Lecture 1

Introduction to ASP.NET

Objectives

In this lecture you will learn the following

+ Knowing about Webserver & Virtual Directory
+ Learning the ASP.NET Architecture
+ Features of ASP.NET
+ Advantages of ASP.NET
+ Advantages of ASP.NET
+ A simple ASP.NET Application

Coverage Plan

Lecture 1

1.1 Snap Shot

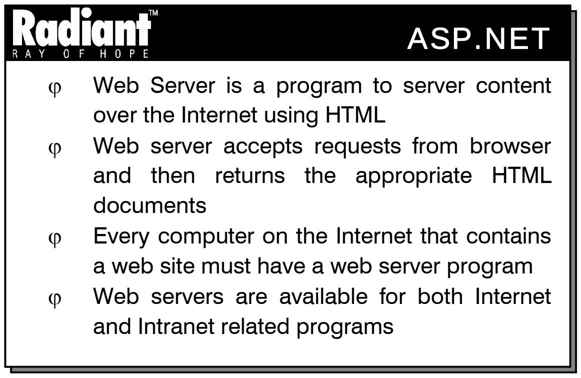
This session deals with webservers, the difference between static and dynamic web pages, how to create a virtual directory. This session then proceeds on to ASP.NET, its architecture, its features and advantages. A Web server is a server, which uses the Hypertext Transfer Protocol (HTTP) and sends the files which consists of Web pages to Web users when requested. All the web pages are stored in a web server. ASP.NET is the next generation of Microsoft's Active Server Page (ASP), a feature of their Internet Information Server (IIS). ASP.NET allows to create Web pages dynamically by inserting queries to a relational database in the Web page.

1.2 Web Server

Generally, all the machines on the Internet can be categorized into two types namely servers and clients. Machines that provide services, like Web servers or FTP servers, to other machines are called as servers. Machines that are used to connect to those services are known as clients.

The term Server can be used for any of the following:

- A computer on a network that sends files to or runs applications for other computers on the network
- A software that runs on the server computer and performs the work of serving files or running applications
- A code that exchanges information with another upon request



Radiant™ RAY OF HOPE ASP.NET

- φ Web Server is a program to server content over the Internet using HTML
- φ Web server accepts requests from browser and then returns the appropriate HTML documents
- φ Every computer on the Internet that contains a web site must have a web server program
- φ Web servers are available for both Internet and Intranet related programs

Web server is a server to serve content over the Internet using the Hypertext Markup Language (HTML). The Web server accepts requests from browsers like Netscape and Internet Explorer and then returns the appropriate HTML documents. Every computer on the Internet that contains a web site must have a Web server program.

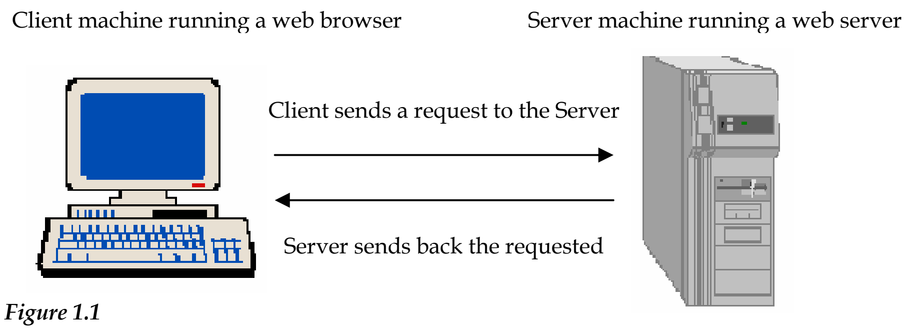

Client machine running a web browser
Server machine running a web server
Client sends a request to the Server
Server sends back the requested

Figure 1.1

When the client asks for information such as a file, the Web server gets the file and sends it to the client. In most cases, the Web server does not read or process this file, but sends it to the client.

A number of Server-Side technologies can be used to increase the power of the server beyond its ability to deliver standard HTML pages, which include:

- CGI scripts
- Server-Side Includes (SSI)
- Active Server Pages (ASP)

The most popular Web servers are Microsoft's Internet Information Server (Internet Information Server), which comes with the Windows NT server, Netscape FastTrack and Enterprise servers and Apache server, a Web server for UNIX-based operating systems. Other Web servers include Novell's Web Server for users of its NetWare operating system and IBM's family of Lotus Domino servers, primarily for IBM customers.

Web servers are available for both Internet and Intranet related programs. Some of the functionalities include:

- Serving E-Mail
- Downloading requests for File Transfer Protocol files
- Building and publishing Web pages

Considerations in choosing a Web Server include the ability to work with the operating system and other servers. Selection should also be based on the ability to handle Server-Side programming, publishing, search engine, and site building tools that accompany it.

Web Page

Web page is a World Wide Web document. A Web page typically consists of an HTML file, with associated files for graphics and scripts, in a particular directory on a particular computer (and thus identifiable by a URL). Web pages can be classified as static or dynamic web pages.

A **static web page** is a page whose content is changed before being sent over the Internet to the user's computer. Such pages appear to the user to be **static**. An example of this kind of a web page is a page that is created by a search engine. The static web page will not have any animation or lively user interactions.

A **dynamic web page** is a page that contains animated GIFs, Java applets, ActiveX Controls, or DHTML. It is a Web page created automatically based on information provided by the user, or generated with ASP.

1.3 Virtual Directory

ASP.NET

- φ Virtual directory is created in order to publish from any directory not contained within the home directory
- φ Virtual directory is not contained in the home directory
- φ Virtual directory has an *alias*, a name that web browsers use to access that directory

All the documents of a web site are stored in directories. The Web server cannot publish documents that are not within these specified directories. The first step in deploying a web site is to determine how the files need to be organized.

If a special directory is not created to store the documents, there is a default home directory in which all the documents can be published by copying the Web files into the default home directory.

Home directory is the root directory for a web site, where the content files are stored. It is also called as a document root or Web root. In Internet Information Services, the home directory and all its subdirectories are available to users by default. Normally, the home directory for a site contains the home page.

A **Virtual directory** is created in order to publish from any directory not contained within the home directory. A virtual directory is not contained in the home directory but appears to client browsers as though it were. A virtual directory has an *alias*, a name that Web browsers use to access that directory. Since an alias is usually shorter than the path of the directory, it is more convenient for users to type the alias.

An alias is more secure and advantageous as:

- Users do not know where the files are physically located on the server
- Users cannot utilize that information to modify the files
- Makes it easier to move directories in the site
- Instead of changing the URL for the directory, it is possible to change the mapping between the alias and physical location of the directory

Creating a Virtual Directory

ASP.NET

Creating a Virtual Directory

- φ In the Internet Information Services snap-in, the web site must be selected to which a directory is added
- φ Click the Action button, point to New, and select the Virtual Directory
- φ Use the New Virtual Directory Wizard to complete the task

As seen in the previous section, if the web site contains files that are located on a drive different from the home directory, or on other computers, virtual directories must be created to include those files in the web site. To use a directory on another computer, the directory's Universal Naming Convention (UNC) name must be specified and the user must be provided with a user name and password for access permission.

The following steps are used to create a virtual directory.

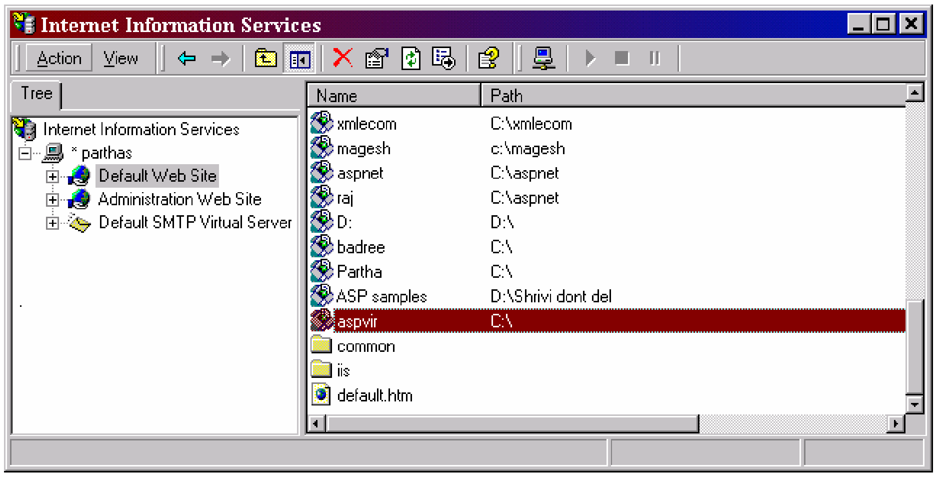
- In the Internet Information Services snap-in, select the web site or FTP site to which a directory has to be added
- Click the **Action** button, and then point to **New**, and select **Virtual Directory**
- Use the **New Virtual Directory** Wizard to complete the task

Note : While using NTFS, a virtual directory can be created by right-clicking a directory in the Windows Explorer, clicking **Sharing**, and then selecting the **Web Sharing** property sheet.

To delete a virtual directory the following steps have to be followed:

- In the Internet Information Services snap-in, select the virtual directory that is to be deleted
- Click the **Action** button, and select **Delete**

An ASP.NET application can be stored either in the Home or Virtual directory.



Internet Information Services
- * parthas
  - Default Web Site
  - Administration Web Site
  - Default SMTP Virtual Server

Name	Path
xmlecom	C:\xmlecom
magesh	c:\magesh
aspnet	C:\aspnet
raj	C:\aspnet
D:	D:\
badree	C:\
Partha	C:\
ASP samples	D:\Shrivi dont del
aspvir	C:\
common	
iis	
default.htm	

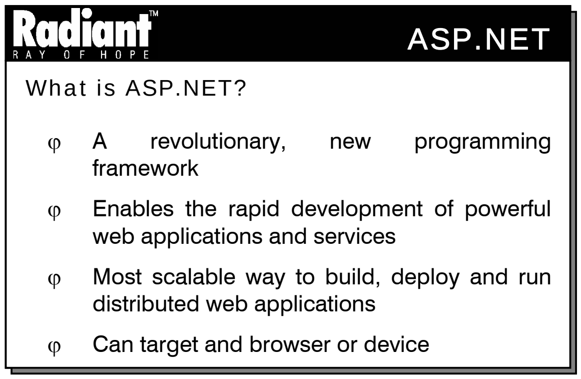
Figure 1.2

1.4 Introduction to ASP.NET

ASP.NET is an entirely new framework from Microsoft for building Next Generation Web Applications. ASP.NET is component-based and modularized. Every page, object, and HTML element in ASP.NET can be accessed at runtime as a component.

The key component of the new .NET framework has the potential to save organizations time and money by allowing them to establish object-oriented frameworks for the web applications. Microsoft has done an excellent job by automating common web development tasks. But more importantly they have created a tool that gives developers the ability to handle any business problem.

What is ASP.NET?



ASP.NET

What is ASP.NET?

- A revolutionary, new programming framework
- Enables the rapid development of powerful web applications and services
- Most scalable way to build, deploy and run distributed web applications
- Can target and browser or device

ASP.NET is a revolutionary new programming framework that enables the rapid development of powerful web applications and services. Part of the emerging Microsoft .NET Platform, it provides the easiest and most scalable way to build, deploy and run distributed web applications that can target any browser or device.

How ASP.NET uses .NET framework

ASP.NET is a compiled .NET-based environment. The developer can author applications in any .NET compatible language, including Visual Basic, C# and Jscript.NET. Additionally, the entire .NET Framework is available to any ASP.NET application. Developers can easily access the benefits of these technologies, which include a managed Common Language Runtime environment, type safety, inheritance, and so on.

ASP.NET has been designed to work seamlessly with WYSIWYG HTML editors and other programming tools, including Microsoft Visual Studio.NET. Not only does this make Web development easier, but also provides all the benefits that these tools have to offer, including a GUI that developers can use to drop server controls onto a Web page, as well as fully integrated debugging support.

1.5 ASP.NET Architecture

This section provides an overview of the ASP.NET infrastructure and the subsystem relationships. These relationships are shown in the following illustration:

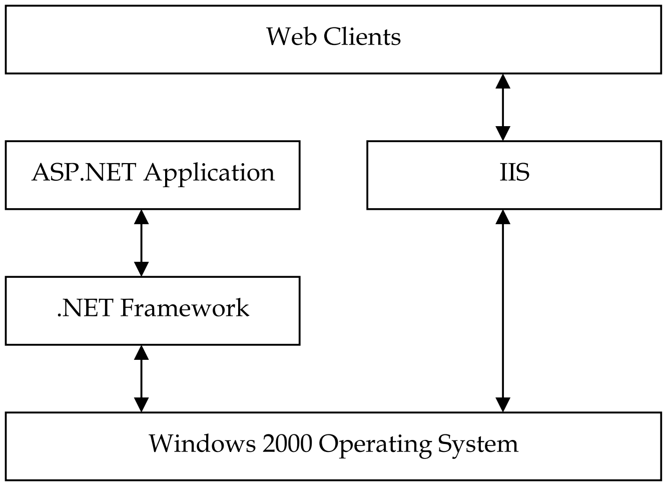


Figure 1.3

As shown, all Web clients communicate with ASP.NET applications through IIS. IIS deciphers the request, authenticates, and finds the requested resource (such as an ASP.NET application). If authorized, it returns the appropriate resource to the client. In addition to the built-in ASP.NET security features, an ASP.NET application can use the Common Language Runtime low-level security features.

