2020

Programming With ASP.Net

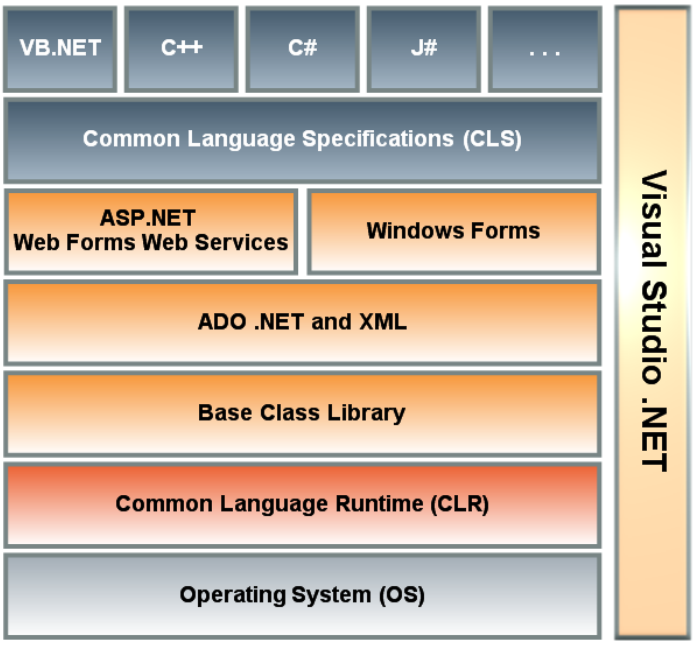
BCA / BSc.I.T Semester - 5

ASP.NET is an open-source, server-side web-application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, applications and services.

Prof. Pranav Trivedi
Geetanjali College

Chapter-1: Framework And Web Contents

1.1 Overview of .Net Framework :



- You should understand that the .NET Framework is really a cluster of several technologies:

- **The .NET languages:** These include **C# and VB .NET** (Visual Basic .NET), the object oriented and modernized successor to Visual Basic 6.0; these languages also include JScript .NET (a server-side version of JavaScript), J# (a Java clone), and C++ with Managed Extensions.
- **The CLR (Common Language Runtime):** The CLR is the engine that executes all .NET programs and provides automatic services for these applications, such as security checking, memory management, and optimization.
- **The .NET Framework class library:** The class library collects thousands of pieces of prebuilt functionality that you can "snap in" to your applications. These features are sometimes organized into technology sets, such as ADO.NET (the technology for creating database applications) and Windows Forms (the technology for creating desktop user interfaces).
- **ASP.NET:** This is the engine that hosts web applications and web services, with almost any feature from the .NET class library. ASP.NET also includes a set of web-specific services.
- **Visual Studio:** This optional development tool contains a rich set of productivity and debugging features. The Visual Studio setup CDs (or DVD) include the complete .NET Framework, so you won't need to download it separately.

- **ASP.NET File Types**
- ASP.NET have many types of files. They are,

1) **.aspx :** These are **ASP.NET web pages** (the .NET equivalent of the .asp file in an ASP application). 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.
2) **.ascx :** These are ASP.NET **user controls**. User controls are similar to web pages, except that they can't be accessed 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.
3) **.asmx :** These are ASP.NET **web services**. Web services work differently than web pages, but they still share the same application resources, configuration settings, and memory.
4) **web.config :** This is the **XML-based configuration file** for your ASP.NET application. It includes settings for customizing security, state management, memory management, and much more.
5) **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).
6) **.cs / .vb :** These are **code-behind files** that contain **C# code or VB code**. They allow you to separate the application from the user interface of a web page. **The Page Class :** Every web page is a custom class that inherits from System.Web.UI.Page. By inheriting from this class, your web page class acquires a number of properties that your code can use. These include properties for enabling caching, validation, and tracing.

- **Application Web Servers :**

- When you develop Web projects in Visual Studio, you need a Web server to test or run them. Visual Studio lets you test with different Web servers, including IIS Express, **Internet Information Services (IIS), or the built-in Visual Studio Development Server.** You can use any of these servers with a file-based Web application project. For a file-based Web site project, you can use IIS Express or the built-in Visual Studio Development Server.
- The following table provides summary guidance for choosing a Web server in Visual Web Developer.

Web server	When to use
IIS Express	Use when the target Web server is IIS 7 but you do not want to (or cannot) use the full version of IIS 7. This requires Visual Studio 2010 Service Pack 1, and IIS Express must be installed separately. IIS Express hosts sites in a manner that is very similar to IIS 7.
Visual Studio Development Server	Use when you are working with an existing project or your site targets an older version of IIS, such as IIS 6, and it is not very important that your testing environment match the production environment closely. This server option is the default in Visual Studio. However, the Visual Studio Development Server runs in a different security context than full IIS, and may fail to reveal errors that can occur when you deploy to a production version of IIS.
IIS	Use when you want to test your Web application using the server environment that is closest to what the live site will run under, and it is practical for you to install and work with IIS on your development computer. However, it can be more complex to configure debugging and other tasks than if you use IIS Express or the Visual Studio Development Environment. Requires you to run Visual Studio as an administrator.

- **Installation of IIS:**

➢ **IIS Express offers the following features:**

- It supports and enables the same extensibility model and **Web.config file settings as IIS 7.**
- It does not require changes in your Web application code.
- It can be installed side-by-side with the full IIS web server as well as with the Visual Studio Development Server. You can choose a different Web server for each project.

➢ **In corporate environments, IIS Express offers the following features:**

- It does not require an administrator account in order to run or debug applications.
- It does not serve requests to a browser on another computer, making its approval easier in corporate environments.
- It supports multiple developers on the same computer. Configuration files, settings, and Web content are maintained on a per-user basis under the **%systemdrive%\users\<username> folder.**
- It can be installed on versions of Windows that do not support IIS 7.

➢ **IIS Express Requirements**

To install IIS Express in your computer, you must have the following:

- Windows XP, Vista, or Windows 7, or Windows Server 2008 or 2008 R2.
- The .NET Framework 4 or later.
- Visual Studio 2010 SP1 or Visual Web Developer 2010 Express SP1.

- **Installing IIS Express**

  IIS Express is not installed automatically as part of Visual Studio 2010 SP1. To install IIS Express, you can use the Microsoft Web Platform Installer.
- **To install IIS Express in Visual Studio 2010 SP1**

1. In a browser, go to the installation page of the Microsoft.com/Web site.
2. Download the installer and then follow the steps to finish the installation.

- **Asp.net Controls:**

  The ASP.NET Framework (version 3.5) contains over 70 controls. These controls can be divided into seven groups:

- **Standard Controls :** The standard controls enable you to render standard form elements such as **buttons, input fields, and labels**.

- **Validation Controls :** The validation controls enable you to validate form data before you submit the data to the server. For example, you can use a **RequiredFieldValidator** control to check whether a user entered a value for a required input field.

- **Rich Controls :** The rich controls enable you to render things such as **calendars, file upload buttons, rotating banner advertisements**, and multi-step wizards.

- **Data Controls :** The data controls enable you to work with data such as **database data.** For example, you can use these controls to submit new records to a database table or display a list of database records. Navigation Controls. The navigation controls enable you to display standard navigation elements such as menus, tree views, and bread crumb trails.

- **Login Controls :** The login controls enable you to display login, change password, and registration forms.

- **Web Part Controls :** The Web Part controls enable you to build personalizable portal applications.

- **HTML Controls :** The HTML controls enable you to convert any HTML tag into a server-side control.

- With the exception of the HTML controls, you declare and use all the ASP.NET controls in a page in exactly the same way. For example, if you want to display a text input field in a page, then you can declare a TextBox control like this:

  <asp:TextBox id="TextBox1" runat="Server" />

- This control declaration looks like the declaration for an HTML tag. Remember, however, unlike an HTML tag, a control is a .NET class that executes on the server and not in the web browser.

- When the TextBox control is rendered to the browser, it renders the following content:

  <input name="TextBox1" type="text" id="TextBox1" />

- The first part of the control declaration, the asp: prefix, indicates the namespace for the control. All the standard ASP.NET controls are contained in the **System.Web.UI.WebControls** namespace. The prefix asp: represents this namespace.Next, the declaration contains the name of the control being declared. In this case, a TextBox control is being declared.

- This declaration also includes an ID attribute. You use the ID to refer to the control in the page within your code. Every control must have a unique ID.

- **Adding controls to webform:**

➢ **Label Control**

- Whenever you need to **modify the text displayed in a page dynamically,** you can use the Label control. Any string that you assign to the Label control's Text property is displayed by the Label when the control is rendered. You can assign simple text to the Text property or you can assign HTML content.
- As an alternative to assigning text to the Text property, you can place the text between the Label control's opening and closing tags. Any text that you place before the opening and closing tags gets assigned to the Text property.

- The Label control supports several properties you can use to format the text displayed by the Label (this is not a complete list):
  BackColor : Enables you to change the background color of the label.
  BorderColor : Enables you to set the color of a border rendered around the label.
  BorderStyle : Enables you to display a border around the label. Possible values are NotSet, None, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, and Outset.
  BorderWidth : Enables you to set the size of a border rendered around the label.
  CssClass : Enables you to associate a Cascading Style Sheet class with label.
  Font : Enables you to set the label's font properties.
  ForeColor : Enables you to set the color of the content rendered by the label.
  Style : Enables you to assign style attributes to the label.
  ToolTip : Enables you to set a label's title attribute. (In Microsoft Internet Explorer, the title attribute is displayed as a floating tooltip.)

➢ **TextBox Control:**

- The TextBox control can be used to **display three different types of input fields depending on the value of its TextMode property**. The TextMode property accepts the following three values:
  SingleLine : Displays a single-line input field.
  MultiLine : Displays a multi-line input field.
  Password : Displays a single-line input field in which the text is hidden.

- You can use the following properties to control the rendering characteristics of the TextBox control (this is not a complete list):
  AccessKey : Enables you to specify a key that navigates to the TextBox control.
  AutoCompleteType : Enables you to associate an AutoComplete class with the TextBox control.
  AutoPostBack : Enables you to post the form containing the TextBox back to the server automatically when the contents of the TextBox is changed.
  Columns : Enables you to specify the number of columns to display.
  Enabled : Enables you to disable the text box.
  MaxLength Enables you to specify the maximum length of data that a user can enter in a text box (does not work when TextMode is set to Multiline).
  ReadOnly : Enables you to prevent users from changing the text in a text box.
  Rows : Enables you to specify the number of rows to display.
  TabIndex : Enables you to specify the tab order of the text box.
  Wrap : Enables you to specify whether text word-wraps when the TextMode is set to Multiline.

- The TextBox control also supports the following method:
  Focus : Enables you to set the initial form focus to the text box.

- The TextBox control supports the following event:
  TextChanged : Raised on the server when the contents of the text box are changed.

- Notice that the TextBox control also includes a property that enables you to associate the TextBox with a particular AutoComplete class. When **AutoComplete** is enabled, the user does not need to re-enter common informationsuch as a first name, last name, or phone numberin a form field. If the user has not disabled AutoComplete on his browser, then his browser prompts him to enter the same value that he entered previously for the form field (even if the user entered the value for a form field at a different website).

