndex is changed (PageIndexChanging) and just after (PageIndexChanged).

TemplateField templates

Mode	Description
HeaderTemplate	Determines the appearance and content of the header cell.
FooterTemplate	Determines the appearance and content of the footer cell (if you set ShowFooter to true).
ItemTemplate	Determines the appearance and content of each data cell.
AlternatingItemTemplate	Determines the appearance and content of even-numbered rows.
EditItemTemplate	Determines the appearance and controls used in edit mode.
InsertItemTemplate	Determines the appearance and controls used in edit mode. The GridView doesn't support this template, but the DetailsView and FormView controls do.

DetailsView control

- ✓ The DetailsView control uses a table-based layout where each field of the data record is displayed as a row in the control.
- ✓ Unlike the GridView control, the DetailsView control displays one row from a data source at a time by rendering an HTML table.
- ✓ The DetailsView supports both declarative and programmatic data binding.
- ✓ The DetailsView control is often used in master-detail scenarios where the selected record in a master control determines the record to display in the DetailsView control. It shows the details for the row in a separate space.
- ✓ We can provide styles or CSS for customizing the appearance of the DetailsView.
- ✓ By default displays information in two columns.

drawback: In pager navigation, whole set of records are retrieved from database even though one record data is displayed in the control.

Features of DetailsView control:

- ➢ Tabular rendering
- ➢ Supports column layout, by default two columns at a time
- ➢ Optional support for paging and navigation.
- ➢ Built-in support for data grouping
- ➢ Built-in support for edit, insert and delete capabilities

DetailsView control

- ✓ Set AutoGenerateRows to false to stop automatic generation of rows. Then you can display only the columns you want in the DetailsView.
- ✓ You can display, page, edit, insert, and delete database records with the DetailsView.
- ✓ If you need more control over the appearance of the DetailsView, including the particular order in which columns are displayed, then you can use fields with the DetailsView control.
- **BoundField**—Enables you to display the value of a data item as text.
- **CheckBoxField**—Enables you to display the value of a data item as a check box.
- **CommandField**—Enables you to display links for editing, deleting, and selecting rows.
- **ButtonField**—Enables you to display the value of a data item as a button (image button, link button, or push button).
- **HyperLinkField**—Enables you to display the value of a data item as a link.
- **ImageField**—Enables you to display the value of a data item as an image.
- **TemplateField**—Enables you to customize the appearance of a data item.
- ✓

DetailsView control

```
<asp:DetailsView ID = "DetailsView1" runat = "server" AutoGenerateRows =
    "False"
DataSourceID = "sourceProducts">
     <Fields>
     <asp:BoundField DataField = "ProductID" HeaderText = "ProductID"
     ReadOnly = "True" />
     <asp:BoundField DataField = "ProductName" HeaderText = "ProductName" />
     <asp:BoundField DataField = "SupplierID" HeaderText = "SupplierID" />
     <asp:BoundField DataField = "CategoryID" HeaderText = "CategoryID" />
     <asp:BoundField DataField = "QuantityPerUnit" HeaderText = "QuantityPerUnit" />
     <asp:BoundField DataField = "UnitPrice" HeaderText = "UnitPrice" />
     <asp:BoundField DataField = "UnitsInStock" HeaderText = "UnitsInStock" />
     <asp:BoundField DataField = "UnitsOnOrder" HeaderText = "UnitsOnOrder" />
     <asp:BoundField DataField = "ReorderLevel" HeaderText = "ReorderLevel" />
     <asp:CheckBoxField DataField = "Discontinued" HeaderText = "Discontinued" />
     </Fields>
. . .
</asp:DetailsView>
```

DetailsView properties

- ✓ AutoGenerateDeleteButton, AutoGenerateEditButton, AutoGenerateInsertButton propertie are used for enabling delete/edit/insert for detailsview.
- ✓ AutoGenerateEditButton → Gets or sets a value indicating whether the built-in controls to edit the current record are displayed in a DetailsView control.
- ✓ AutoGenerateInsertButton → Gets or sets a value indicating whether the built-in controls to insert a new record are displayed in a DetailsView control.
- ✓ AutoGenerateRows → Gets or sets a value indicating whether row fields for each field in the data source are automatically generated and displayed in a DetailsView control.
- ✓ BackImageUrl → Gets or sets the URL to an image to display in the background of a DetailsView control.
- ✓ DataMember → Gets or sets the name of the list of data that the data-bound control binds to, in cases where the data source contains more than one distinct list of data items.
- ✓ DataSourceID → Gets or sets the ID of the control from which the data-bound control retrieves its list of data items.
- ✓ **Hyperlinks are displayed at the bottom of the control.**

DetailsView control styles

- ✓ **AlternatingRowStyle** → allows you to set the appearance of the alternating data rows in a DetailsView control.
- ✓ **CommandRowStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of a command row in a DetailsView control.
- ✓ **EditRowStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the data rows when a DetailsView control is in edit mode.
- ✓ **EmptyDataRowStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the empty data row displayed when the data source bound to a DetailsView control does not contain any records.
- ✓ **FieldHeaderStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the header column in a DetailsView control.
- ✓ **FooterStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the footer row in a DetailsView control.
- ✓ **HeaderStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the header row in a DetailsView control.
- ✓ **InsertRowStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the data rows in a DetailsView control when the DetailsView control is in insert mode.
- ✓ **PagerStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the pager row in a DetailsView control.
- ✓ **RowStyle** → Gets a reference to the TableItemStyle object that allows you to set the appearance of the data rows in a DetailsView control.

FormView

- ✓ **Requires Templates**
- ✓ **Displays columns without a table**
- ✓ **Templates supported by FormView control:**
  - ItemTemplate
  - EditItemTemplate
  - InsertItemTemplate
  - FooterTemplate
  - HeaderTemplate
  - EmptyDataTemplate
  - PagerTemplate

FormView

You can use FormView to display multiple item values in a single value:

Ex:

```
<asp:FormView ID = "FormView1" runat = "server" DataSourceID =
  "sourceProducts">
<ItemTemplate>
<b > In Stock:</b>
<%# Eval("UnitsInStock") %>
<br />
<b > On Order:</b>
<%# Eval("UnitsOnOrder") %>
<br />
<b > Reorder:</b>
<%# Eval("ReorderLevel") %>
<br />
</ItemTemplate>
</asp:FormView>
```

FEATURES

Configuration files:

web.config → Used to set application specific configuration settings.

machine.config → located in OS dir.

Tools:

- Configuration settings editor
- Website Administration tool

- .