Integrating with IIS

This release of ASP.NET uses IIS 5.0 as the primary host environment. When considering ASP.NET authentication, the interaction with IIS authentication services must be understood.

There are three different kinds of authentication in IIS 5.0: Basic, Digest, and Integrated Windows Authentication. The type of authentication used can be set in the IIS administrative services. If a URL containing an ASP.NET application is requested, both the request and information about authentication get handed over to the application. ASP.NET provides two additional types of authentication: Cookie authentication and Passport authentication.

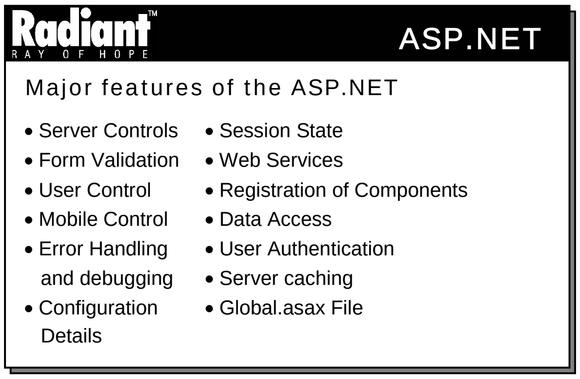
Cookie-Based Authentication

Cookie authentication is a process that allows the application to collect credentials directly from the client requestor (usually name and password), and determine their authenticity. IIS authentication is not used by the application however, the IIS authentication settings are important to the ASP.NET cookie authentication process. Unless the IIS "Anonymous Access" setting is enabled, requests that do not meet the criteria for IIS authentication will be rejected and never reach the ASP.NET application.

Passport Authentication

Passport authentication is a centralized authentication service provided by Microsoft that offers a single sign-in and core profile services for member sites.

1.6 ASP.NET Features



Radiant™ RAY OF HOPE ASP.NET

Major features of the ASP.NET

• Server Controls	• Session State
• Form Validation	• Web Services
• User Control	• Registration of Components
• Mobile Control	• Data Access
• Error Handling and debugging	• User Authentication
	• Server caching
• Configuration Details	• Global.asax File

The major features of the ASP.NET are as follows:

Server Controls

ASP.NET introduces the concept of Server Controls, which enable the values of the controls to be manipulated on the Server-Side.

An amazing feature of the Server-Side controls is the ability of these controls to fire Server-Side events. Thus, the control appearing at the client, with runat attribute set to server can fire Server-Side events. This is particularly useful when validating and dynamically generating forms.

The Server-Side controls can be subdivided into the following categories based on the functionalities that they offer.

Intrinsic Controls

These controls create HTML elements on the client browser with the added capability of maintaining the state.

List Controls

These Controls are used to display a wide variety of lists on the Client-Side.

Rich Controls

These are used to output HTML in a rich format to the client. The calendar control is an example of this category.

Validation Controls

These make the art of Server-Side validation a cakewalk.

Session State

In ASP.NET, the Session state is automatically maintained by using a hidden field called view state. Further, ASP.NET offers the flexibility to turn the Session state on or off at both the page level as well as the control level. It is also possible to store the Session state directly in the SQL Server database and retrieve at a later time.

Form Validation

ASP.NET offers a rich set of server controls to achieve the task of Form Validation. These controls are popularly known as Validators or validation controls and can be used to validate all types of validations.

Web Services

ASP.NET offers a new concept called Web Services by which a web application can expose its functionality to others on the Internet by employing a protocol such as Simple Object Access Protocol (SOAP) and in XML format. This is entirely a new concept. Web Services can be employed for displaying news headlines, weather reports etc.

User Controls

ASP.NET offers code reusability by introducing the concept of “User Controls.” A User Control is a pseudo control, wrapped with common code that can expose properties, methods and events. These can be employed for opening and closing a database connection, which happen quite frequently in all the pages. User Controls can be employed to wrap up the most frequently used HTML code and reuse it across several Web pages.

Registration of Components

In web development, it is a common practice to apply the business logic as COM Components. This concept has been received very well by the developers, as there is a performance boost when such business objects are employed. This performance boost is due to the fact that the components are compiled. But the maintenance poses a great problem. If a component has to be unregistered from use by the web server, the server has to be restarted.

ASP.NET has the solution for this problem. If a component is to be registered in ASP.NET, it has to be copied in the web application's "bin" directory. The file has to be deleted to unregister the component.

Mobile Controls

ASP.NET is in tune with M-Commerce by providing .NET Mobile Web SDK that allows applications to be built covering a wide range of mobile devices ranging from cellular phones to Personal Digital Assistants.

Data Access

ASP.NET has introduced the next generation of ADO known as ADO.NET with respect to data access. ADO.NET places more emphasis on disconnected recordsets by employing XML as a medium of communication between these recordsets and the DataStore.

Error Handling and Debugging

ASP.NET offers a cleaner approach to error handling and debugging as compared to its predecessor ASP. It is now possible to specify individual error pages for each ASP.NET page by employing the new ErrorPage directive.

```
<%@page Errorpage = "/myErrorPage.aspx"%>
```

In the above example, if any error occurs on the page, the user is directed to myErrorPage.aspx. This type of error handling enables developers to circumvent the IIS error pages altogether.

User Authentication

ASP.NET offers several kinds of authentication for checking the validity of the users accessing a web site. In addition to support for Basic, Digest and Windows NT(NTLM), ASP.NET also supports Passport authentication (explained later).

Server-Side Caching

The process of retaining most frequently visited web pages in memory is called caching. Caching of ASP.NET Page leads to an increase in its performance. ASP.NET supports the following two types of caching

- Output caching
- Data Caching

Output caching is the process of caching the entire page, this kind of caching is advisable for sites with heavy traffic. Data caching is the process of caching a part of the page that includes objects and data.

Configuration Details

ASP.NET maintains all the configuration details pertaining to a web application in a file called Config.web which is in a human readable format (XML). The whole range of configuration details including which request to handle on specific situation, tracing options etc.

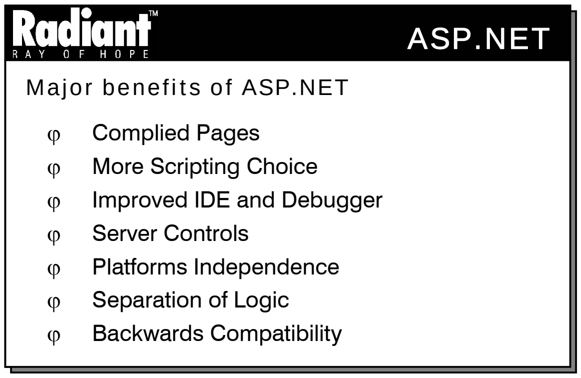
Global.asax File

In ASP.NET it is possible to specify the event-handlers that need to be invoked on the occurrence of particular events. These event-handlers are defined in a global file by name Global.asax. The Global.asax file of ASP.NET has a provision for the following event-handlers

- Application_OnStart
- Application_OnEnd
- Session_OnStart
- Session_OnEnd
- Application_BeginRequest
- Security_OnAuthenticate

Finally, ASP.NET provides a powerful framework for developing and deploying Next Generation Web Applications. ASP.NET will have a significant impact on the web application development in future.

1.7 Advantages of ASP.NET


Radiant™ RAY OF HOPE ASP.NET

Major benefits of ASP.NET

φ Complied Pages
φ More Scripting Choice
φ Improved IDE and Debugger
φ Server Controls
φ Platforms Independence
φ Separation of Logic
φ Backwards Compatibility

The greatly improved ASP.NET runtime handles many routine tasks in the code. Although there are major benefits of ASP.NET, here are some that will prove beneficial to the majority of the development tasks.

Compiled Pages

A page is compiled into a .NET class when it is requested the first time. The runtime engine will detect if any changes have been made to the source code and recompile it if necessary. The user can optionally specify a class file with the page, otherwise it will be created by the ASP runtime.

More Scripting Choice

The user can code pages with Visual Basic, C#, or JScript.NET. VBScript has been incorporated into the VB syntax.

Improved IDE and Debugger

The entire .NET family shares the same IDE and debugging tools. The user will benefit from the feature rich debugging tools of ASP.NET.

Server Controls

Server-Side components automate many common development tasks. These controls can detect the version of the browser and generate the proper HTML and JavaScript code.

Browser Independence

The ASP.NET runtime is capable of detecting the browser type of the requesting client and generate the proper HTML or JavaScript code. This has the potential of making ASP.NET applications browser independent.

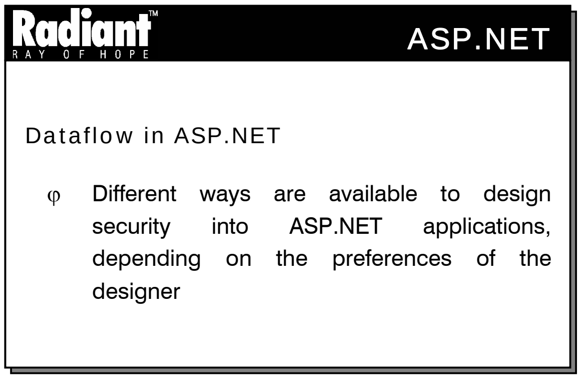
Separation of Logic

ASP.NET has been designed to separate business logic and presentation by utilizing ASP and COM. The task of encapsulating the logic from the user's presentation has been made easier. Application separation is the key to productivity. By creating .NET components and class files, the user can maximize code re-use.

Backward Compatibility

Organizations will be able to leverage their existing investments in COM and COM+. ASP.NET applications can interact with legacy COM objects.

1.8 Dataflow in ASP.NET



Radiant™ RAY OF HOPE

ASP.NET

Dataflow in ASP.NET

φ Different ways are available to design security into ASP.NET applications, depending on the preferences of the designer

The data flow in this scenario is shown in the following illustration:

As shown in the figure 1.4, the sequence of events is as follows:

1. A client generates request for a protected resource.

2. IIS receives the request, and if the requestor is authenticated by IIS - or if IIS "Anonymous Access" is enabled, it is passed on to the ASP.NET application. Because the authentication mode in the ASP.NET application is set to "cookie", any IIS authentication is not used.

3. ASP.NET checks to see if a valid authentication cookie is attached to the request. If the identity's credentials have been confirmed previously, the request is tested for authorization. The authorization test is performed by ASP.NET and accomplished by comparing the credentials contained in the request's authorization cookie with the authorization settings in the application configuration files (config.web). If the user is authorized, access is granted to the protected resource - or the application may require an additional test of the credentials before authorizing access to the protected resource, depending on the design of the application.

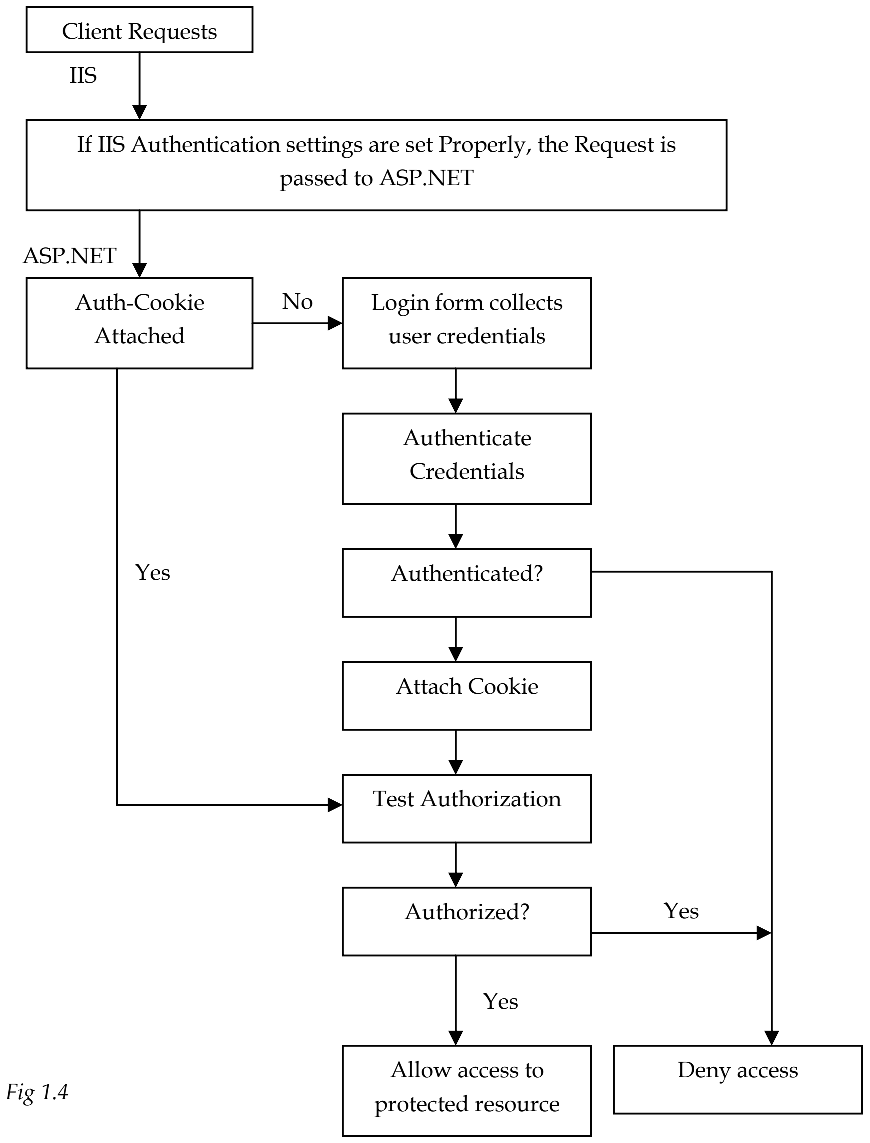

Client Requests
IIS
If IIS Authentication settings are set Properly, the Request is passed to ASP.NET
ASP.NET
Auth-Cookie Attached
No
Login form collects user credentials
Authenticate Credentials
Yes
Authenticated?
Attach Cookie
Test Authorization
Authorized?
Yes
Yes
Allow access to protected resource
Deny access

Fig 1.4

4. If there is no cookie attached to the request, ASP.NET redirects the request to a login page (the path of which resides in the application's configuration file) where the client user enters the required credentials (usually a name and password).