➢ Checkbox Control

- Checkbox control is used to accept **the choice from user**. It is used to display multiple choices from which user can select none of them or many or all of them. For example, if you want to accept Hobbies of user, you can use CheckBox control.

- The CheckBox control supports the following properties (this is not a complete list):

  AccessKey : Enables you to specify a key that navigates to the TextBox control.

  AutoPostBack : Enables you to post the form containing the CheckBox back to the server automatically when the CheckBox is checked or unchecked.

  Checked : Enables you to get or set whether the CheckBox is checked.

  Enabled : Enables you to disable the TextBox.

  TabIndex : Enables you to specify the tab order of the check box.

  Text : Enables you to provide a label for the check box.

  TextAlign : Enables you to align the label for the check box. Possible values are Left and Right.

- The CheckBox control also supports the following method:

  Focus : Enables you to set the initial form focus to the check box.

- The CheckBox control supports the following event:

  CheckedChanged : Raised on the server when the check box is checked or unchecked.

➢ Radio Button Control

- You always use the RadioButton control in a group. Only one radio button in a group of RadioButton controls can be checked at a time.

- The RadioButton control supports the following properties (this is not a complete list):

  AccessKey : Enables you to specify a key that navigates to the RadioButton control.

  AutoPostBack : Enables you to post the form containing the RadioButton back to the server automatically when the radio button is checked or unchecked.

Checked : Enables you to get or set whether the RadioButton control is checked.
Enabled : Enables you to disable the RadioButton.
GroupName : Enables you to group RadioButton controls.
TabIndex : Enables you to specify the tab order of the RadioButton control.
Text : Enables you to label the RadioButton control.
TextAlign : Enables you to align the RadioButton label. Possible values are Left and Right.

- The RadioButton control supports the following method:
  Focus : Enables you to set the initial form focus to the RadionButton control.

- Finally, the RadioButton control supports the following event:
  CheckedChanged : Raised on the server when the RadioButton is checked or unchecked.

➢ **Button Control**

- Button control is used to submit the data to the server. Button control works like a Push Button when you click the data is submitted to the server.

- The Button control supports the following properties (this is not a complete list):
  AccessKey : Enables you to specify a key that navigates to the Button control.
  CommandArgument : Enables you to specify a command argument that is passed to the Command event.
  CommandName : Enables you to specify a command name that is passed to the Command event.
  Enabled : Enables you to disable the Button control.
  OnClientClick : Enables you to specify a client-side script that executes when the button is clicked.
  PostBackUrl : Enables you to post a form to a particular page.
  TabIndex : Enables you to specify the tab order of the Button control.
  Text : Enables you to label the Button control.
  UseSubmitBehavior : Enables you to use JavaScript to post a form.

- The Button control also supports the following method:
  Focus : Enables you to set the initial form focus to the Button control.

- The Button control also supports the following two events:
  Click : Raised when the Button control is clicked.
  Command : Raised when the Button control is clicked. The CommandName and CommandArgument are passed to this event.

➢ **Link Button Control**

- The LinkButton control, like the Button control, enables you to post a form to the server. Unlike a Button control, however, the LinkButton control renders a link instead of a push button.

- Behind the scenes, the LinkButton control uses JavaScript to post the form back to the server. The hyperlink rendered by the LinkButton control looks like this:

  Submit

- Clicking the LinkButton invokes the JavaScript __doPostBack() method, which posts the form to the server. When the form is posted, the values of all the other form fields in the page are also posted to the server.

- The LinkButton control supports the following properties (this is not a complete list):

  AccessKey : Enables you to specify a key that navigates to the Button control.

  CommandArgument : Enables you to specify a command argument that is passed to the Command event.

  CommandName : Enables you to specify a command name that is passed to the Command event.

  Enabled : Enables you to disable the LinkButton control.

  OnClientClick : Enables you to specify a client-side script that executes when the LinkButton is clicked.

  PostBackUrl : Enables you to post a form to a particular page.

  TabIndex : Enables you to specify the tab order of the LinkButton control.

  Text : Enables you to label the LinkButton control.

- The LinkButton control also supports the following method:

  Focus : Enables you to set the initial form focus to the LinkButton control.

- The LinkButton control also supports the following two events:

  Click : Raised when the LinkButton control is clicked.

  Command : Raised when the LinkButton control is clicked. The CommandName and CommandArgument are passed to this event.

➢ Image Button Control

- The ImageButton control, like the Button and LinkButton controls, enables you to post a form to the server. However, the ImageButton control always displays an image.

- The ImageButton includes both an ImageUrl and AlternateText property. The ImageUrl contains the path to the image that the ImageButton displays. The AlternateText property is used to provide alternate text for the image used by screen readers and text-only browsers.
- Notice that the event handler for an Image control's Click event is different than that for the other button controls. The second parameter passed to the event handler is an instance of the ImageClickEventArgs class. This class has the following properties:

  X : The x coordinate relative to the image the user clicked.

  Y : The y coordinate relative to the image the user clicked.

- You can use the ImageButton control to create a simple image map. The ImageButton can be used to create a server-side image map. Server-side image maps are not accessible to persons with disabilities. A better method for creating an ImageMap is to use the ImageMap control, which enables you to create a client-side image map.

- The ImageButton control supports the following properties (this is not a complete list):
  AccessKey : Enables you to specify a key that navigates to the ImageButton control.
  AlternateText : Enables you to provide alternate text for the image (required for accessibility).
  DescriptionUrl : Enables you to provide a link to a page that contains a detailed description of the image (required to make a complex image accessible).
  CommandArgument : Enables you to specify a command argument that is passed to the Command event.
  CommandName : Enables you to specify a command name that is passed to the Command event.
  Enabled : Enables you to disable the ImageButton control.
  GenerateEmptyAlternateText : Enables you to set the AlternateText property to an empty string.
  ImageAlign : Enables you to align the image relative to other HTML elements in the page. Possible values are AbsBottom, AbsMiddle, Baseline, Bottom, Left, Middle, NotSet, Right, TextTop, and Top.
  ImageUrl : Enables you to specify the URL to the image.
  OnClientClick : Enables you to specify a client-side script that executes when the ImageButton is clicked.
  PostBackUrl : Enables you to post a form to a particular page.
  TabIndex : Enables you to specify the tab order of the ImageButton control.

- The ImageButton control also supports the following method:
  Focus : Enables you to set the initial form focus to the ImageButton control.

- The ImageButton control also supports the following two events:
  Click : Raised when the ImageButton control is clicked.
  Command : Raised when the ImageButton control is clicked. The CommandName and CommandArgument are passed to this event.

➢ **Image Control**

- Image control is used to display an image. It has got some following properties.

  AlternateText Enables you to provide alternate text for the image (required for accessibility).
  DescriptionUrl Enables you to provide a link to a page that contains a detailed description of the image (required to make a complex image accessible).

GenerateEmptyAlternateText Enables you to set the AlternateText property to an empty string.

ImageAlign Enables you to align the image relative to other HTML elements in the page. Possible values are AbsBottom, AbsMiddle, Baseline, Bottom, Left, Middle, NotSet, Right, TextTop, and Top.

ImageUrl Enables you to specify the URL to the image.

- The Image control supports three methods for supplying alternate text. If an image represents page content, then you should supply a value for the AlternateText property. For example, if you have an image for your company's logo, then you should assign the text "My Company Logo" to the AlternateText property.

- If an Image control represents something really complexsuch as a bar chart, pie graph, or company organizational chartthen you should supply a value for the DescriptionUrl property. The DescriptionUrl property links to a page that contains a long textual description of the image.

- Finally, if the image is used purely for decoration (it expresses no content), then you should set the GenerateEmptyAlternateText property to the value TRue. When this property has the value TRue, then an alt="" attribute is included in the rendered <img> tag. Screen readers know to ignore images with empty alt attributes.

➢ Image Map Control

- The ImageMap control enables you to create a client-side image map. An image map displays an image. When you click different areas of the image, things happen.

- For example, you can use an image map as a fancy navigation bar. In that case, clicking different areas of the image map navigates to different pages in your website.

- You also can use an image map as an input mechanism. For example, you can click different product images to add a particular product to a shopping cart.

- An ImageMap control is composed out of instances of the HotSpot class. A HotSpot defines the clickable regions in an image map. The ASP.NET framework ships with three HotSpot classes:

  CircleHotSpot : Enables you to define a circular region in an image map.

  PolygonHotSpot : Enables you to define an irregularly shaped region in an image map.

  RectangleHotSpot : Enables you to define a rectangular region in an image map.

- Each RectangleHotSpot includes Left, Top, Right, and Bottom properties that describe the area of the rectangle. Each RectangleHotSpot also includes a

NavigateUrl property that contains the URL to which the region of the image map links.

The ImageMap control supports the following properties (this is not a complete list):

AccessKey Enables you to specify a key that navigates to the ImageMap control.

AlternateText Enables you to provide alternate text for the image (required for accessibility).

DescriptionUrl Enables you to provide a link to a page which contains a detailed description of the image (required to make a complex image accessible).

GenerateEmptyAlternateText Enables you to set the AlternateText property to an empty string.

HotSpotMode Enables you to specify the behavior of the image map when you click a region. Possible values are Inactive, Navigate, NotSet, and PostBack.

HotSpots Enables you to retrieve the collection of HotSpots contained in the ImageMap control.

ImageAlign Enables you to align the image map with other HTML elements in the page. Possible values are AbsBottom, AbsMiddle, Baseline, Bottom, Left, Middle, NotSet, Right, TextTop, and Top.

ImageUrl Enables you to specify the URL to the image.

TabIndex Enables you to specify the tab order of the ImageMap control.

Target Enables you to open a page in a new window.

- The ImageMap control also supports the following method:

  Focus Enables you to set the initial form focus to the ImageMap control.

- Finally, the ImageMap control supports the following event:

  Click Raised when you click a region of the ImageMap and the HotSpotMode property is set to the value PostBack.

➢ Panel Control

- The Panel control enables you to work with a group of ASP.NET controls. You can use a Panel control to hide or show a group of ASP.NET controls.

- The Panel control supports the following properties (this is not a complete list):

DefaultButton : Enables you to specify the default button in a Panel. The default button is invoked when you press the Enter button.

Direction : Enables you to get or set the direction in which controls that display text are rendered. Possible values are NotSet, LeftToRight, and RightToLeft.

GroupingText : Enables you to render the Panel control as a fieldset with a particular legend.

HorizontalAlign : Enables you to specify the horizontal alignment of the contents of the Panel. Possible values are Center, Justify, Left, NotSet, and Right.

ScrollBars : Enables you to display scrollbars around the panel's contents. Possible values are Auto, Both, Horizontal, None, and Vertical.

- **Hyperlink Control**

- The HyperLink control enables you to create a link to a page. Unlike the LinkButton control, the HyperLink control does not submit a form to a server.

- The HyperLink control supports the following properties (this is not a complete list):
  Enabled : Enables you to disable the hyperlink.
  ImageUrl : Enables you to specify an image for the hyperlink.
  NavigateUrl : Enables you to specify the URL represented by the hyperlink.
  Target : Enables you to open a new window.
  Text : Enables you to label the hyperlink.