5. The application code checks the credentials to confirm their authenticity (usually in an event handler) and if authenticated, attaches a cookie ticket containing the credentials to the request. If authentication fails, the request is usually returned with an "Access Denied" message.

6. If the user is authenticated, ASP.NET checks authorization as in step 3, and can either allow access to the originally requested, protected resource or redirect the request to some other page, depending on the design of the application. It can otherwise, direct the request to a custom form of authorization where the credentials are tested for authorization to the protected resource. Usually if authorization fails, the request is returned with an "Access Denied" message.

1.9 A Simple ASP.NET Application

An ASP.NET application can be saved either in a Home directory or Virtual directory.

Practice 1.1

The following example illustrates how an ASP.NET application can be saved.

```
<HTML>
<HEAD>
<TITLE>ASP Script</TITLE>
<script language="C#" runat="server">
 void Page_Load(Object s,EventArgs x)
{
Response.Write("Welcome to Radiant");
}
</script>
</HEAD>
</HTML>
```

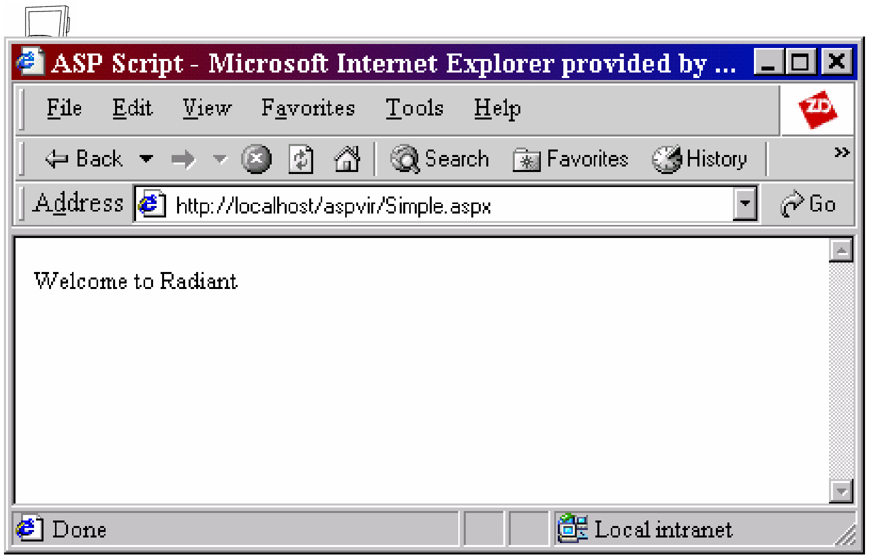
Save this file as Simple.aspx in the **Inetpub\wwwroot** directory, which is the **home directory**. (If no alteration has been made in the Internet Information Server's Service Manager, this will be the home directory for the default web site).

(or)

Place the Simple.aspx file in any desired directory, and create a **virtual directory** (see Creating a Virtual Directory) to map to the physical directory in which the Simple.aspx file has been stored.

Open Simple.aspx in the Web browser by specifying its URL as in the following:

`http://machine_name/ Simple.aspx`



ASP Script - Microsoft Internet Explorer provided by ...
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost/aspvir/Simple.aspx
Go
Done
Local intranet

Welcome to Radiant

As it can be seen from the above output, the application is saved in a virtual directory called "aspvir".

1.10 Short Summary

- Web server is a program that is used to serve content over the Internet using the Hypertext Markup Language (HTML)
- When the client asks for information such as a file, the Web server gets the file and sends it to the client
- Web servers are available for both Internet and Intranet related programs
- A Web page typically consists of an HTML file with associated files for graphics and scripts in a particular directory on a particular computer (and thus identifiable by a URL)
- A static web page is a page whose content is changed before being sent over the Internet to the user's computer. Such pages appear to the user to be static
- A dynamic web page is a page that contains animated GIFs, Java applets, ActiveX Controls, or DHTML
- Home directory is the root directory for a web site, where the content files are stored
- A virtual directory is not contained in the home directory
- ASP.NET is an entirely new framework from Microsoft for building Next Generation Web Applications.
- The major features of the ASP.NET are server control, session state, form validation, web services, user control, mobile control, data access, error handling and debugging, user authentication and server- side caching etc.
- The various advantages provided by the ASP.NET environment are compiled pages, more scripting charges, server controls, platform independence and separation of logic etc.
- There are different ways of securing ASP.NET applications, depending on the preferences of the designer
- An ASP.NET application can be saved either in the Home directory or Virtual directory

1.11 Brain Storm

1. What is a web server? What are the various Server-Side technologies that are available to deliver standard web pages.
2. What is the difference between a static and dynamic web page?
3. What is a virtual directory and in what way does it differ from a home directory?
4. What is ASP.NET? How does it use the .NET framework?
5. What are the features of ASP.NET?
6. What are the advantages of ASP.NET?

Lecture 2

HTML Server-Side Controls

Objectives

In this lecture you will learn the following

+ Basic coding in ASP.NET
+ Learning various server-side controls

Coverage Plan

Lecture 2

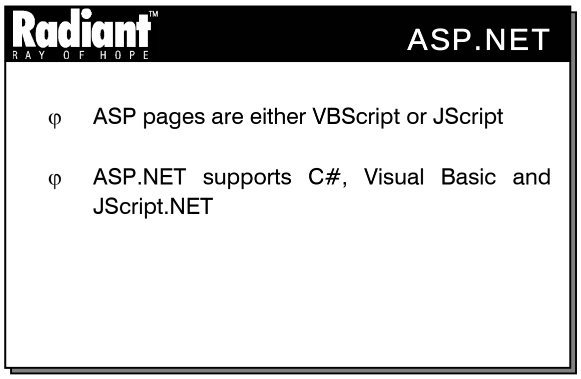
2.1 Snap Shot

This session is focussed on the basics of coding in ASP.NET, structure of code blocks and various HTML Server-Side Controls.

ASP executes components on the server and generates sections of the page that are returned to the user. This concept is extended in ASP.NET through server controls. The requirement to convert an HTML element into a server control is the addition of an extra attribute: **runat="server"**. Any HTML element in a page can be marked using this method and ASP.NET processes the elements on the server and generates output which is suitable to a specific client.

One of the most tedious tasks in creating interactive Web sites and applications is managing the values passed to the server from HTML form controls, and maintaining the values between page requests. Hence, one of the aims of ASP.NET is to simplify this programming task that involves no extra effort on the part of the programmer, and works well with any browser that supports basic HTML.

2.2 Basics of Coding in ASP.NET



Radiant™ RAY OF HOPE ASP.NET

φ ASP pages are either VBScript or JScript

φ ASP.NET supports C#, Visual Basic and JScript.NET

Most of the existing ASP pages use either VBScript or Jscript. Since both of these are being interpreted languages, they are no longer supported by ASP.NET.

ASP.NET, currently offers built-in support for three languages: C#, Visual Basic, and JScript.NET. Component-based applications can be created using any of the Visual Studio .NET languages: Managed Extensions (produced using Visual C++ .NET), Visual Basic, and C#.

An ASP.NET page is restricted to code written in a SINGLE language. The default language is C#, but any other compliant language can be declared as the default for the page through a directive like:

```
<%@ Language = VB%>
```

The language may also be declared within <script> blocks as shown below:

```
<script language = VB>
```

If different languages are declared in separate script blocks on the same page, an error will be thrown.

"Render functions" are not supported in ASP.NET. It is possible to insert literal HTML within a procedure body in the earlier versions of ASP:

```
<% Sub SomeProcedure() %>
<H3> Render this line in bold text. </H3>
<% End Sub %>
```

In an ASP.NET page, this can be written as:

```
<script runat=server>
Sub SomeProcedure()
Response.Write("<H3> Render this line in bold text.</H3>")
End Sub
</script>
```

Postback events

A Web page must be static to be viewed through the widest range of browsers. This means any interaction with the user (such as updating the display in response to a button click) must be resolved by calling the Web server and sending a new page.

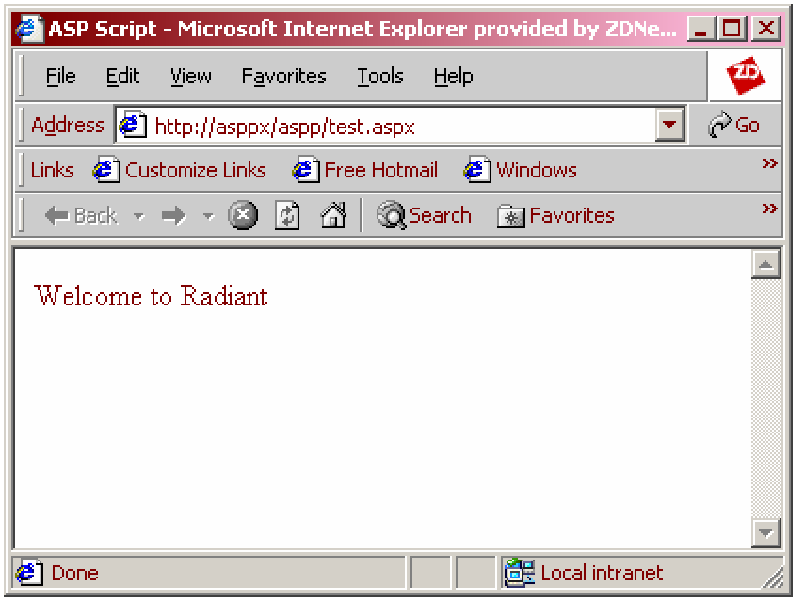
The simplest way of implementing a request-response dialog is through the HTML form element. When a form is submitted, it invokes its target URL, sends the page and data from its data entry controls. A well-written ASP page submits the form to it, using hidden fields to cache the data needed to reconstruct the current state in the refreshed client display.

A postback event occurs on the client, but handled by the code in a copy of the page running on the server. For example, in an ordinary page a pushbutton can have an OnClick event, which is handled by the client-side code. In ASP.NET pushbutton can have an OnServerClick event, which causes a round-trip back to the server. When accessed by the server- side event handler, the values of the form's control properties reflect the data submitted by the client-side form, and any updates performed by the handler are reflected in the refreshed client display. Postback events are the key to the advanced features of DataGrid and DataList controls. They simplify development by hiding the difficulties of the stateless HTTP model.

Practice 2.1

The following code shows a simple ASP.NET application.

```
<HTML>
<HEAD>
<TITLE>ASP Script</TITLE>
<script language="C#" runat="server">
void Page_Load(Object sender,EventArgs e)
{
     Response.Write("Welcome to Radiant");
}
</script>
</HEAD>
<HTML>
```


ASP Script - Microsoft Internet Explorer provided by ZDNe...
File Edit View Favorites Tools Help
Address http://asppx/aspp/test.aspx
Go
Links Customize Links Free Hotmail Windows
Back Search Favorites
Done
Local intranet

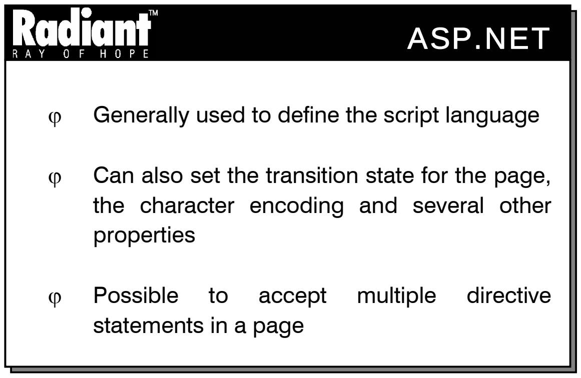
Welcome to Radiant

In the above program, the **Response.Write** method is used in **Page_Load** to display the message in the browser.

2.3 Structure of Code Blocks

In the previous versions of ASP, the **<script>** element or the script delimiters **<%...%>** were used to enclose script code in a page. In ASP.NET all functions and subroutines must be placed within the **<script>** section. The script delimiters **<%...%>** cannot be used.

Page Directives


Radiant™
RAY OF HOPE
ASP.NET

φ Generally used to define the script language

φ Can also set the transition state for the page, the character encoding and several other properties

φ Possible to accept multiple directive statements in a page

In the previous versions of ASP, a single page can contain only one page directive. The directive is generally used to define the script language that is used to code. For example,

```
<%@Language="JScript"%>
```

The transition state for the page, the character encoding and several other properties can also be set by using this directive. In ASP.NET, it is possible to accept multiple directive statements in a page. The default directive has the name **Page**.

```
<%@Page Language="JScript"%>
```

The above directive supports the existing ASP attributes like **Transition="Required"**. The other directives supported are listed in Table 2.1.

Attribute	**Values supported**	**Description**
BUFFER	True or False	Specifies if response buffering is on.
CONTENTTYPE	Any content type string	HTTP content type for the output.
CODEPAGE	Valid codepage value	Locale code page of the file.
CULTURE	Valid COM+ culture string	Culture setting of the page.
RESPONSEENCODING	GetEncoding() values	Encoding of response content.
LCID	Valid LCID identifier	Locale identifier for the code.
DESCRIPTION	String description	Text description of the page.
ENABLESESSIONSTATE	True or False (read-only)	Session state requirements.
ERRORPAGE	Valid HTTP URL	URL for unhandled errors.
INHERITS	Page-derived COM+ class	"Code-behind" class for the page to inherit from.
LANGUAGE	VB, JScript, C# or {other}	Language used when compiling all <%…%> blocks in the page.
MAINTAINSTATE	True or False	Indicates whether viewstate should be maintained across page requests.
TRANSACTION	Not supported, Supported,	Indicates the transaction semantics.
	Required or RequiresNow	
TRACE	True or False	Indicates whether tracing is enabled.
SRC	Source file name	'Code-behind' class to be compiled.
WARNINGS	True or False	Indicates whether compiler warnings should be treated as errors or ignored.

Table 2.1

In order to maintain backward compatibility, the directive statements in a page having no recognized directive name like Page, Register etc. will be considered as the **Page** directive statement. For example,

```
<%@Language="JScript"%>
```

is taken as

```
<%@Page Language="JScript"%>
```

Web Forms Server Controls

Server controls are specifically designed to work with Web Forms. They differ from Windows controls since they work within the ASP.NET framework. Due to this, server controls feature unique design considerations. The different types of controls are:

- HTML Server Controls - HTML elements are exposed to the server. They expose an object model that maps very closely to the HTML elements that they render

- ASP.NET Web Controls - Controls having more built-in features than HTML server controls. ASP.NET server controls include controls like buttons, text boxes and also special-purpose controls like calendar. ASP.NET server controls does not necessarily reflect HTML syntax

- Validation Controls - Controls that incorporate logic to test the user input. A validation control is attached to an input control to test data entered by the user for that input control. Validation controls allows to check for a required field, to test against a specific value or pattern of characters, between ranges, and so on

- User Controls - Controls that are created by the user as Web Forms pages. Web Forms User Controls can be embedded in other Web Forms pages and hence menus, toolbars, and other reusable elements can be created easily

HTML controls offer Web developers the power of the Web Forms page framework while retaining the familiarity and easier use of HTML elements. These controls look exactly like HTML, except they have a runat=”server” attribute/value pair in the opening tag of an HTML element. For example, the following HTML would not only create a text box on a Web page, but would also create an instance of the **HtmlInputText** server control:

```
<input id=”myTextBox” type=”text” runat=”server”>
```

Note: The runat and id properties are mandatory to all the HtmlServerControls.

Without runat=”server”, this line of HTML would be parsed into a standard text box. This model allows developers to import an HTML page authored by a Web designer and program against selected elements on the page. In addition, once an element is converted to a control, a Microsoft .NET Framework class is created for the element and its attributes are exposed as properties on the HTML control.

To enable programmatic referencing of the control, developers can include a unique id attribute. In the above example, the id=”myTextBox” defines this programmatic ID, allowing developers to manipulate the contents of this text box with events and other code that they write.

ASP.NET offers direct object model support for the most commonly used HTML elements. For those elements that are not directly supported, there is the HtmlGenericControl, which supports the <span>, <div>, <body>, and <font> elements, among others.

In addition, HTML controls offer the following features:

- A set of events for which handlers can be written in much the same way as it is done in a client-based form, except that the event is handled in the server code
- The ability to handle events in client script
- None of them emit ECMAScript to the client when responding to a request
- Automatic management of the values of the form’s controls. If the form makes a round trip to the server, HTML controls are automatically populated with the values they had when the form was submitted to the server

- Interaction with validation controls helps to verify that a user has entered appropriate information into a control
- Databinding to a single field, property, method, or expression (simple data binding) or a collection (list databinding)
- Support for HTML 4.0 styles
- Pass-through of custom attributes. The required attributes can be added to HTML control. The Web Forms framework will read them and render them without any change in functionality. This allows the user to add browser-specific attributes to the controls
- All controls that post back events must be nested within an HtmlForm control. For example, the following input elements would be parsed correctly as HTML server controls:

```
<form runat=”server” id=”myForm”>
   <input type=”text” id=”myTextBox” runat=”server”>
   <input type=”submit” id=”mySubmitButton”
      OnServerClick=”SubmitBtn_Click” runat=”server”>
</form>
```

- All HTML controls must be well formed and should not overlap. Unless noted, elements must be closed, either with an ending slash within the tag, or with a closing tag. For example,

```
<span id=”mySpan” runat=”server”/>
```

or

```
<span id=”mySpan” runat=”server”>Span text to be displayed here</span>
```

2.4 HTML Server-Side Controls

The various controls explained under this section are:

- HtmlInputButton
- HtmlInputFile
- HtmlInputImage
- HtmlInputText
- HtmlForm
- HtmlSelect
- HtmlTable
- HtmlTableCell
- HtmlTableRow
- HtmlTextArea
- HtmlInputCheckBox
- HtmlInputHidden
- HtmlInputRadioButton
- HtmlAnchor
- HtmlGeneric
- HtmlImage

The hierarchy of the controls is given in the following figure.

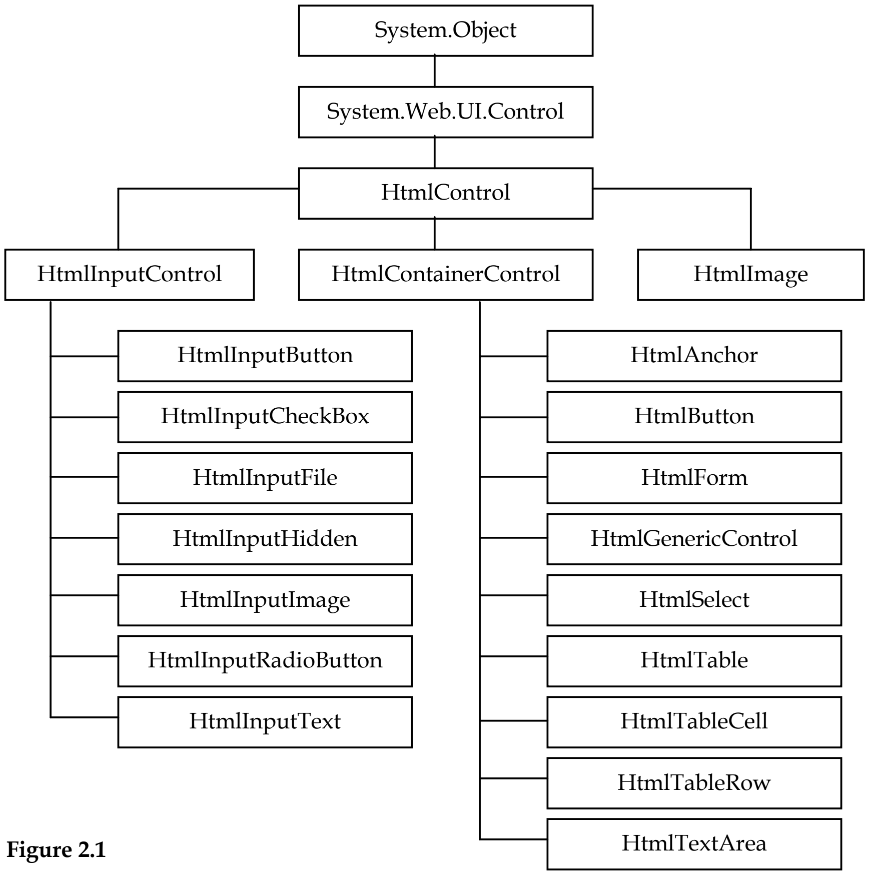
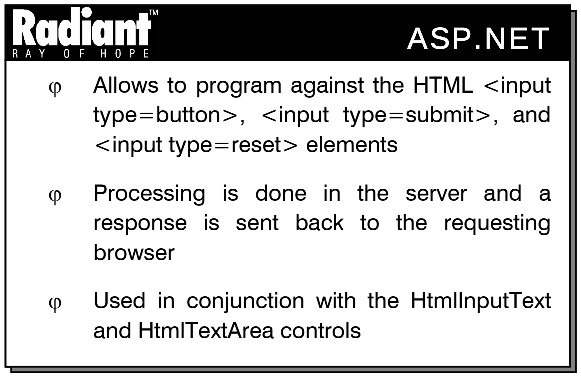


Figure 2.1

HtmlInputButton Control


Radiant™ RAY OF HOPE ASP.NET

- Allows to program against the HTML <input type=button>, <input type=submit>, and <input type=reset> elements
- Processing is done in the server and a response is sent back to the requesting browser
- Used in conjunction with the HtmlInputText and HtmlTextArea controls

When a user clicks on HTML Input Button control, input from the form that contains the embedded control is posted to the server. The processing is done in the server and a response is sent back to the requesting browser.

The HtmlInputButton is used in conjunction with the HtmlInputText and HtmlTextArea controls to create user input or authentication pages that can be processed on the server.

Syntax

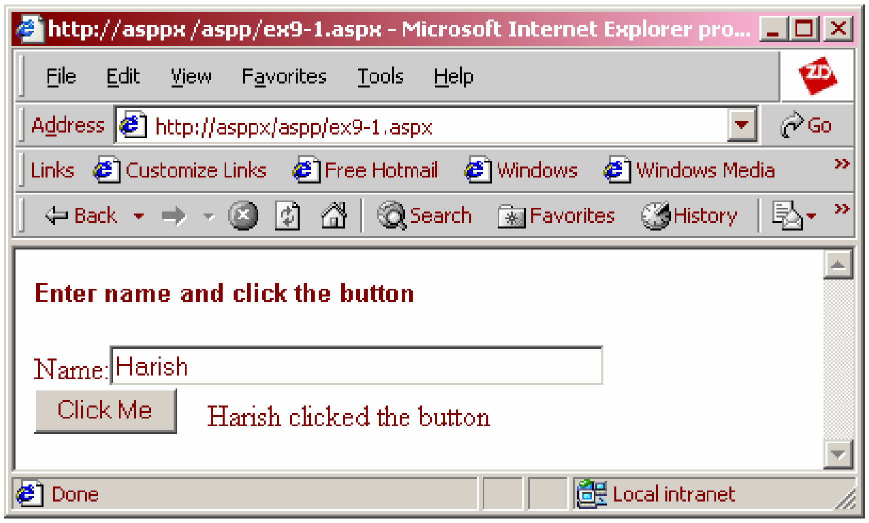
```
<input
   type=button | submit | reset
   runat="server"
   id="programmaticID"
   OnServerClick="OnServerClickhandler"
>
```

- **id** is the id of the button
- **OnServerClick** is the handler for the button click event

Practice 2.2

The following program displays a text box and a button and displays a message when the button is clicked.

```
<html>
<head>
<script language="C#" runat="server">
void Button1_Click(object Source, EventArgs e)
{
      Span1.InnerHtml= Name.Value + " clicked the button";
}
</script>
</head>
<body>
    <h5><font face="Arial">Enter name and click the button</font></h5>
    <form runat=server>
    <p> Name:<input id="Name" type=text size=40  runat="server"><br>
        <input type=button value="Click Me" OnServerClick="Button1_Click"
                  runat="server"> &nbsp;&nbsp;
        <span id=Span1 runat=server />
    </form>
</body>
</html>
```


http://asppx/aspp/ex9-1.aspx - Microsoft Internet Explorer pro...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-1.aspx Go
Links Customize Links Free Hotmail Windows Windows Media
Back Search Favorites History
Done
Local intranet

Enter name and click the button

Name: Harish

Click Me   Harish clicked the button

The above program displays a text box and a button using the input tag. When the button is clicked after entering the text in the textbox, the Button_Click1 function is called and the text is displayed using span.