- Notice that you can specify an image for the HyperLink control by setting the ImageUrl property. If you set both the Text and ImageUrl properties, then the ImageUrl property takes precedence.

Chapter-2: Validation And State Managment

➢ What is Validation?

- Validation is a process of **testing and ensuring** that the user has entered required and properly formatted information through the web form.

➢ Client Side Validation:

- When validation is done using a **script (usually in the form of JavaScript) in the page** that is posted to the end user's browser to perform validations on the data entered in the form before the form is posted back to the originating server. Then, client-side validation has occurred.
- Client-side validation is quick and responsive for the end user.
- Client-side validation is the more insecure form of validation. When a page is generated in an end user's browser, this end user can look at the code of the page quite easily (simply by right-clicking his mouse in the browser and selecting View Code).

➢ Server Side Validation:

- When validation occurs on **server**, where application resides it is called server side validation.
- The more secure form of validation is server-side validation.
- It is more secure because these checks cannot be easily bypassed.
- In server-side validation, all the input validations and error recovery process is carried out on the server side. It can be done using programming languages like **C#.NET, VB.NET etc.**

Validation Control:

➢ RequiredFieldValidator Control

- The RequiredFieldValidator control enables you to require a user to enter a value into a form field before submitting the form. You must set two important properties when using the RequiredFieldValdiator control:

  ControlToValidate : The ID of the form field being validated.

  Text : The error message displayed when validation fails.

By default, the RequiredFieldValidator checks for a nonempty string (spaces don't count). If you enter anything into the form field associated with the RequiredFieldValidator, then the RequiredFieldValidator does not display its validation error message. You can use the RequiredFieldValidator control's InitialValue property to specify a default value other than an empty string.

- **RangeValidator Control**

- The RangeValidator control enables you to check whether the value of a form field falls between a certain minimum and maximum value. You must set five properties when using this control:

  ControlToValidate : The ID of the form field being validated.
  Text : The error message displayed when validation fails.
  MinimumValue : The minimum value of the validation range.
  MaximumValue : The maximum value of the validation range.
  Type : The type of comparison to perform. Possible values are String, Integer, Double, Date, and Currency.

- If you don't enter any value into the age field and submit the form, no error message is displayed. If you want to require a user to enter a value, you must associate a RequiredFieldValidator with the form field.

- Don't forget to set the Type property when using the RangeValidator control. By default, the Type property has the value String, and the RangeValidator performs a string comparison to determine whether a values falls between the minimum and maximum value.

- **CompareValidator Control**

- The CompareValidator control enables you to perform three different types of validation tasks. You can use the CompareValidator to perform a data type check. In other words, you can use the control to determine whether a user has entered the proper type of value into a form field, such as a date in a birth date field.

- You also can use the CompareValidator to compare the value entered into a form field against a fixed value. For example, if you are building an auction website, you can use the CompareValidator to check whether a new minimum bid is greater than the previous minimum bid.

- Finally, you can use the CompareValidator to compare the value of one form field against another. For example, you use the CompareValidator to check whether the value entered into the meeting start date is less than the value entered into the meeting end date.

- The CompareValidator has six important properties:

ControlToValidate : The ID of the form field being validated.

Text : The error message displayed when validation fails.

Type : The type of value being compared. Possible values are String, Integer, Double, Date, and Currency.

Operator : The type of comparison to perform. Possible values are DataTypeCheck, Equal, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and NotEqual.

ValueToCompare : The fixed value against which to compare.

ControlToCompare : The ID of a control against which to compare.

- Just like the RangeValidator, the CompareValidator does not display an error if you don't enter a value into the form field being validated. If you want to require that a user enter a value, then you must associate a RequiredFieldValidator control with the field.

➢ RegularExpressionValidator Control

- The RegularExpressionValidator control enables you to compare the value of a form field against a regular expression. You can use a regular expression to represent string patterns such as email addresses, Social Security numbers, phone numbers, dates, currency amounts, and product codes.

- Just like the other validation controls, the RegularExpressionValidator doesn't validate a form field unless the form field contains a value. To make a form field required, you must associate a RequiredFieldValidator control with the form field.
- There are different expressions that can be given as Expression

Character	Meaning for Character in Expression
\w	Any word character (Letter, Number or Underscore)
\d	Any digit
\D	Any character that is not Digit
\s	Any white space character like Tab or Space
\S	Any non-white space character

- **Some expressions with examples.**

Expression	Meaning for Expression
\w {10}	10 word character (Letter, Number or Underscore)
\w {5,10}	Word character between 5 to 10
\d {5,10)	Any digits ranging from 5 to 10
[A-D]	Any 1 alphabet between A to D
[A-Z]{5,15}	Any alphabet between A to D but the no. of alphabets can be between 5 to 15.

\d{2}-\d{7,10}	Ex. 99-999999 (2 digits then hyphen and after that digits between 7 to 10

- **CustomValidator Control**

- If none of the other validation controls perform the type of validation that you need, you can always use the CustomValidator control. You can associate a custom validation function with the CustomValidator control.

- The CustomValidator control has three important properties:
  ControlToValidate : The ID of the form field being validated.
  Text : The error message displayed when validation fails.
  ClientValidationFunction : The name of a client-side function used to perform client-side validation.

- The CustomValidator also supports one event:
  ServerValidate : This event is raised when the CustomValidator performs validation.

- You associate your custom validation function with the CustomValidator control by handling the ServerValidate event.

- The second parameter passed to the ServerValidate event handler is an instance of the ServerValidateEventArgs class. This class has two properties:
  Value : Represents the value of the form field being validated.
  IsValid : Represents whether validation fails or succeeds.
  ValidateEmptyText : Represents whether validation is performed when the form field being validated does not contain a value.

- **ValidationSummary Control**

- The ValidationSummary control enables you to display a list of all the validation errors in a page in one location. This control is particularly useful when working with large forms. If a user enters the wrong value for a form field located toward the end of the page, then the user might never see the error message. If you use the ValidationSummary control, however, you can always display a list of errors at the top of the form.

- You might have noticed that each of the validation controls includes an ErrorMessage property. We have not been using the ErrorMessage property to represent the validation error message. Instead, we have used the Text property.

- The distinction between the ErrorMessage and Text property is that any message that you assign to the ErrorMessage property appears in the ValidationSummary control, and any message that you assign to the Text property appears in the body of the page. Normally, you want to keep the error message for the Text property short (for example, "Required!"). The message assigned to the ErrorMessage property, on

the other hand, should identify the form field that has the error (for example, "First name is required!").

- The ValidationSummary control supports the following properties:
  DisplayMode : Enables you to specify how the error messages are formatted. Possible values are BulletList, List, and SingleParagraph.
  HeaderText : Enables you to display header text above the validation summary.
  ShowMessageBox : Enables you to display a popup alert box.
  ShowSummary : Enables you to hide the validation summary in the page.

- If you set the ShowMessageBox property to the value true and the ShowSummary propertyto the value False, then you can display the validation summary only within a popup alert box.

Calendar Control:

- The Calendar control enables you to display a calendar. You can use the calendar as a date picker or you can use the calendar to display a list of upcoming events.

- The Calendar control supports the following properties (this is not a complete list):
  DayNameFormat : Enables you to specify the appearance of the days of the week. Possible values are FirstLetter, FirstTwoLetters, Full, Short, and Shortest.
  NextMonthText : Enables you to specify the text that appears for the next month link.
  NextPrevFormat : Enables you to specify the format of the next month and previous month link. Possible values are CustomText, FullMonth, and ShortMonth.
  PrevMonthText : Enables you to specify the text that appears for the previous month link.
  SelectedDate : Enables you to get or set the selected date.
  SelectedDates : Enables you to get or set a collection of selected dates.
  SelectionMode : Enables you to specify how dates are selected. Possible values are Day, DayWeek, DayWeekMonth, and None.
  SelectMonthText : Enables you to specify the text that appears for selecting a month.
  SelectWeekText : Enables you to specify the text that appears for selecting a week.
  ShowDayHeader : Enables you to hide or display the day names at the top of the Calendar control.
  ShowNextPrevMonth : Enables you to hide or display the links for the next and previous months.
  ShowTitle : Enables you to hide or display the title bar displayed at the top of the calendar.
  TitleFormat : Enables you to format the title bar. Possible values are Month and MonthYear.
  TodaysDate : Enables you to specify the current date. This property defaults to the current date on the server.

VisibleDate : Enables you to specify the month displayed by the Calendar control. This property defaults to displaying the month that contains the date specified by TodaysDate.

- The Calendar control also supports the following events:
  DayRender : Raised as each day is rendered.
  SelectionChanged : Raised when a new day, week, or month is selected.
  VisibleMonthChanged : Raised when the next or previous month link is clicked.

➢ **Ad Rotator Control :**

- The AdRotator control enables you to randomly display different advertisements in a page. You can store the list of advertisements in either an XML file or in a database table.

- The AdRotator control supports the following properties (this is not a complete list):
  AdvertisementFileEnables you to specify the path to an XML file that contains a list of banner advertisements.
  AlternateTextFieldEnables you to specify the name of the field for displaying alternate text for the banner advertisement image. The default value is AlternateText.
  DataMemberEnables you to bind to a particular data member in the data source.
  DataSourceEnables you to specify a data source programmatically for the list of banner advertisements.
  DataSourceIDEnables you to bind to a data source declaratively.
  ImageUrlFieldEnables you to specify the name of the field for the image URL for the banner advertisement. The default value for this field is ImageUrl.
  KeywordFilterEnables you to filter advertisements by a single keyword.
  NavigateUrlFieldEnables you to specify the name of the field for the advertisement link. The default value for this field is NavigateUrl.
  TargetEnables you to open a new window when a user clicks the banner advertisement.

- The AdRotator control also supports the following event:
  AdCreatedRaised after the AdRotator control selects an advertisement but before the AdRotator control renders the advertisement.

- Notice that the AdRotator control includes a KeywordFilter property. You can provide each banner advertisement with a keyword and then filter the advertisements displayed by the AdRotator control by using the value of the KeywordFilter property.

- This property can be used in multiple ways. For example, if you are displaying more than one advertisement in the same page, then you can filter the advertisements by page regions. You can use the KeywordFilter to show the big banner advertisement on the top of the page and box ads on the side of the page.

- You can also use the KeywordFilter property to filter advertisements by website section. For example, you might want to show different advertisements on your website's home page than on your website's search page.

- While AdRotator control the important property is Advertisement File which is XML file. Following is the example of Advertisement File, which has no. of fields which are related to your AdRotator property file.