HtmlInputFile Control

The HtmlInputFile control is used to design a page that allows uploading a file to a directory designated on the Web Server. The control allows to program against the HTML <input type=file> element. This control allows uploading of a binary or text file from a browser to the server. File upload is enabled in HTML.

Syntax

```
<input
   type=file
   runat="server"
   id="programmaticID"
   accept="MIMEencodings"
   maxlength="maxfilepathlength"
   size="widthoffilepathtextbox"
   postedfile="uploadedfile"
>
```

- **type** is the file type (text, binary, etc)
- **postedfile** is the name of the file that is uploaded
- **maxlength** is the maximum length of the file path

Practice 2.3

The following program illustrates how to upload a file to the Web server.

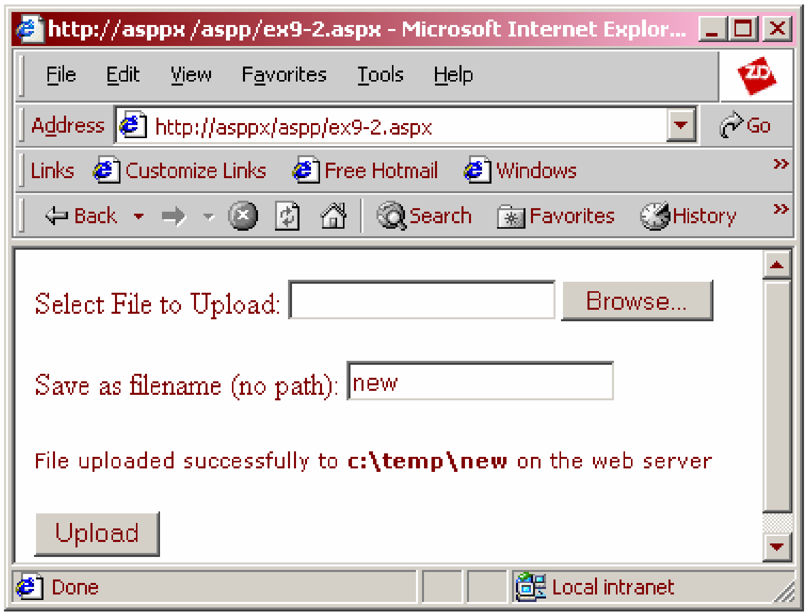
```
<html>
<head>
<script language="C#" runat="server">
void Button1_Click(object Source, EventArgs e)
{
    if (Text1.Value == "")
    {
         Span1.InnerHtml = "Please enter a file name";
         return;
    }
    if (File1.PostedFile != null)
    {
         try
         {
            File1.PostedFile.SaveAs("d:\\files\\"+Text1.Value);
            Span1.InnerHtml = "File uploaded successfully to
               <b>c:\\temp\\"+Text1.Value+"</b> on the web server";
         }
         catch (Exception exc)
         {
                Span1.InnerHtml = "Error saving file
            <b>c:\\temp\\"+Text1.Value+"</b><br>"+ exc.ToString();
         }
     }
}
```

```
</script>

</head>
<body>

  <form enctype=”multipart/form-data” runat=”server”>
  Select File to Upload: <input id=”File1" type=file runat=”server”>
    <p> Save as filename (no path): <input id=”Text1" type=”text”
           runat=”server”>
    <p> <span id=Span1 style=”font: 8pt verdana;” runat=”server” />
    <p> <input type=button id=”Button1" value=”Upload”
           OnServerClick=”Button1_Click” runat=”server”>
  </form>

</body>
</html>
```


http://asppx /aspp/ex9-2.aspx - Microsoft Internet Explor...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-2.aspx Go
Links Customize Links Free Hotmail Windows
Back Search Favorites History
Browse...
Upload
Done
Local intranet

Select File to Upload:

Save as filename (no path): new

File uploaded successfully to **c:\temp\new** on the web server

The above program prompts the user to select a file to upload to the Web server. It requires a new filename and if it is not entered, an error message is displayed. The program also displays an error message if there is a problem in saving the file. When the file is uploaded successfully a message is displayed.

HtmlInputImage Control

The HtmlInputImage control allows to program against the HTML <input type=image> element. This control is used in conjunction with the **HtmlInputText**, **HtmlTextArea**, and other controls to construct user input forms. Since this control is run on the server, it offers the same customizations for the button like HTML. This control offers greater support for earlier browsers than the HtmlButton control.

Syntax

```
<input
```

```
   type=image
   runat=”server”
   id=”programmaticID”
   src=”imagepath”
   align=”imagealignment”
   alt=”alttext”
   OnServerClick=”OnServerClickhandler”
   width=”borderwidthinpixels”
>
```

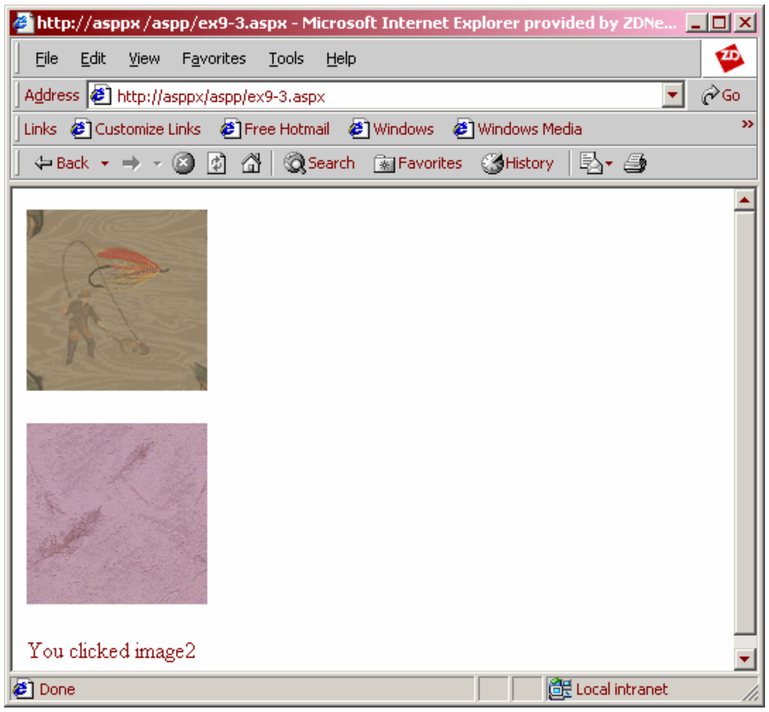
- **src** is the source path of the image file
- **align** is the alignment of the image in the browser
- **alt** is the alternate text that has to be displayed in case the image file is absent in the specified path

Practice 2.4

The following program displays two images and responds to the mouse move and mouse click events.

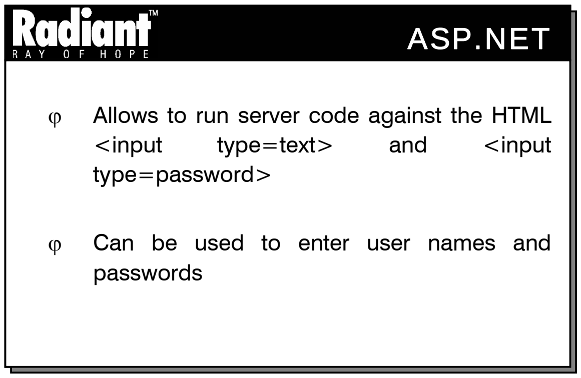
```
<head>
<script language=”C#” runat=”server”>
   void Button1_Click(object Source, ImageClickEventArgs e)
   {
      Message.InnerHtml=”You clicked image1";
   }
   void Button2_Click(object Source, ImageClickEventArgs e)
   {
      Message.InnerHtml=”You clicked image2";
   }
</script>
</head>
<form runat=”server”>
  <p><input type=image id=”InputImage1" src=”d:/images/one.bmp”
             OnServerClick=”Button1_Click” runat=”server”>
  <p><input type=image id=”InputImage2" src=”d:/images/one.bmp”
             onmouseover=”this.src=’d:/images/two.bmp’;”
onmouseout=”this.src=’d:/images/one.bmp’;” OnServerClick=”Button2_Click”
runat=”server”>
  <p><span id=”Message” runat=”server”/>
</form>
```

In the above program, two images are loaded using the Html-input control. When the mouse cursor is moved over the second image, the image changes to another image using the **onmouseover**. When the mouse is moved out of the second image, the original image is loaded using **onmouseout**. When either of the images is clicked, a message is displayed indicating the image that is clicked. This is handled by the OnServerClick event.


http://asppx /aspp/ex9-3.aspx - Microsoft Internet Explorer provided by ZDNe...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-3.aspx
Go
Links Customize Links Free Hotmail Windows Windows Media
Back Search Favorites History
Done
Local intranet

You clicked image2

HtmlInputText Control


Radiant RAY OF HOPE
ASP.NET

φ Allows to run server code against the HTML <input type=text> and <input type=password>

φ Can be used to enter user names and passwords

This control can be used in conjunction with the **HtmlInputButton**, **HtmlInputImage**, or **HtmlButton** control to process user input on the server. Values can be assigned and retrieved to the properties HtmlInputText MaxLength, Size, and Value.

Note : This control does not require a closing tag.

Syntax

```
<input
   type=text | password
   runat="server"
   id="programmaticID"
   maxlength="max#ofcharacters"
   size="widthoftextbox"
   value="defaulttextboxcontents"
```

>

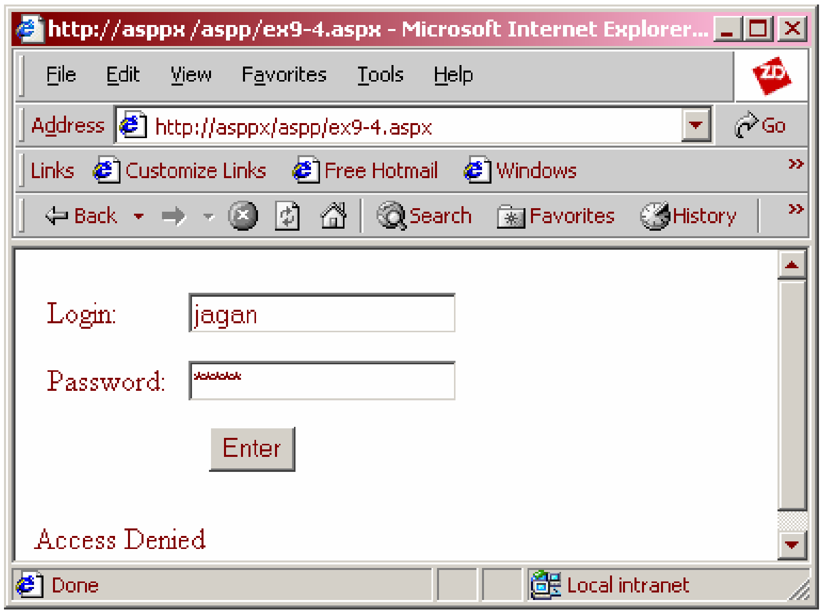
- **type** specifies whether the input characters are for an ordinary field or for a password field. If set to password, then an asterisk (*) is displayed instead of the character which is typed to preserve secrecy
- **maxlength** attribute specifies the maximum length of the text that can be typed in the text box

Practice 2.5

The following program checks the authentication of the user.

```
<head>
<script language="C#" runat="server">
void SubmitBtn_Click(object Source, EventArgs e)
{
   if (Password.Value == "ASP.NET")
      Message.InnerHtml="Access Granted";
   else
      Message.InnerHtml="Access Denied";
}
</script>
</head>

<body>
   <form runat="server">
   <table cellpadding=5>
   <tr>
      <td>Login: </td>
      <td><input id="Name" runat="server"></td>
   </tr>
   <tr>
      <td>Password: </td>
      <td><input id="Password" type=password runat="server"></td>
   </tr>
   <tr>
      <td colspan=2 align="center"><input type=submit value="Enter"
           OnServerClick="SubmitBtn_Click" runat="server"> </td>
   </tr>
   </table>
   <p><span id="Message" runat="server"/>
</form>
</body>
```


http://asppx /aspp/ex9-4.aspx - Microsoft Internet Explorer...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-4.aspx
Go
Links Customize Links Free Hotmail Windows
Back Search Favorites History
Enter
Done
Local intranet

Login: jagan
Password: *****
Access Denied

In the above program, two text boxes are displayed to enter the user name and password. If the password entered is ASP.NET, then the message **Access Granted** is displayed, otherwise the message **Access Denied** is displayed.

HtmlForm Control

The HtmlForm control allows to program against the HTML <form> element. All Web Form controls, whether HTML, Web, pagelet or custom, must be nested between well-formed opening and closing tags of the HtmlForm control in order to take advantage of the page framework's postback services.

Note: More than one HtmlForm control cannot be included on a single Web Forms page.

By default, the method attribute of the HtmlForm control is set to POST, and the action attribute is set to the URL of the source page. These attributes can be customized to suit the needs of an individual. However, this can break the built-in view state and postback services provided by the Web Forms Page Framework.

Syntax

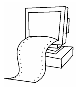
```
<form
   runat="server"
   id="programmaticID"
   method=POST | GET
   action="srcpageURL"
>
Other controls, input forms, and so on.
</form>
```

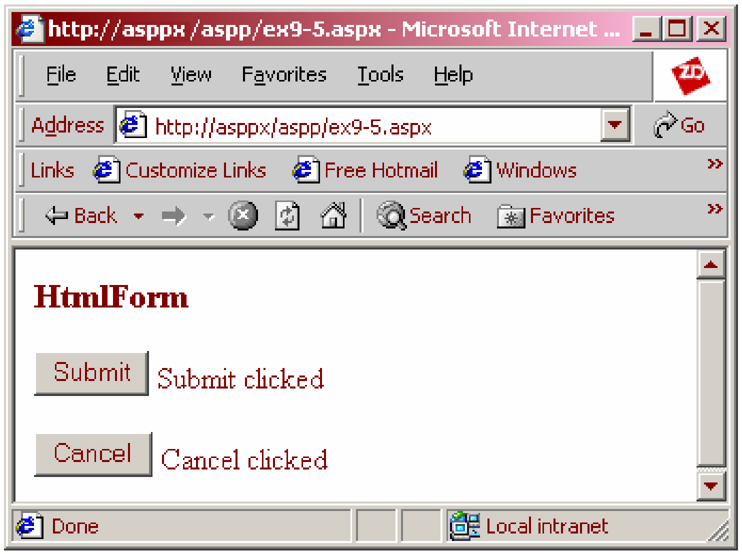
- **method** is the way in which the form is displayed
- **action** is the URL source

Practice 2.6

The following program illustrates the use of forms.

```
<head>
<script language="C#" runat="server">
void Button1_OnClick(object Source, EventArgs e)
{
   Span1.InnerHtml="Submit clicked";
}
void Button2_OnClick(object Source, EventArgs e)
{
   Span2.InnerHtml="Cancel clicked";
}
</script>
</head>
<body>
<div class="header"><h3>HtmlForm</h3></div>
<form id=ServerForm runat="server">
   <p><button id=Button1 runat="server" OnServerClick="Button1_OnClick">
Submit</button>
      <span id="Span1" runat="server" />
   <p><button id=Button2 runat="server" OnServerClick="Button2_OnClick">
           Cancel</button>
      <span id=Span2 runat="server" />
</form>
</body>
```




http://asppx /aspp/ex9-5.aspx - Microsoft Internet ...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-5.aspx
Go
Links Customize Links Free Hotmail Windows
Back Search Favorites
Done
Local intranet

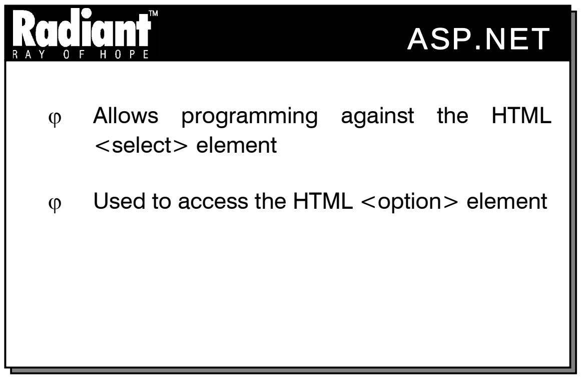
HtmlForm

Submit Submit clicked

Cancel Cancel clicked

The above program displays two buttons in a form that has the id **ServerForm**. The form has two buttons namely Submit and Cancel. When the buttons are clicked, the appropriate messages appear on the right side of the button.

HtmlSelect Control


Radiant™ RAY OF HOPE ASP.NET

φ Allows programming against the HTML <select> element

φ Used to access the HTML <option> element

Syntax

```
<select
   runat="server"
   id="programmaticID"
   OnServerChange="onserverchangehandler"
   DataSource="databindingsource"
   DataTextField="fieldtodatabindoptionttext"
   DataValueField="fieldtodatabindoptionvalue"
   MultipleItems="collectionofoptionelements"
   SelectedIndex="indexofcurrentlyselecteditem"
   Size="#ofvisibleitems"
   Value="currentitemvalue"
>
<option>value1</option>
<option>value2</option>
</select>
```

- **OnServerChange** is the handler for the select control
- **DataSource** is the source from which the items for the select list are to be taken
- **DataTextField** specifies the text that has to be displayed in the select list
- **DataValueField** specifies the value corresponding to the **DataTextField**
- **MultipleItems** option specifies whether multiple items can be selected
- **SelectedIndex** is the index value of the selected item

Practice 2.7

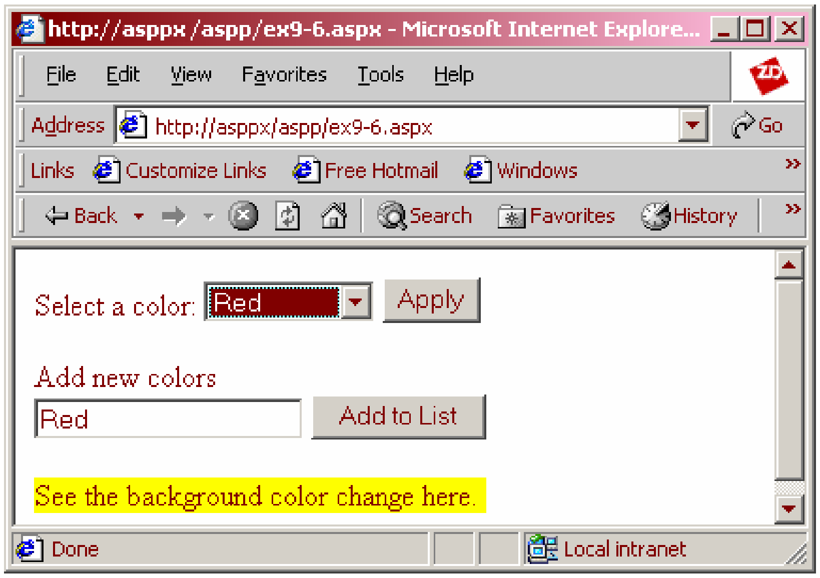
The following program creates a selectable drop-down list and allows to choose an option.

```
<head>
<script language="C#" runat="server">

void Apply_Click(object Source, EventArgs e)
{
   Message.Style["background-color"]=ColorSelect.Value;
}
void AddToList_Click(object Source, EventArgs e)
{
   ColorSelect.Items.Add(Text1.Value);
}

</script>
</head>

<body>
  <form runat="server">
   <p>Select a color:
   <select id="ColorSelect" runat="server">
      <option>SkyBlue</option>
      <option>LightGreen</option>
      <option>Blue</option>
      <option>Yellow</option>
   </select>
   <input type="button" runat="server" Value="Apply"
            OnServerClick="Apply_Click">
    <p>Add new colors<br>
   <input type="text" id="Text1" runat="server">
   <input type="button" runat="server" Value="Add to List"
            OnServerClick="AddToList_Click">
 <p><span id="Message" runat="server"> See the background color change here.
   </span>
 </form>
</body>
```


http://asppx /aspp/ex9-6.aspx - Microsoft Internet Explore...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-6.aspx
Go
Links Customize Links Free Hotmail Windows
Back Search Favorites History
Select a color: Red
Apply
Add new colors
Red
Add to List
See the background color change here.
Done
Local intranet

The above program displays a listbox of colors. On selecting an option from the list and clicking the **Apply** button, the color of the text at the bottom of the page is changed. If a new color is to be added to the list, the name is typed in the text box and the **Add to List** button is clicked.

HtmlTable Control

The **HtmlTable** control allows to program against the HTML <table> element. A table control can be dynamically bound to add rows and cells to the table using the methods provided by the **HtmlTableRowCollection** and **HtmlTableCellCollection**.

Syntax

```
<table runat=”server”
   id=”accessID”
   align=left | center | right
   bgcolor=”bgcolor”
   border=”borderwidthinpixels”
   bordercolor=”bordercolor”
   cellpadding=”spacingwithincellsinpixels”
   cellspacing=”spacingbetweencellsinpixels”
   height=”tableheight”
   rows=”collectionofrows”
   width=”tablewidth”
>
</table>
```

HtmlTableCell Control

The HtmlTableCell control allows programming against the HTML <td> and <th> elements. Cell can be added dynamically to an HtmlTableRow control, in response to control events or by binding an HtmlTable control to the entries in a data source.

Syntax

```
<td or th
   runat=”server”
   id=”programmaticID”
   align=”alignmentofcontentincell”
   bgcolor=”bgcolor”
   bordercolor=”bordercolor”
   colspan=”#ofcolscellspans”
   height=”tableheight”
   nowrap=”True | False”
   rowspan=”#ofrowscellspans”
   valign=”vertalignmentofcellcontent”
   width=”cellwidth”
>
CellContent
</td or /th>
```

HtmlTableRow Control

Creates a table row control. The **HtmlTableRow** control allows you to program against the HTML <tr> element. Rows can be dynamically added to an HtmlTable control, whether in response to control events or through binding an **HtmlTable** control to the entries in a data source.

Syntax

```
<tr runat=”server”
   id=”accessID”
   align=”tablecontentalignment”
   bgcolor=”tablebgcolor”
   bordercolor=”bordercolor”
   height=”tableheight”
   cells=”collectionoftablecells”
   valign=”verticalalignmentofrowcontent”
>
  <td>cellcontent</td>
</tr>
```

Practice 2.8

The following program illustrates the use of table, tablecell and tablerow.

```
<head>
<script language=”C#” runat=”server”>

void Page_Load(Object sender, EventArgs e)
{
   int row=0;
   int numrows=int.FromString(Select1.Value);
   int numcells=int.FromString(Select2.Value);
   for (int j=0; j<numrows; j++)
   {
      HtmlTableRow r=new HtmlTableRow();
      if (row%2==  1)
         r.BgColor=”Gainsboro”;
         row++;
         for (int i=0; i<numcells; i++)
         {
              HtmlTableCell c=new HtmlTableCell();
        c.Controls.Add(new LiteralControl(j.ToString() + “, “ +
                            i.ToString()));
              r.Cells.Add(c);
         }
         Table1.Rows.Add(r);
   }
}
</script>
</head>
```

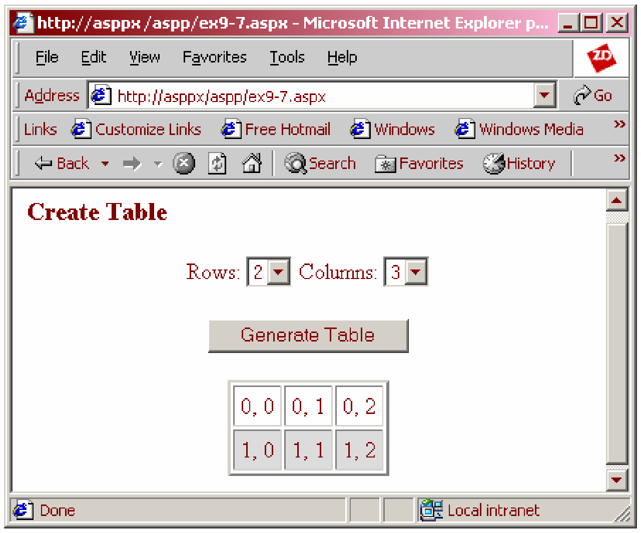
```
<body>
<div class="header"><h3>Create Table</h3></div>
<div align="center">
<form runat="server">
<table>
   <tr>
      <td>Rows:</td>
      <td>
       <select id="Select1" runat="server">
         <option Value="2">2</option>
         <option Value="3">3</option>
         <option Value="4">4</option>

       </select></td>

      <td>Columns:</td>
      <td>
       <select id="Select2" runat="server">
         <option Value="2">2</option>
         <option Value="3">3</option>
         <option Value="4">4</option>
       </select></td></tr>
   </table>
<p><input type="submit" runat="server" value="Generate Table" > </form>
<p><table id="Table1" runat="server" CellPadding=4 Border=2 /></p>

</div>
</body>
</html>
```

The above example uses the Add function to add cells and rows to a table dynamically. The number of rows and columns of the table are selected from the two drop down list boxes, one for the number of rows of the table and the other for the number of columns.


http://asppx/aspp/ex9-7.aspx - Microsoft Internet Explorer p...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-7.aspx Go
Links Customize Links Free Hotmail Windows Windows Media
Back Search Favorites History
Done
Local intranet

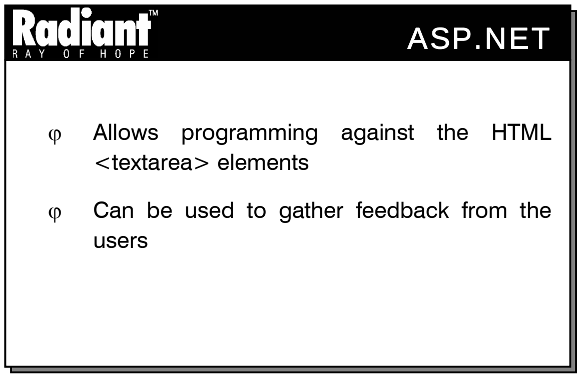
Create Table

Rows: 2 Columns: 3

Generate Table

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2

HtmlTextArea Control



Radiant RAY OF HOPE ASP.NET

- Allows programming against the HTML <textarea> elements
- Can be used to gather feedback from the users