File Name : Ad.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
        <ImageUrl>/Images/img1.gif</ImageUrl>
        <Width>500</Width>
        <Height>100</Height>
        <NavigateUrl>http://www.orkut.com</NavigateUrl>
        <AlternateText>Orkut Image</AlternateText>
        <Impressions>50</Impressions>
        <Keyword>Group</Keyword>
  </Ad>

  <Ad>
        <ImageUrl>/Images/img2.gif</ImageUrl>
        <Width>500</Width>
        <Height>100</Height>
        <NavigateUrl>http://www.gmail.com</NavigateUrl>
        <AlternateText>Gmail Image</AlternateText>
        <Impressions>50</Impressions>
        <Keyword>EMail</Keyword>
  </Ad>
</Advertisements>
```

Explanations of tags in Ad.xml File

- **<ImageUrl> :** Tag to specify virtual path of image
- **<Width> and <Height> :** Tags to specify width and height of image to be displayed.
- **<NavigateUrl> :** Tag to specify the link when you click in image.
- **<AlternateText> :** Tag to specify text if image is not showed due to some reason.
- **<Impressions> :** Tag to specify importance of image to show. Higher impression images are show more no. of times.
- **<Keyword> :** Tag to specify the keyword. If you specify some keyword, only related images are loaded. If not specified, all images are loaded.

➢ Internet Explorer Control:

- All web controls begin by inheriting from the WebControl base class. This class defines the essential functionality for tasks such as data binding and includes some basic properties that you can use with any control.
- Some of the new properties of WebControl class are as follows :

Property	Description
AccessKey	**Specifies the keyboard shortcut as one letter. For example, if you set this to Y, the Alt+Y keyboard combination will automatically change focus to this web control. This feature is supported only on Internet Explorer 4.0 and higher.**
Controls	**Provides a collection of all the controls contained inside the current control. Each object is provided as a generic System.Web.UI.Control object, so you will need to cast the reference to access control-specific properties.**
EnableViewState	**Set this to false to disable the automatic state management for this control. In this case, the control will be reset to the properties and formatting specified in the control tag every time the page is posted back. 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.**
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. If the control is placed directly on the page (rather than inside another control), it will return a reference to the page object.**
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. Pressing Tab moves the user to the control with the next lowest TabIndex, provided it is enabled. This property is supported only in Internet Explorer 4.0 and higher.**
ToolTip	**Displays a text message when the user hovers the mouse above the control. Many older browsers don't support this property.**
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.**

State Management:

View State:

- The HTTP protocol, the fundamental protocol of the World Wide Web, is a stateless protocol. Each time you request a web page from a website, from the website's perspective, you are a completely new person.

- The ASP.NET Framework, however, manages to transcend this limitation of the HTTP protocol. For example, if you assign a value to a Label control's Text property, the Label control retains this value across multiple page requests.

- The standard ASP.NET controls retain the values of their properties across postbacks. For example, if you change the text displayed by a Label control, the Label control will continue to display the new text even if you repeatedly post the page containing the Label control back to the server.

- The ASP.NET Framework takes advantage of a hidden form field named __VIEWSTATE to preserve the state of control properties across postbacks. If you want your controls to preserve the values of their properties, then you need to add the values of your control properties to this hidden form field.

- You can use the ViewState property of the Control or Page class to add values to View State. The ViewState property exposes a dictionary of key and value pairs. For example, the following statement adds the string Hello World! to View State:

  ViewState("message") = "Hello World!"

- The ASP.NET Framework uses a trick called View State. If you open the page in your browser and select View Source, you'll notice that the page includes a hidden form field named __VIEWSTATE that looks like this:

  <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUKLTc2ODE1OTYxNw9kFgICBA9kFgIC

  Aw8PFgIeBFRleHQFATFkZGT3tMnThg9KZpGak55p367vfInj1w==" />

- This hidden form field contains the value of the Label control's Text property (and the values of any other control properties that are stored in View State). When the page is posted back to the server, the ASP.NET Framework rips apart this string and re-creates the values of all the properties stored in View State. In this way, the ASP.NET Framework preserves the state of control properties across postbacks to the web server.

- By default, View State is enabled for every control in the ASP.NET Framework. If you change the background color of a Calendar control, the new background color is remembered across postbacks. If you change the selected item in a DropDownList, the selected item is remembered across postbacks. The values of these properties are automatically stored in View State.

- View State is a good thing, but sometimes it can be too much of a good thing. The __VIEWSTATE hidden form field can become very large. Stuffing too much data into View State can slow down the rendering of a page because the contents of the hidden field must be pushed back and forth between the web server and web browser.

- You can determine how much View State each control contained in a page is consuming by enabling tracing for a page. The page should include a trace="true" attribute in its <%@ Page %> directive, which enables tracing.

- Every ASP.NET control includes a property named EnableViewState. If you set this property to the value False, then View State is disabled for the control. In that case, the values of the control properties are not remembered across postbacks to the server.

➢ Session State:

- The session state is used to maintain the session of each user throughout the application. Session allows information to be stored in one page and access in another page and support any type of object. In many websites we will see the functionality like once if we login into website they will show username in all the pages for that they will store username in session and they will access that session username in all the pages.

- Whenever user enters into website new session id will generate for that user. This session Id will delete when he leave from that application. If he enters again he will get new session Id.

- **Session State** contains information that is pertaining to a specific session (by a particular client/browser/machine) with the server. It's a way to track what the user is doing on the site.. **across multiple pages**...amid the statelessness of the Web. e.g. the contents of a particular user's shopping cart is session data. Cookies can be used for session state.

- Session variables are stored in a SessionStateItemCollection object that is exposed through the HttpContext.Session property. In an ASP.NET page, the current session variables are exposed through the Session property of the Page object.

- Session variables are created by referring to the session variable by name. You do not have to declare a session variable or explicitly add it to the collection. The following example shows how to create session variables in an ASP.NET page for the first and last name of a user, and set them to values retrieved from TextBox controls.

```
Session["FirstName"] = FirstNameTextBox.Text;
Session["LastName"] = LastNameTextBox.Text;
```

- ASP.NET session state supports several different storage options for session data. Each option is identified by a value in the SessionStateMode enumeration. The following list describes the available session state modes:
- **InProc mode**, which stores session state in memory on the Web server. This is the default.
- **StateServer mode**, which stores session state in a separate process called the ASP.NET state service. This ensures that session state is preserved if the Web application is restarted and also makes session state available to multiple Web servers in a Web farm.
- **SQLServer mode** stores session state in a SQL Server database. This ensures that session state is preserved if the Web application is restarted and also makes session state available to multiple Web servers in a Web farm.
- **Custom mode,** which enables you to specify a custom storage provider.
- **Off mode**, which disables session state.

➢ Application State:

- Application state is a data repository available to all classes in an ASP.NET application. Application state is stored in memory on the server and is faster than storing and retrieving information in a database. Unlike session state, which is specific to a single user session, application state applies to all users and sessions. Therefore, application state is a useful place to store small amounts of often-used data that does not change from one user to another.
- Application state is stored in an instance of the HttpApplicationState class. This class exposes a key-value dictionary of objects.
- You can use application state in two ways. You can add, access, or remove values from the Contents collection directly through code. The HttpApplicationState class can be accessed at any time during the life of an application. However, it is often useful to load application state data when the application starts. To do so, you can put code to load application state into the Application_Start method in the Global.asax file.
- When using application state, you must be aware of the following important considerations:
- **Resources** Because it is stored in memory, application state is very fast compared to saving data to disk or a database. However, storing large blocks of data in application state can fill up server memory, causing the server to page memory to

disk. As an alternative to using application state, you can use the ASP.NET cache mechanism for storing large amounts of application data. The ASP.NET cache also stores data in memory and is therefore very fast; however, ASP.NET actively manages the cache and will remove items when memory becomes scarce. For more information see ASP.NET Caching Overview.

- **Volatility** Because application state is stored in server memory, it is lost whenever the application is stopped or restarted. For example, if the Web.config file is changed, the application is restarted and all application state is lost unless application state values have been written to a non-volatile storage medium such as a database.

- **Scalability** Application state is not shared among multiple servers serving the same application, as in a Web farm, or among multiple worker processes serving the same application on the same server, as in a Web garden. Your application therefore cannot rely on application state containing the same data for application state across different servers or processes. If your application will run in multi-processor or multi-server environments, consider using a more scalable option, such as a database, for data that must preserve fidelity across the application.

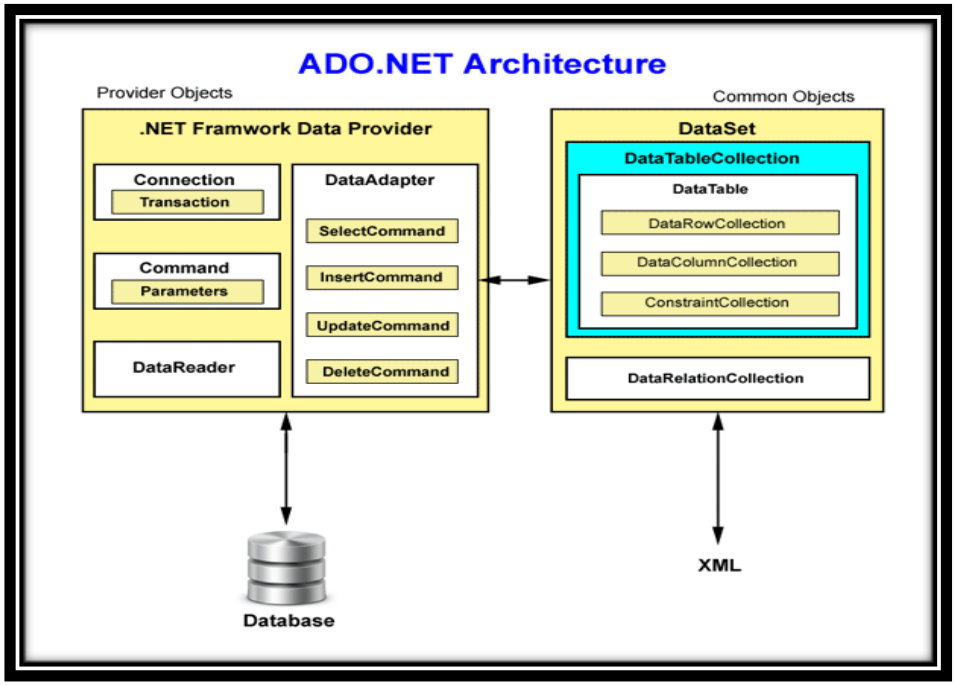
- **Concurrency** Application state is free-threaded, which means that application state data can be accessed simultaneously by many threads. Therefore, it is important to ensure that when you update application state data, you do so in a thread-safe manner by including built-in synchronization support. You can use the Lock and UnLock methods to ensure data integrity by locking the data for writing by only one source at a time. You can also reduce the likelihood of concurrency problems by initializing application state values in the Application_Start method in the Global.asax file.

Chapter-3: ADO.NET And Database

Architecture of ADO.NET:

What is ADO.NET?

- ADO stands for **ActiveX Data Objects**
- ADO.NET is a database technology of .NET Framework used to **connect application system and database server.**
- ADO.NET is a part of the .NET Framework
- ADO.NET consists of a set of classes used to handle data access
- ADO.NET uses XML to store and transfer data among applications, which is not only an industry standard but also provide fast access of data for desktop and distributed applications.
- ADO.NET is scalable and interoperable.



- It contains two types of architectures.
  1. Connected architecture.
  2. Disconnected architecture.

1. Connected Architecture:

- In **ADO.NET** Connected architecture connection to the database has to be in opened state to access the database.
- In Connected architecture connection string is used to connect to the database.
- Connection should be opened and closed.
- Commands are executed using methods like

Method	Commands	Return type
ExecuteNonQuery	Insert,Update,Delete,...	int
ExecuteReader	Select	SqlDataReader
ExecuteScalar	Aggregate commands	Object

- **ExecuteNonQuery** method is used to execute commands and return number of rows effected.
- **ExecuteReader** method is used to execute select command that retreives data and stores it in SqlDataReader which is capable of reading data row by row.
- **SqlDataReader** is read only and it reads in sequential order only. Read( ) method of SqlDataReader is used to read data row by row and it returns bool value.
- **ExecuteScalar** method is used to execute sql command that returns one value as its return type is object.
- If **ExecuteScalar** is used to execute select query that retrieves table data. It can return only first row first column cell value.

2. Dis-Connected Architecture:

- In **ADO.NET** Disconnected architecute data is retreived from database and stored in dataset.
- Data in dataset can be accessed even when the connection to the database is closed.
- SqlDataAdapter is used to open the connection ,execute the command, store data in dataset and close the connection.
- DataSet is capable of storing more than one table.
- DataSet is used to store data on client side.

da.fill(ds);

- The above line of code is used to execute the command by opening the connection to store data in dataset and closing the connection.

Create Connection using ADO.NET object Model:

- To do any operation on database, the first step is to **get the connection**. When creating a Connection object, you need to specify a value for its Connection-String property. This ConnectionString defines all the information the computer needs to find the data source, log in, and choose an initial database.

- As specified earlier, you can use different ways to perform your database operations. You can use **OleDb connection** method to connect database or can choose specific provider. If you have specific provider, its better to use that one instead of using OleDb, because specific provider speeds up rather then OleDb.

- Following is an example of creating connection using OleDb. You need to specify provider if you are using OleDb connection.

```
String x = "Provider=SQLOLEDB.1; Data Source=KARISHMA\HNS;
Initial Catalog=HNSDB;
        Integrated Security=SSPI";
OleDbConnection con = new OleDbConnection(x);
```

- Following is an example of creating connection using SqlConnection. You do not need to specify provider as it is specially used to connect sql server.

```
String x = "Data Source=KARISHMA\HNS; Initial Catalog=HNSDB; Integrated
                Security=SSPI";
SqlConnection con = new SqlConnection(x);
```

- Instead of specifying connection string each time, you can specify a global application variable in **Web.config file,** which you can retrieve in any page. Web.config file has a special tag under **<configuration> tag**. Inside <configuration> tag, you can specify <connectionStrings>, which can be used in all application to connect database. You can specify more then one connection string variables in this section. Following is a peace of code that can be specified under Web.config file.

```
<configuration xmlns=http://schemas.microsoft.com/.NetConfiguration/v2.0>
    <connectionStrings>
        <add name="a" connectionString="Data Source=KARISHMA\HNS;
                Initial Catalog=HNSDB; Integrated Security=SSPI" />
    </connectionStrings>
```

- Later, when you want to connect database in any form, you can use following code to use the "a" variable specified under Web.config file.

```
String x = WebConfigurationManager.ConnectionStrings[“a”].ConnectionString;
SqlConnection con = new SqlConnection(x);
```

Using Data Reader to Fetch Data

- Following are the steps to use Data Reader for retrieving record using Data Reader.

- Create and Open Connection
- Create Command object and specify Select command
- Create Data Reader object and assign ExecuteReader() method of command.
- Repeat through loop using Read() method of data reader.
- Close Data Reader and Connection.

Consider following code to retrieve data using Data Reader.

```
protected void Button1_Click(object sender, EventArgs e)
{
    String x = "Data Source=RAM\\SQLHNS; Initial
        Catalog=HNSDB; User Id=sa; Password=sasa";

    SqlConnection con = new SqlConnection(x);
    con.Open();
    SqlCommand cmd = new SqlCommand("select * from
        student", con);

    SqlDataReader dr;
    dr = cmd.ExecuteReader();

    while (dr.Read())
    {
       Response.Write(dr["studid"].ToString());
       Response.Write(dr["name"].ToString());
       Response.Write(dr["sex"].ToString());
       Response.Write(dr["age"].ToString());
       Response.Write(dr["city"].ToString());
       Response.Write("<hr>");
    }

    dr.Close();
    con.Close();
}
```

Close the Connection:

```
dr.close();
con.close();
```

Connection Class:

- For any type of database operation, connection is the first task to do. In ADO.NET, whether you are using connected or disconnected architecture, Connection class will be common for all.
- In Connection, We will be using SQL Server so we will discuss about SQL Server.

Property	Description
ConnectionString	This is connection string which is used to open SQL Server Connection. If you are using OleDb Connection then you need to specify one additional information called "Provider"
Connection TimeOut	This gets or sets the timeout period which has already last till now to establish connection.
Database	Gives the name of database server with which you are connected or going to connect
DataSource	Gives the name of database instance with which you are connected. In SQL Server, Database property will give you name of SQL Sever whereas DataSource property will give you name of database within SQL Server.
Server Version	Gives you version details of SQL Server with which you are connected.

Method	Description
Open	This method is used to open the connection. If you are using Connected Architecture, you must open the connection before doing any operation and if you are using disconnected architecture then you don't have to open the connection. Data adapter will automatically open the connection.
Close	This method is used to close the connection.

Begin Transaction	Allows you to start the transaction.
ChangeDatabase	Allows you to change the name of database. As SQL Server can have more than one databases so you can change database runtime by using this method.
CreateCommand	Allows you to create the object of SQL Command which is used in Connected Architecture.

```
String x = "Data Source=KARISHMA\HNS; Initial Catalog=HNSDB; Integrated
                Security=SSPI";
SqlConnection con = new SqlConnection(x);
```

Command Class:

- This class allows you to give the commands like **insert, update, delete, select etc**. If you want to do only transactional commands like Insert/ Update/ Delete/ Create Table/ Delete Table / etc. you will need only Command class along with Connection class.

Property	Description
Connection	This property specifies the connection object. Here, you need to specify the object of SqlConneciton class.
CommandText	Specifies the command text which can be INSERT, UPDATE, DELETE, SELECT or any other.
CommandType	Specifies the type of Command inside Command Object. It can be text or stored procedure.
Command Timeout	Allows you to get or set the time out period for command object. If it reaches timeout period, It generates the error.
Parameters	It is the important property of Command object which is the collection of parameters specified as Command Text.

Method	Description
ExecuteNonQuery	This method is used incase you have given transactional commands like

	INSERT, UPDATE, DELETE etc.
ExecuteReader	This method is used of you have specified SELECT.
ExecuteScalar	This method is used when you have specified SELECT command. But it returns value of First Row's First Column.
Parameters.Add()	Allow you to add parameter to the Parameters Collection. In Add() you need to specify name, data type and size of parameter.
Parameters.Clear()	Allows you to remove all parameters from Parameters Collection.q
Paramters.Remove()	Allows you to remove all parameters from Parameters Collection.
Parameters.RemoveAt()	Allows you to remove particular parameter from Parameters Collection by specifying its index value.