Syntax

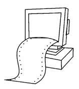
```
<textarea
   runat="server"
   id="programmaticID"
   cols="numberofcolsintextarea"
   name="namepassedtobrowser"
   rows="numberofrowsintextarea"
   OnServerChange="onserverchangehandler"
>
textareacontent
</textarea>
```

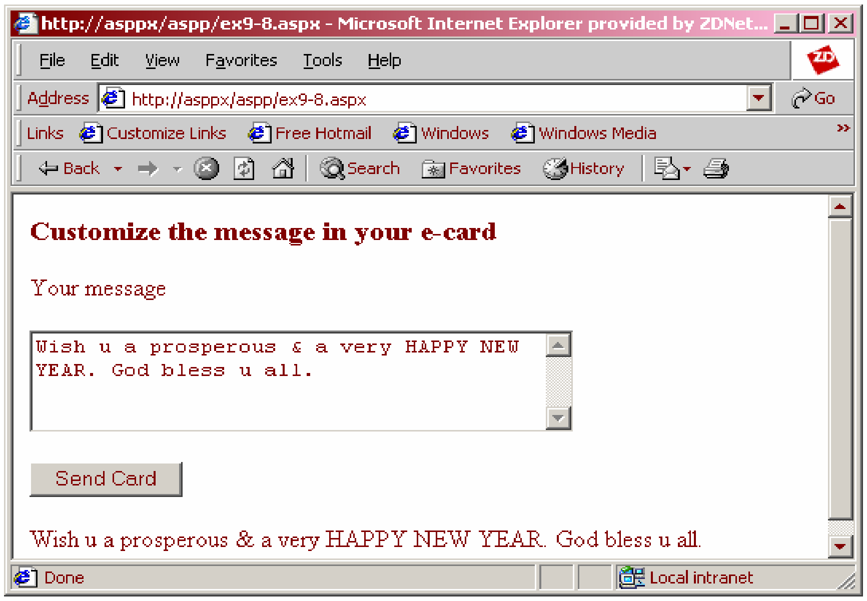
- **cols** indicates the width of the control
- **rows** indicates the height of the control
- **OnServerChange** is the handler for the control

Practice 2.9

The following program displays a text area.

```
<head>
<script language="C#" runat="server">
void SubmitBtn_Click(Object sender, EventArgs e)
{
   Message.InnerHtml= TextArea1.Value;
}
</script>
</head>
<H3>Customize the message in your e-card</H3>
<body>
  <form runat="server">
     <p>Your message</p>
     <textarea id="TextArea1" runat="server" cols=40 rows=4 />
     <p><input type=submit value="Send Card" runat="server"
OnServerClick="SubmitBtn_Click">
     <p><span id="Message" runat="server" />
  </form>
</body>
```




http://asppx/aspp/ex9-8.aspx - Microsoft Internet Explorer provided by ZDNet...
File Edit View Favorites Tools Help
Address http://asppx/aspp/ex9-8.aspx
Go
Links Customize Links Free Hotmail Windows Windows Media
Back Search Favorites History
Send Card
Done
Local intranet

Customize the message in your e-card

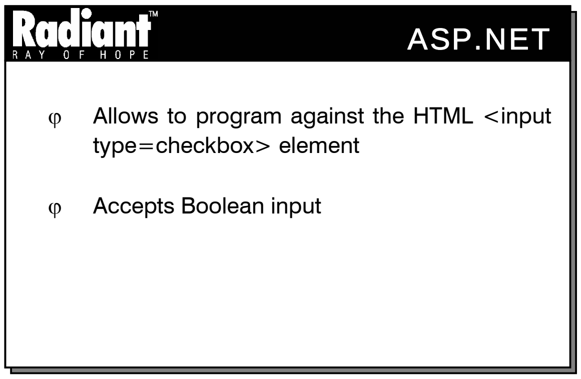
Your message

```
Wish u a prosperous & a very HAPPY NEW
YEAR. God bless u all.
```

Wish u a prosperous & a very HAPPY NEW YEAR. God bless u all.

In the above program, a text area is displayed along with a button. On clicking the button, after entering text in the text area, the typed text is displayed below the button.

HtmlInputCheckBox Control


Radiant RAY OF HOPE
ASP.NET

- Allows to program against the HTML <input type=checkbox> element
- Accepts Boolean input

When checked, its **Checked** property returns true. **HtmlInputCheckBox** is typically used with other user-input controls, such as the HtmlInputButton control, to determine whether a check box is selected. This is done by evaluating the **Checked** property on this control.

Syntax

```
<input type="radio"
   runat="server"
   id="accessID"
   checked
   name="radiobuttongroup"
>
```

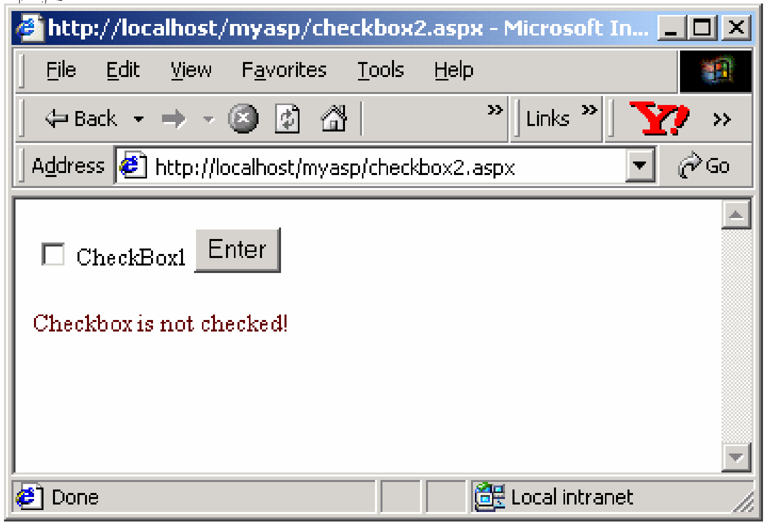
The **HtmlCheckBox** control does not have an event that notifies the server when a user has selected the control. Checkbox can be used with one of the button controls — such as the HtmlInputButton,

Practice 2.10

HtmlInputImage, or HtmlButton — to cause a postback that will allow to program the **HtmlCheckBox** control itself.

The following example illustrates the use of HtmlInputCheckBox control. When a user clicks the input button included on the form, the button's click event handler evaluates the value of the **HtmlInputCheckBox** control's **Checked** property, then displays a message in a span control that depends on the value of the property.

```
<head>
<script language=”C#” runat=”server”>
void Button1_Click(object Source, EventArgs e) {
   if (Check1.Checked==  true) {
      Message.InnerHtml=”Checkbox is checked!”;
   }
   else {
     Message.InnerHtml=”Checkbox is not checked!”;
   }
}
</script>
</head>
<form runat=”server”>
   <input id=”Check1" type=checkbox runat=”server”> CheckBox1
   <input type=button id=”Button1" value=”Enter”
      OnServerClick=”Button1_Click” runat=”server”>
<p><span id=”Message” style=”color:#600" runat=”server” />
</form>
```


http://localhost/myasp/checkbox2.aspx - Microsoft In...
File Edit View Favorites Tools Help
Back
Links
Address http://localhost/myasp/checkbox2.aspx
Go
CheckBox1
Enter
Done
Local intranet

Checkbox is not checked!

In the above program a check box and a button are used. In the OnServerClick event of the button, we check, whether the checkbox is checked or not. It also prints whether the checkbox is checked or unchecked.

HtmlInputRadioButton

The **HtmlInputRadioButton** control creates a single radio button input field. By default, each individual radio button can be selected.

Syntax

```
<input type=”radio”
   runat=”server”
   id=”controlID”
   checked
   name=”radiobuttongroup”
>
```

This control does not require an opening and closing tag. The **HtmlRadioButton** control does not have an event that notifies the server when a user has selected one of the buttons.

A set of radio buttons can be grouped together in cases where only one button from the set of options needs to be selected. This is done by setting a common value for the **Name** attribute on each radio button in the group. For example, the following block of code creates a group of radio buttons in which only one radio button may be selected.

```
<input type=radio name=”optgroup1">Red
<input type=radio name=”optgroup1">White
<input type=radio name=”optgroup1">Blue
```

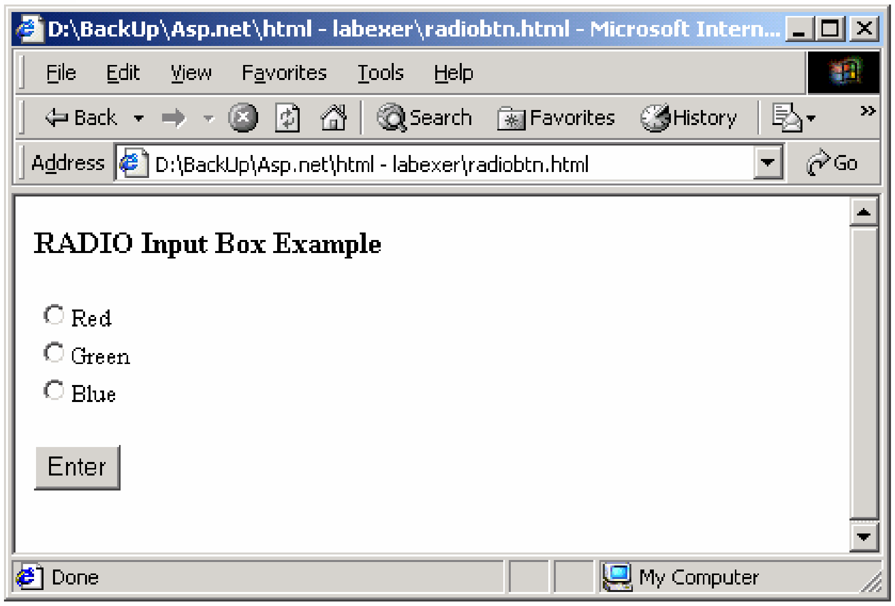
However, the selected state must be tested on the individual radio buttons. **HtmlInputRadioButton** allows to program against the HTML <input type=radio> element.

Practice 2.11

The sample given below illustrates the use of HtmlInputRadioButton control.

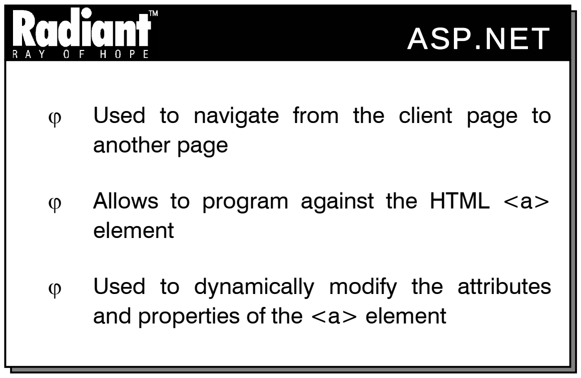
```
<form runat=”server”>
  <h3>RADIO Input Box  Example</h3>
   <input type=”radio” id=”Red” name=”Mode” runat=”server”/>Red<br>
<input type=”radio” id=”Green” name=”Mode” runat=”server”/>Green<br>
   <input type=”radio” id=”Blue” name=”Mode” runat=”server”/>Blue
   <p><input type=button id=”Button1" value=”Enter”
            OnServerClick=”Button1_Click” runat=”server”>
   <p><span id=”Message” runat=”server” />

</form>
<script language=”C#” runat= “server”>
void Button1_Click(object Source, EventArgs e) {
   if (Red.Checked==  true)
      Message.InnerHtml=”Red is checked”;
   else if (Green.Checked==  true)
      Message.InnerHtml=”Green is checked”;
   else if (Blue.Checked==true)
      Message.InnerHtml=”Blue is checked”;
}
</script>
```


D:\BackUp\Asp.net\html - labexer\radiobtn.html - Microsoft Intern...
File Edit View Favorites Tools Help
Back Search Favorites History
Address D:\BackUp\Asp.net\html - labexer\radiobtn.html Go
RADIO Input Box Example
Red
Green
Blue
Enter
Done
My Computer

To respond to user selection of radio button controls, a set of radio button controls are declared. A button control causes postback when clicked by the user. A span control is used here to render the postback message. Event-handling code is included to process the information gathered from the radio button group.

HtmlAnchor Control



Radiant™ RAY OF HOPE ASP.NET

- Used to navigate from the client page to another page
- Allows to program against the HTML <a> element
- Used to dynamically modify the attributes and properties of the <a> element

The **HtmlAnchor** control is used to navigate from the client page to another page, or to another location within the same page.

This control can be generated by using a data source. Control events can be used dynamically to generate this control.

Syntax

```
<a runat="server"
   id="accessID"
   href="linkurl"
   name="bookmarkname"
   OnServerClick="OnServerClickhandler"
   target="linkedcontentframeorwindow"
   title="titledisplayedbybrowser"
>
   linktext
</a>
```

Target values must begin with a letter, except when mentioning the following special values that begin with an underscore: _blank, _self, _parent, and _top. This control requires an opening and closing tag.

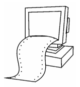
Note: Remember to embed the **HtmlAnchor** control inside the opening and closing tags of an HtmlForm control.

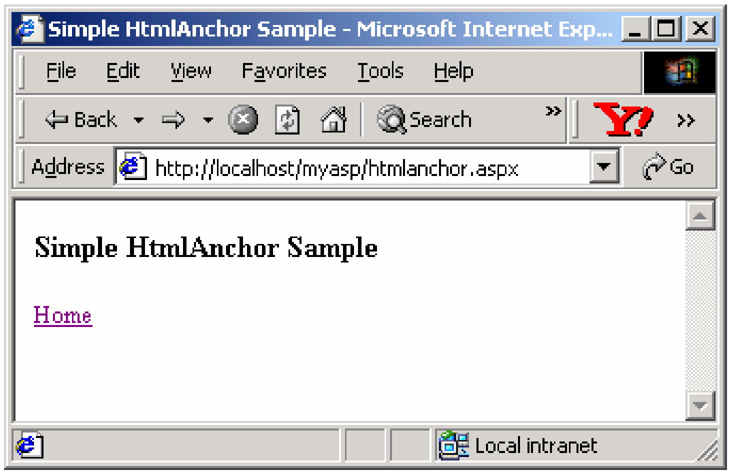
The hyperlinks can be generated on a Web Forms page by binding the **HtmlAnchor** control to a data source.

Practice 2.12

The following example shows how to include an **HtmlAnchor** control and to bind the address of the links (URLs), as well as to link the text in order to display for the controls.

```
<html>
<head>
<title>Simple HtmlAnchor Sample</title>
<script language="C#" runat=server>
void Page_Load(Object sender, EventArgs e)
{
   anchor1.HRef="/";
}
</script>
</head>
    <body>
        <h3>Simple HtmlAnchor Sample</h3>
        <form runat=server>
           <p><a id=anchor1 runat="server">Home</a>
        </form>
    </body>
</html>
```




Simple HtmlAnchor Sample - Microsoft Internet Exp...
File Edit View Favorites Tools Help
Back Search
Address http://localhost/myasp/htmlanchor.aspx Go
Local intranet

Simple HtmlAnchor Sample

Home

In this example the anchor control is used to load another page. The HRef refers to Index.html and when the anchor is clicked, the index.html page is loaded in the browser.

Generating the HRef Attribute Dynamically

Declare an **HtmlAnchor** control in the BODY of an HTML document, as shown below.

```
<form runat="server">
   <p><a id=anchor1 runat="server">Home</a>
</form>
```

An event-handling code that assigns a URL to the HtmlControl HRef property has to be writen. The following code generates a URL that will let the user navigate the site's home page when the **Page_Load** event occurs.

```
<script language="C#" runat="server">
   void Page_Load(Object sender, EventArgs e) {
      anchor1.HRef="/";
   }
</script>
```

HtmlGenericControl

The **HtmlGenericControl** provides a server control implementation for all other HTML server control tags that are not directly represented by a specific HTML server control. These include the <span>, <body>, <div>, and <font> elements, among others.

Syntax

```
<span | body | div | font | others
   runat="server"
   id="accessID"
>
element content here
</span | body | div | font | others>
```

An **HtmlGenericControl** is created on the server in response to tags that include the runat="server" attribute in elements that do not map directly to a specific HTML control. The control maps the name of the tag of the particular element to be used as an HTML control to the page framework through the

TagName property. This control inherits the functionality from the HtmlContainerControl class, which allows to dynamically change the inner content of HTML control tags.

The following samples illustrate the use of **HtmlGenericControl**.

Assign an **id** attribute to the form which enables the programmatic access to the control.

```
<body id=Body runat=server>
```

Declare an **HtmlInputButton** control and assign it an **id** attribute to allow programmatic access to the control. Here, we declare an **OnServerClick** designate to instruct the framework about the function when the button is clicked as given below.

```
<input type="submit" runat="server" Value="Apply"
   OnServerClick="SubmitBtn_Click">
```

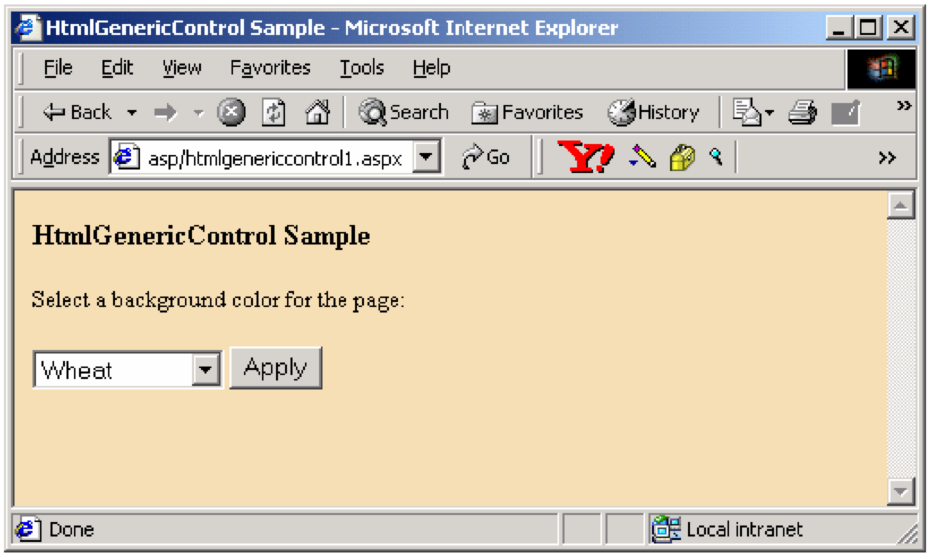
In the script declaration block at the head of the page, create event-handling code that manipulates the value entered by the user in the option to display the color in the background

```
void SubmitBtn_Click(object Source, EventArgs e) {
   Body.Attributes["bgcolor"]=ColorSelect.Value;
}
```

Practice 2.13

The following example shows how an **HtmlGenericControl** is used to modify a background color of the page. The sample also demonstrates how to use the AttributeCollection class to programmatically access the attributes that can be declared on any HTML control, in this case, the HTML BODY element.

```
<html>
<head>
<script language="C#" runat="server">
void SubmitBtn_Click(object Source, EventArgs e) {
   Body.Attributes["bgcolor"]=ColorSelect.Value;
}
</script>
</head>
<body id=Body runat="server">
<div class="header"><h3>HtmlGenericControl Sample</h3></div>
<form runat="server">
   <p>Select a background color for the page: <p>
   <select id="ColorSelect" runat="server">
      <option>White</option>
      <option>Wheat</option>
      <option>Red</option>
      <option>Green</option>
   </select>
   <input type="submit" runat="server"
      Value="Apply" OnServerClick="SubmitBtn_Click">
</form>
</body>
</html>
```


HtmlGenericControl Sample - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address asp/htmlgenericcontrol1.aspx Go
Done
Local intranet

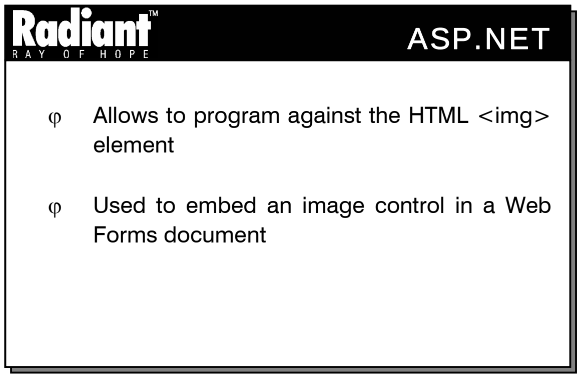
HtmlGenericControl Sample

Select a background color for the page:

Wheat Apply

The above program uses an HtmlGenericControl to display a list box and a button. When a color is selected and the Apply button is clicked, the background color of the form changes accordingly.

HtmlImage Control


Radiant
RAY OF HOPE
ASP.NET

φ Allows to program against the HTML <img> element

φ Used to embed an image control in a Web Forms document

The **HtmlImage** control allows to program against the HTML <img> element. This control allows to dynamically set and retrieve image attributes that include the **src**, **width**, **height**, **border**, **alt**, and **align** attributes.

This control embeds an image control in a Web Forms document.

Syntax

```
<img runat="server"
   id="accessID"
   alt="alttext"
   align= top | middle | bottom | left | right
   border="borderwidth"
   height="imageheight"
   src="imageURI"
   width="imagewidth"
>
```

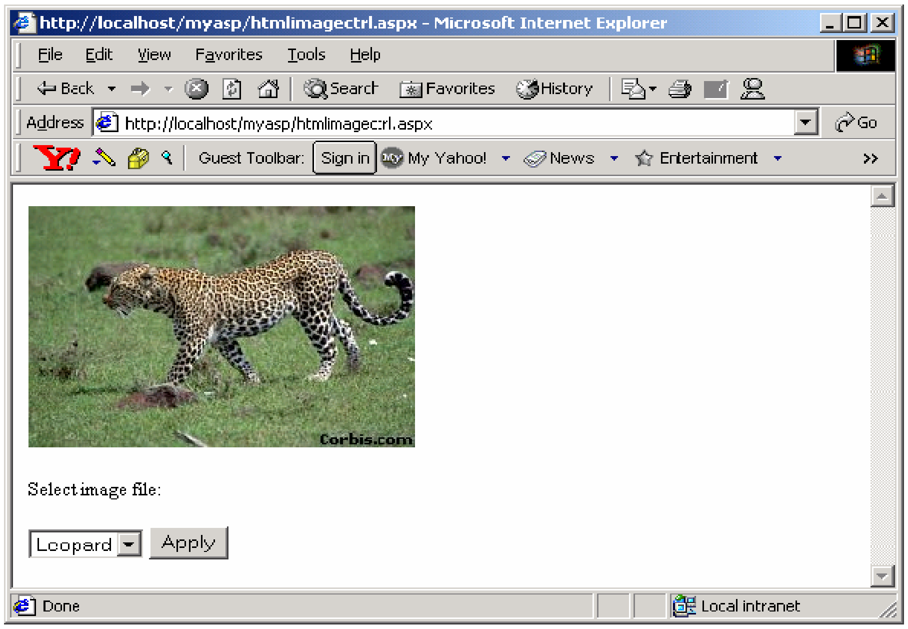
Working with HtmlImage

The **HtmlImage** server control renders an image file specified by its **Src** property. The **HtmlImage** control allows to program against the HTML <img> element, enabling developers to dynamically set and retrieve image attributes, including the image file's **src**, **width**, **height**, **border**, **alt**, and **align** attributes.

Practice 2.14

The following example shows how to dynamically change a displayed image in a Web Forms page based on the choice of the user.

```
<html>
<head>
<script language="C#" runat="server">
void SubmitBtn_Click(object Sender, EventArgs e) {
   Image1.Src="d:/myprog/asp.net/Images/" + Select1.Value;
}
</script>
</head>
<body>
<form runat="server">
<img ID="Image1" src="d:/myprog/asp.net/images/bobcat.jpg" runat="server"/>
   <p>Select image file: </p>
   <select id="Select1" runat="server">
      <option Value="Bobcat.jpg">Bobcat</option>
      <option Value="Cheetah.jpg">Cheetah</option>
      <option Value="Leopard.jpg">Leopard</option>
      <option Value="Lynx.jpg">Lynx</option>
      <option Value="Tiger.jpg"> Tiger</option>
</select>
   <input type="submit" runat="server" Value="Apply"
      OnServerClick="SubmitBtn_Click">
</form>
</body>
</html>
```


http://localhost/myasp/htmlimagectrl.aspx - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost/myasp/htmlimagectrl.aspx
Go
Guest Toolbar: Sign in My Yahoo! News Entertainment
Corbis.com
Done
Local intranet

Select image file:

Leopard Apply

This example shows how to change a displayed image based on the choice of the user. The example uses the click event of the HtmlInputButton control to trigger a handler that specifies the path of the image file to be displayed, in this case, from the application's **images** directory. Notice that the SubmitBtn_Click event handler is simple, specifying the **d:/.../images** directory as the source path for the images to be displayed. An HtmlSelect control provides the list of option for the user. The selected value in the HtmlSelect control in the Web Forms page determines which image to display from the absolute path specified.

2.5 Short Summary

- In ASP.NET, all the functions and subroutines must be placed within the <script> section
- Server controls are specifically designed to work with Web Forms
- The HtmlInputButton is used to display a command button
- HtmlInputText control is used to display a text box
- Tables can be dynamically bound to add rows and cells to the table using the methods provided by the HtmlTableRowCollection and HtmlTableCellCollection
- HtmlAnchor is used to navigate from the client page to another page, or to another location within the same page
- The HtmlGenericControl provides implementation for other HTML server control tags not directly represented by a specific HTML server control
- The HtmlImage server control renders an image file specified by the SRC property

2.6 Brain Storm

1. What are the languages supported by ASP.NET?
2. What is the significance of runat = “server”?
3. How can the text box be made to accept passwords?