```
string x="Data Source=KARISHMA\\KARISHMASQL; Initial Catalog=KARISHMADB; User
Id=sa; Password=sasa";
SqlConnection con=new SqlConneciton(x);
Con.Open();
String sql="insert into student values(101,'karishma','F',25,'Rajkot')";
sqlCommand cmd=new SqlCommand(sql,con);
int ans=cmd.ExecuteNonQuery();
if(ans>0)
        Response.Write("Inserted Successfully");
else
        Response.Write("Problem With Inserting");
con.close();
```

Data Adapter Class:

Data Adapter class plays an important role in Disconnected Architecture. It is a mediator which provides or fills the data into DataSet / DataTable and again updates the data back to the database

Property	Description
SelectCommand	This property works in the background which generates your Select command automatically.
Delete Command	This property also works in background which generates your Delete command. If you have deleted some rows from DataSet or DataTable, SqlCommandBuilder automatically

	generates Delete Command and stores in this.
Insert Command	This property also works in background which generates Insert Command. If you have insert any new row in DataSet or DataTable, SqlCommandBuilder automatically generates Insert Command and stores in this.
Update Command	This property also works in background which generates Update Command. If you have done some changes in DataSet or DataTable, SqlCommandBuilder automatically generates IUpdate Command and stores in this

Method	**Description**
Fill	Most important method of DataAdapter which helps to fill the data into DataSet or DataTable. In DataAdapter we can give only SELECT command which is filled into DataSet or DataTable.
Update	This method is used to save the changes back from DataSet/DataTable to actual database table.

Event	**Description**
FillError	This event is raised when there is problem in filling data into DataSet or DataTable.
RowUpdating	This event is raised while changes are being made in Row. When you call Update() method, DataAdapter makes changes to the actual database table. At that time, While being updating each row this event is raised.
RowUpdated	This event is reaised afer changes are done in Row. When you call Update() method, DataAdapter makes changes to actual database table. After changes have been done, this event is raised.

Data Set Class:

This class is used if you want to load some group of tables into local memory. You can also use Dataset to load Single table data. Dataset is collection of tables.

Property	Description
DataSetName	Allows you to get or set the name of DataSet Object.
HasErrors	Gives the Boolean value to specify whether there are errors while filling or updating data.
IsInitialized	Gives the Boolean value to specify whether the data is stored in it or not.
Relations	Gives you the set of relations incase you have loaded more then one tables into dataset.
Tables	Collection of all tables which are loaded under DataSet object.
Tables.Count	Gives you how many tables are filled into dataset. This is because DataSet is collection of tables.

Method	Description
Clear	Clears the data inside DataSet.
AcceptChanges	Allows you to accept the changes. After calling the method changes are reflected back to database table.
WriteXml	Allow you to create XML file from data set data.
ReadXml	Allows you to read XML file and copy it into dataset.

Data Binding:

- Data Binding means you are creating a control with any table or table column or data of particular row.
- Data Binding is the process of retrieving data from source and dynamically associating it to a property of visual element.
- When you bind any control with some specific DataSet or DataTable, the control is known as Bound Control as it is bounded to particular column or table.
- For example, When you connect any table with GridView or DataGrid its called Data Binding.

- When you connect any column with Textbox or Label control, its called Data Binding.
- There are two types of Data Binding methods:

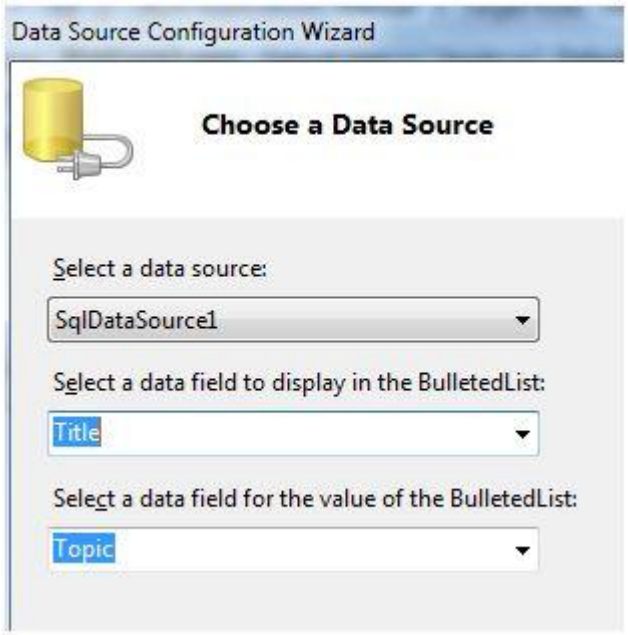
1) Simple Data Binding
2) Complex Data Binding

1) Simple data binding :

- It involves the read-only selection lists. These controls can bind to an array list or fields from a database. Selection lists takes two values from the database or the data source; one value is displayed by the list and the other is considered as the value corresponding to the display.
- Let us take up a small example to understand the concept. Create a web site with a bulleted list and a SqlDataSource control on it. Configure the data source control to retrieve two values from your database.

- Choosing a data source for the bulleted list control involves:

- Selecting the data source control
- Selecting a field to display, which is called the data field
- Selecting a field for the value



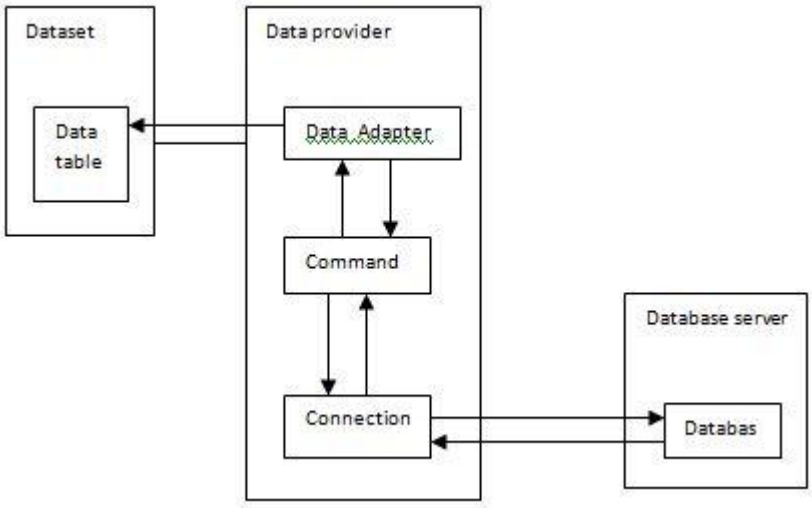
- When the application is run, check that the entire title column is bound to the bulleted list and displayed.

2) Complex Data Binding:

- We have already used declarative data binding in the previous tutorial using GridView control. The other composite data bound controls capable of displaying and manipulating data in a tabular manner are the DetailsView, FormView and RecordList control.

- However, the data binding involves the following objects:

- A dataset that stores the data retrieved from the database
- The data provider, which retrieves data from the database, using a command over a connection
- The data adapter that issues the select statement stored in the command object; it is also capable of update the data in a database by issuing Insert, Delete, and Update statements.
- Relation between the data binding objects:



Data Binding with web:

➢ The .NET framework provides a very flexible and powerful approach to databinding and allows the programmer to have a fine control over the steps involved in the whole process. One of the biggest improvements with .Net has been the introduction of databinding to web pages through the use of .Net server-side web controls. Hence, building data driven web applications has been greatly simplified. Please note that this article only deals with data binding in .NET windows forms.

Advantages of DataBinding:

1. Databinding in .NET can be used to write data driven applications quickly. .NET data binding allows you to write less code with fast execution but still get the work done in the best way.
2. .NET automatically writes a lot of databinding code for you in the background (you can see it in "Windows Generated Code" section), so the developer does not have to spend time writing code for basic databinding, but still has the flexibility of modifying any code that he would like to. We get the benefits of bound as well as unbound approach.
3. Control over the Databinding process by using events. This is discussed in more detail later in the article.

Disadvantages of DataBinding:

1. More optimized code can be written by using the unbound or traditional methods.
2. Complete flexibility can only be achieved by using the unbound approach.

Databinding Concepts

➢ For databinding to take place data provider and a data consumer should exist so that a synchronized link is established between the two. Data providers contain the data

and the data consumers use the data exposed by the data providers and display them.

- .NET has expanded the scope of possible data providers. In .NET any class or component that implements the **IList interface is a valid DataSource**. If a component implements the IList interface then it is transformed into an index based collection.

- Some of the classes that support the IList interface in the NET framework are given below. Please note that any class that implements the IList interface is a valid data provider.

1. Arrays
2. DataColumn
3. DataTable
4. DataView
5. DataSet

- PIList interface only allows you to bind at run time. If you want to support DataBinding at design time you will have to implement the IComponent interface as well. You cannot bind to DataReaders in windows forms (you can in web forms).

- The .NET framework supports simple and complex DataBinding. Simple databinding is supported by controls like TextBoxes. In Simple databinding, only one data value can be displayed by the control at a time. In complex databinding, which is supported by controls like the DataGrid, more than one data value from the DataSource can be displayed.

Web Server Controls:

- Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.
- The syntax for creating a Web server control is:

<asp:control_name id="some_id" runat="server" />

Web Server Control	Description
AdRotator	Displays a sequence of images
Button	Displays a push button
Calendar	Displays a calendar
CalendarDay	A day in a calendar control
CheckBox	Displays a check box
CheckBoxList	Creates a multi-selection check box group
DataGrid	Displays fields of a data source in a grid
DataList	Displays items from a data source by using templates
DropDownList	Creates a drop-down list
HyperLink	Creates a hyperlink

Image	Displays an image
ImageButton	Displays a clickable image
Label	Displays static content which is programmable (lets you apply styles to its content)
LinkButton	Creates a hyperlink button
ListBox	Creates a single- or multi-selection drop-down list
ListItem	Creates an item in a list
Literal	Displays static content which is programmable(does not let you apply styles to its content)
Panel	Provides a container for other controls
PlaceHolder	Reserves space for controls added by code
RadioButton	Creates a radio button
RadioButtonList	Creates a group of radio buttons
BulletedList	Creates a list in bullet format
Repeater	Displays a repeated list of items bound to the control
Style	Sets the style of controls
Table	Creates a table
TableCell	Creates a table cell
TableRow	Creates a table row
TextBox	Creates a text box
Xml	Displays an XML file or the results of an XSL transform

Display Data on webform using Data Bound Controls:

- Data-bound controls are WinForms controls those can easily bind with data components. Microsoft Visual Studio.NET is a rich IDE for ADO.NET data components. Im going to talk about these controls in a moment. In this article, Im going to talk about three main data-bound controls DataGrid, ListBox, and a ComboBox.

- Data-bound controls have properties, which you can set as a data component and theyre ready to present your data in WinForms. DataSource and DisplayMemeber are two important properties.

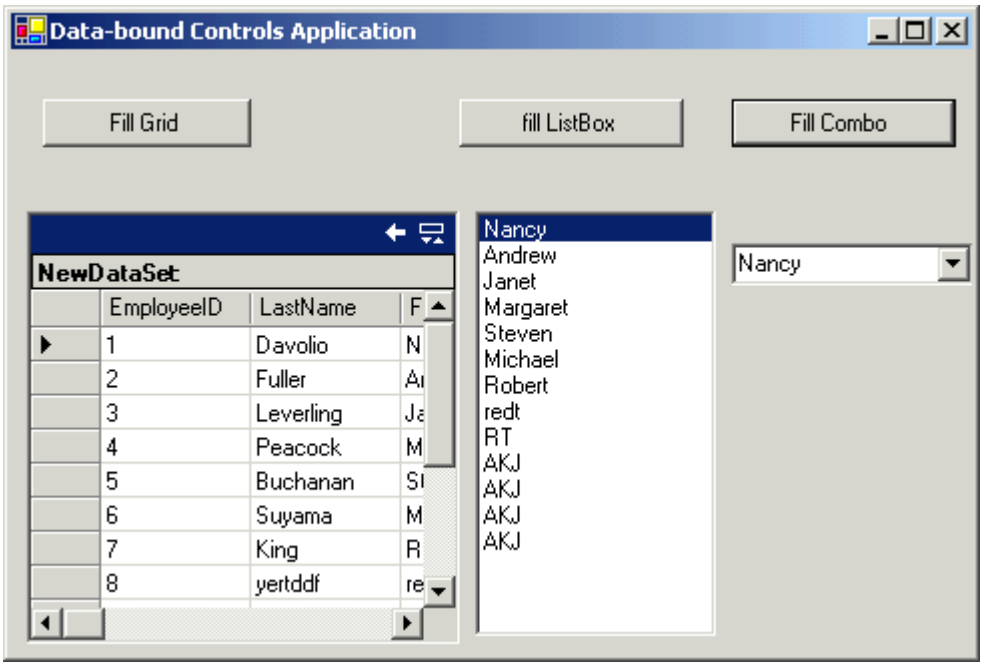
- **DataSource** property of these controls plays a major role. You can set different kind of data components as datasource property of a control. For example, you can set a DefaultViewManager or a DataView as this property.

```
DataSet ds = new DataSet();
dataGrid1.DataSource = ds.DefaultViewManager;
```

- **DisplayMember** property can be set to a database table field name if you want to bind a particular field to the control.

```
DataSet ds = new DataSet();
// Attach dataset's DefaultView to the datagrid control
DataView dv = ds.Tables["Employees"].DefaultView;
```

```
listBox1.DataSource = dv;
listBox1.DisplayMember = "FirstName";
```



Chapter-4: Using XML

Writing and Reading Datasets to XML:

1. Creating a new project.
2. Creating an XML file to be read into the dataset.
3. Creating the user interface.
4. Creating the dataset, read the XML file, and display it in a DataGridView control.
5. Adding code to display the XML Schema based on the XML file in a TextBox control.

1. Create a new project:

2. To create the XML file that will be read into the dataset:

1. From the Project menu, choose Add New Item.
2. Select XML File, name the file authors.xml, and click Add. The XML file loads into the designer and is ready for edit.
3. Paste the following code into the editor below the XML declaration:

3. Create the User Interface:

➢ The user interface for this application will consist of the following:

- A **DataGridView** control that will display the contents of the XML file as data.
- A **TextBox** control that will display the XML Schema for the XML file.
- Two **Button** controls.
  - One button reads the XML file into the dataset and displays it in the **DataGridView** control.
  - A second button extracts the schema from the dataset, and through a **StringWriter** displays it in the **TextBox** control.

4. Creating the dataset, read the XML file, and display it in a DataGridView control.

1. With the source file for Form1 selected in Solution Explorer, click the View Designer button in the Solution Explorer toolbar.
2. From the Toolbox, Data Tab, drag a DataSet onto Form1.
3. Select Untyped dataset on the Add Dataset dialog box, and then click OK.
   DataSet1 is added to the component tray.
4. In the Properties window, set the Name and DataSetName properties to AuthorsDataSet.

5. Adding code to display the XML Schema based on the XML file in a TextBox control.

- In Solution Explorer, select Form1 and click the View Designer button.
- Double-click the Show Schema button.
- The Code Editor opens at the ShowSchemaButton_Click event handler.
- Type the following code into the ShowSchemaButton_Click event handler.

Web Service:

- A web service is a **web-based functionality** accessed using the protocols of the web to be used by the web applications. There are three aspects of web service development:

- **Creating the web service**
- **Creating a proxy**
- **Consuming the web service**

Creating the Web Sevice:

- A web service is an web application which is basically a class consisting of methods that could be used by other applications. It also follows a code-behind architecture like the ASP.Net web pages, although it does not have an user interface.

- The web service will have three methods:

- A defaullt HelloWorld method
- A GetName Method
- A GetPrice Method

Step (1): Select File--> New --> Web Site in Visual Studio, and then select ASP.Net Web Service.

Step (2): A web service file called Service.asmx and its code behind file, Service.cs is created in the App_Code directory of the project.

Step (3): Change the names of the files to StockService.asmx and StockService.cs.

Step (4): The .asmx file has simply a WebService directive on it:

Step (5): Open the StockService.cs file, the code generated in it is the basic Hello World service

Step (6) : Change the code behind file to add the two dimensional array of strings for stock symbol, name and price and two web methods for getting the stock information.

Step (7) : Running the web service application gives a web service test page, which allows testing the service methods.

Step (8) : Click on a method name, and check whether it runs properly.

Step (9): For testing the GetName method, provide one of the stock symbols, which are hard coded, it returns the name of the stock

Consuming the Web Service:

- For using the web service, create a web site under the same solution. This could be done by right clicking on the Solution name in the Solution Explorer. The web page calling the web service should have a label control to display the returned results and two button controls one for post back and another for calling the service.

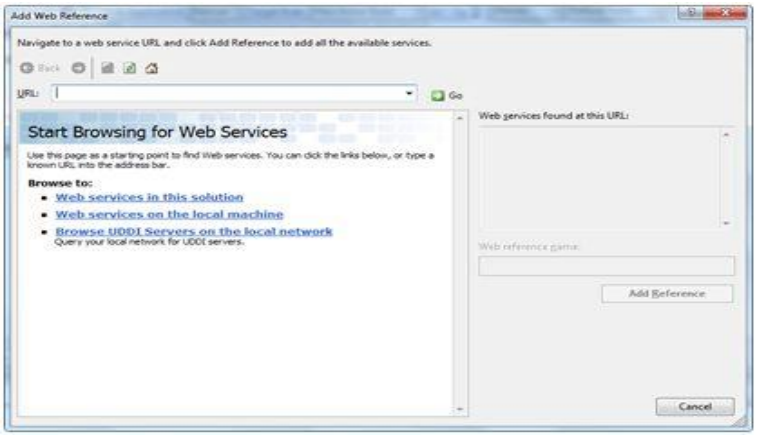
Creating the Proxy:

- A proxy is a stand-in for the web service codes. Before using the web service, a proxy must be created. The proxy is registered with the client application. Then the client application makes the calls to the web service as it were using a local method.

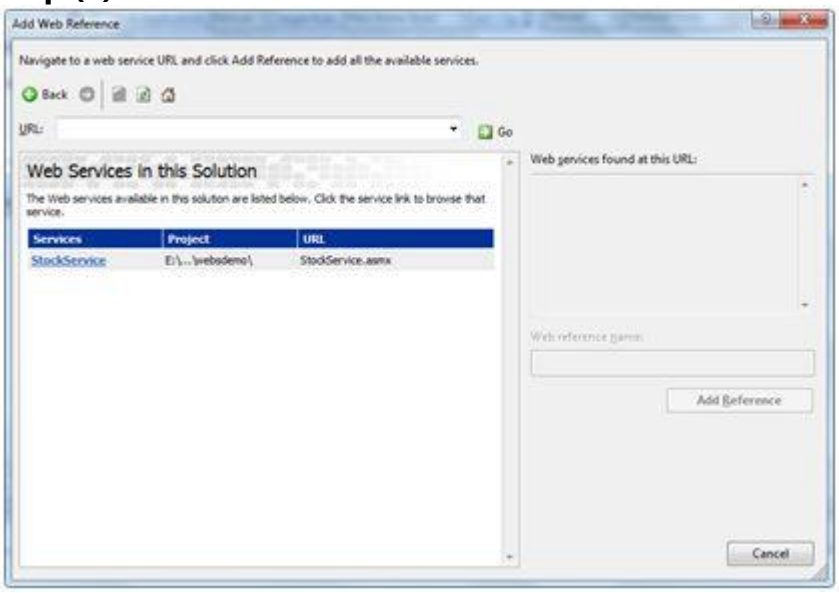
- The proxy takes the calls, wraps it in proper format and sends it as a SOAP request to the server. SOAP stands for Simple Object Access Protocol. This protocol is used for exchanging web service data.

- When the server returns the SOAP package to the client, the proxy decodes everything and presents it to the client application.

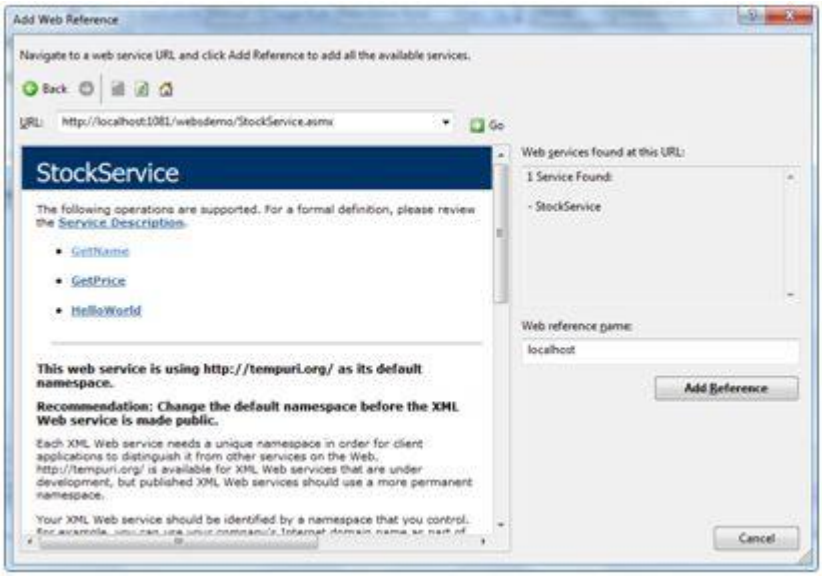
- Before calling the web service using the btnservice_Click, a web reference should be added to the application. This creates a proxy class transparently, which is used by the btnservice_Click event.
  Step (1): Right click on the web application entry in the Solution Explorer and click on 'Add Web Reference'.



Step (2): Select 'Web Services in this solution'. It returns the StockService reference.



Step (3): Clicking on the service opens the test web page. By default the proxy created is called 'localhost', you can rename it. Click on 'Add Reference' to add the proxy to the client application.



- Include the proxy in the code behind file by adding:

using localhost;

Remote Method Call using XML AND SOAP:

- XML-RPC defines a method call as an XML element called **"methodCall"**
- A "methodCall" element will have a required "methodName" element and an optional "params" element.
- A **"methodName**" element specifies the name of the method.
- A **"params"** element specifies a list of "param" elements as parameters for the method.
- A "param" element will have a "value" element.
- A **"value"** element will have a value type element, which contains the actual value of the parameter.
- There are 7 value type elements: "int", "boolean", "string", "double", "dateTime.iso8601", "base64", "struct" and "array". The first 6 are simple elements and the last 2 are complex elements.
- A "struct" element will have a list of "member" elements. A "member" element will have a "name" element and a "value" element.
- An "array" element will have a "data" elements, which will have a list of "value" elements.

- SOAP message is an XML format sent over HTTP to a remote server where the Web service is located.

- The Web service processes the soap request and returns the value to the client using soap response.

- SOAP message is an XML format sent over HTTP to a remote server where the Web service is located.

- The Web service processes the soap request and returns the value to the client using soap response. This soap message contains three basic elements

1. Envelope
2. Header (optional)
3. Body

- The entire soap message should be within the envelope element.

- Each element should have namespace for scoping. Member of the element can be accessed within that namespace.Namespace is required to avoid confusion incase of having same element names in different places within a message.

- In this soap message header element passes the user email id for authentication.

- Any fault messages can have four sub elements.

1. **faultcode(optional**) - contains the type of error occured.
2. **faultstring(optional)** - contains the cause of the error.
3. **faultactor** - specifies the server name where the error is occurred.
4. **details -** contains two sub elements.
   a. message - contains the detail error messge
   b. errorcode - contains the error code.

Web Service Description Language:

A WSDL definition is an XML document with a root definition element from the http://schemas.xmlsoap.org/wsdl/ namespace. The entire WSDL schema is available at http://schemas.xmlsoap.org/wsdl/ for your reference. The definitions element may contain several other elements including types, message, portType, binding, and service, all of which come from the http://schemas.xmlsoap.org/wsdl/ namespace.

```
<!-- WSDL definition structure -->
<definitions
name="MathService"
targetNamespace="http://example.org/math/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
>
<!-- abstract definitions -->
<types> ...
<message> ...
<portType> ...
```

<!-- concrete definitions -->
<binding> ...
<service> ...
</definition>

- The first three elements (**types**, **message**, and **portType**) are all abstract definitions of the Web service interface. These elements constitute the programmatic interface that you typically interface with in your code. The last two elements (**binding** and **service**) describe the concrete details of how the abstract interface maps to messages on the wire.

Element Name	Description
types	A container for abstract type definitions defined using XML Schema
message	A definition of an abstract message that may consist of multiple parts, each part may be of a different type
portType	An abstract set of operations supported by one or more endpoints (commonly known as an interface); operations are defined by an exchange of messages
binding	A concrete protocol and data format specification for a particular portType
service	A collection of related endpoints, where an endpoint is defined as a combination of a binding and an address (URI)

- The **WSDL** types element is a container for **XML Schema type definitions**. The type definitions you place here are referenced from higher-level message definitions in order to define the structural details of the message. Officially, WSDL 1.1 allows the use of any type definition language, although it strongly encourages the use of XML Schema and treats it as its intrinsic type system. The WS-I enforces this by mandating the use of XML Schema in the Basic Profile 1.0.

- The types element contains zero or more schema elements from the http://www.w3.org/2001/XMLSchema namespace. The basic structure of the types element (with namespaces omitted) is as follows (***** means zero or more):
<definitions .... >
<types>
<xsd:schema .... />*
</types>
</definitions>

Chapter-5: Web Application And Configuration

Asp.Net Web Configuration Overview:

- The ASP.NET configuration API comprises a set of **ASP.NET management** objects that you can use to configure Web sites and applications programmatically. Management objects are implemented as a .NET Framework class library. The configuration API programming model helps ensure code consistency and reliability by enforcing data types at compile time.
- The **configuration API** allows you to view data that is merged from all points in the configuration hierarchy as a single collection, instead of viewing the data as separate collections from different configuration files. Additionally, the configuration API enables you to manipulate entire application configurations without directly editing the XML in the configuration files. Finally, the API simplifies configuration tasks by supporting administrative tools, such as the Web Site Administration Tool.

- The **configuration API** simplifies deployment by supporting the creation of configuration files on a computer and running configuration scripts across multiple computers.

- The configuration API does not support the creation of IIS applications.

- ASP.NET website configuration is normally a combination of two files:
- ***machine.config***
- ***web.config***
- Every time we install the .NET framework, there is a *machine.config* file that is created in "*C:\WINDOWS\Microsoft.NET\Framework\[Version]\CONFIG*", which mainly defines:
- Supported configuration file sections,
- the ASP.NET worker process configuration, and
- registers different providers that are used for advanced features such as profiles, membership, and role based security.
- To explore the *web.config* might take a book, but here, I'll try to explore all the important sections that play a pivotal role for an ASP.NET website and its deployment.

- Every web application inherits settings from the *machine.config* file, and application level setting is done in the *web.config* file. We can also override configurations in the *machine.config* file in the *web.config* file. But, a few settings can not be overridden because certain settings are process model settings and can't be changed on a per application basis.

- The entire contents of a configuration file, whether it is *machine.config* or *web.config*, is nested in a <configuration> element.

- ASP.NET uses a multilayered configuration system that allows for using different settings for different parts of an application. For this, we must have an additional subdirectory inside the virtual directory, and these subdirectories will contain their own config files with additional settings. ASP.NET uses **configuration inheritance** so that each subdirectory acquires the settings from the parent directory.

- <configuration> section consists of many other sub elements which are as follows:

- **<configSections>** : This allows you to configure different sections of web application like JavaScript, Web Services, Resources etc.

- **<appSettings>** : This allows you to add any application related variable or settings. You can <add>, <remove> or <clear> any application specific variable or settings.

- **<connectionStrings> :** This allows you to create connectionString variables which can be used in any Web Form of your web application.

- **<system.web> :** This is important element used to configure your web application. This section allows you to do most of configuration as follows:

  <compilation> **<authentication>**
  <caching> **<customErrors>**
  <deployment> **<roleManager>**
  <trace> **<sessionState>**
  <webServices>

- **<system.webServer> :** This section stores information and configuration settings for web servers which will be used for your deploying and running your web application.

- **<runtime> :** This section is used to specify some run time settings for your web application.

Common Configuration Settings:

- Some Common Configuration settings are:

- **Tracing Web Application:**

  Tracing allows you to check for any type of bugs that your web application may have. Along with tracing your web application, you can also maintain log files which give you some additional information. These log files can help you to find out some problems that occur during web application usage.

- **Customizing Errors**:
  While using web site, sometimes different error occur. You are not aware that which error is going to come. Web Site users feel uncomfortable when they get some unexpected screen. By customizing errors, you can redirect your error to specific page which can display error but in your design specific way.

- **Authentication and Authorization**:
  Authenticated users are requirement for most of the web sites. To allow authenticated users, you must authorize users in such a way that any even any unauthorized user visits your web site, should not be able to harm your information. This can be done with the help of combining Authentication and Authorization.

- **Enabling Role Manager:**
  Enabling Role Manager is a feature which is used along with Authentication and Authorization. This feature can be used to allow some specific users. For example, some of the web page of your website are important, you want only "Administrator" level user can access. "Guest" users should not be able to access these pages.

- **Session Configuration**:
  Session Configuration allows you to specify session settings like Timeout Period, Session Mode, Cookie Based Session or CookielessSession, SqlCommandTimeout, NetworkTimeout, etc.

- **Trust Levels**:

  There Can be different types of web site Users. For example, Web master will have access to all the web pages of web site. Web Developer will have some limited access to web pages of Website; Administrator will have access to all the back end pages for web site administration where at last level is Users who will have access to only those pages which are informative. So different types of trust levels can be defined using trust levels.

- **Web Service Configuration**:
  Web services are configured to lock some resources from being accessed by some specific users.
- **Caching:**
  Caching in short means, keeping your output or data into cache memory so that it can be accessed in a faster way instead of gathering output from web Server or searching data from Database.

Tracing:

- ASP.NET tracing enables you to view **diagnostic information** about a single request for an ASP.NET page. ASP.NET tracing enables you to follow a page's execution path, display diagnostic information at run time, and debug your application. ASP.NET tracing can be integrated with system-level tracing to provide multiple levels of tracing output in distributed and multi-tier applications.
- ASP.NET tracing offers the following features:

- **Debugging statments** You can write debug statements in your code without having to remove them from your application when it is deployed to production servers. You can also write variables or structures in a page and trace through the execution path of your page or application.

- **Integrated tracing functionality** You can route messages emitted by the System.Diagnostics.Trace class to ASP.NET tracing output, and route messages emitted by ASP.NET tracing to System.Diagnostics.Trace. You can also forward ASP.NET instrumentation events to System.Diagnostics.Trace. For more information, see Walkthrough: Integrating ASP.NET Tracing with System.Diagnostics Tracing.

- **Programmatic access to trace messages** You can access and manipulate trace messages from your code for finer control over the format of trace messages or for additional processing that you require.

- **Application-level tracing** The application-level tracing option lets you view the most recent tracing data available without restarting a tracing session and without increasing the amount of tracing data that the server must store. The most recent tracing data is displayed, and older tracing data is discarded.

- There are two ways:

(i)Page Level Tracing
(ii) Application Level Tracing

Page Level Tracing:

- We can control whether tracing is enabled or disabled for individual pages. If tracing is enabled, when the page is requested, ASP.NET appends to the page a series of tables containing execution details about the page request. Tracing is disabled by default in an ASP.NET application.

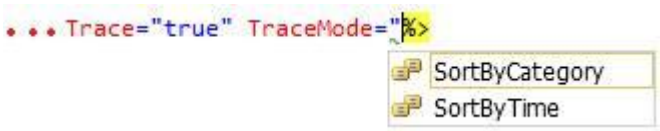
- To enable Page Level Tracing follow the steps:

i) Include Trace="true" in <%@ Page Title="" Language="C#"...%> directive, for example:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEv
entWireup="true" CodeFile="Default.aspx.cs"Inherits="chat_Default" Trace="true"%
>
```

Look at the above code, I'll be using Trace=true at the end.

ii) Optionally, we can use TraceMode attribute in above <%@ Page Title="" Language="C#"...%> directive, for example:



SortByCategory: Set TraceMode to SortByTime to sort trace messages in the order in which they are processed.

SortByTime: Set TraceMode to SortByCategory to sort trace messages by the categories.

iii) **Now press F5 to run the application, you will see the immediate trace record on an existing page that you have set Trace and TraceMode**.

You can see the traced record in the Trace Viewer as well; you will learn this in 'Application Level Tracing'.

Application Level Tracing

- Instead of enabling tracing for individual pages, you can enable it for your entire application. In that case, every page in your application displays trace information. Application tracing is useful when you are developing an application because you can easily enable it and disable it without editing individual pages. When your application is complete, you can turn off tracing for all pages at once.

- When you enable tracing for an application, ASP.NET collects trace information for each request to the application, up to the maximum number of requests you specify. The default number of requests is 10. You can view trace information with the trace viewer.

- By default, when the trace viewer reaches its request limit, the application stops storing trace requests. However, you can configure application-level tracing to always store the most recent tracing data, discarding the oldest data when the maximum number of requests is reached.

- To enable Application Level Tracing follow the steps:

  i) Delete your Page Level Tracking for better result.

  ii) Open the Web.config file and add the following information to it; if there is not a Web.config file available then add a new one in the root.

```
.....
<system.web>
  <trace enabled="true" pageOutput="true" requestLimit="40" localOnly="false"/>
</system.web>
</configuration>
```

  iii) The above code has many attributes used, find the detailed information about them below.

  Enabled: Set it true to enable tracing for the application; otherwise, false. The default is false. You can override this setting for individual pages by setting the Trace attribute in the @ Page directive of a page to true or false.

  PageOutput: Set it true to display trace both in pages and in the trace viewer (trace.axd); otherwise, false. The default is false.

  RequestLimit: The number of trace requests to store on the server. The default is 10.

  LocalOnly: Set it true to make the trace viewer (trace.axd) available only on the host Web server; otherwise, false. The default is true.

Custom Error:

- <customErrors> is used mostly to deal with different types of standard errors.
- Web.Config file gives you <customErrors> element, by configuring which you can define your own "Custom Error Message" for ASP.NET Web Application. This helps you by

- **Showing your own customized error message** : User can understand it as it can be written in easy language.

- **Showing all the error messages in a common format:** users don't feel that they are thrown out of the scope due to some error.

- In <customErrors> section we can specify a redirect page as default error page or specify to a particular page based on the HTTP error code that is raised. We can use

this method to customize the error message that the user receives. If an error occurs that is not trapped at any of the previous levels in the application, this custom page is displayed.

- The <customErrors> section in Web.config has two attributes that affect what error page is shown:

  DefaultRedirect: The defaultRedirect attribute is optional .If provided, it specifies the URL of the custom error page and indicates that the custom error page should be shown instead of Runtime Error.

  Mode: The mode attribute is required and accepts one of 3 values. On, Off, or RemoteOnly. These values have the following behaviou:

  On: This option specifies that custom errors are enabled. If no defaultRedirect is specified, users see a usual error.

  Off : This option specifies that custom errors are disabled. This allows the display of detailed errors.

  RemoteOnly: This option specifies that custom errors are shown only to remote clients and ASP.NET errors are shown to the local host. This is the default.

- Customisation of error page can be implemented by adding a value for an attribute defaultRedirect in the **<customErrors>** tag of the configuration file *web.config*. This file determines configuration settings for the underlying application.

Authentication And Authorization:

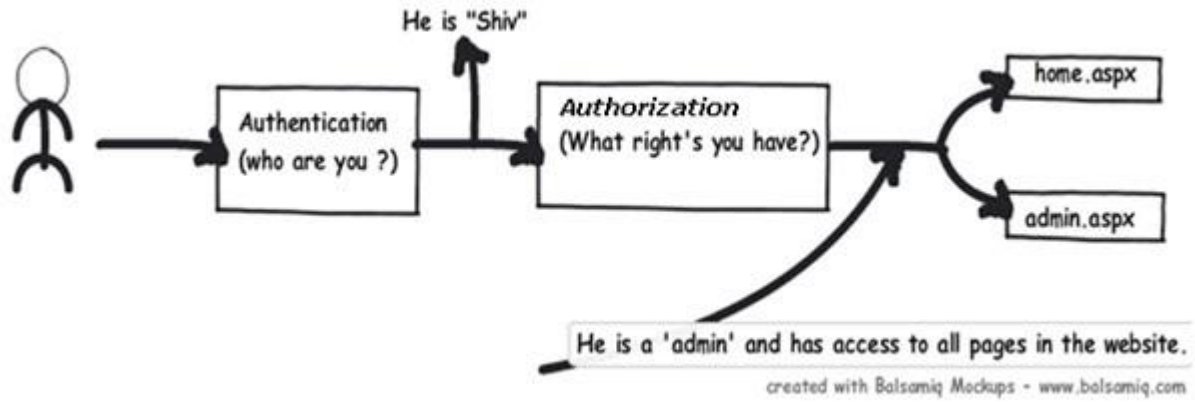
- ***Authentication*** is knowing the identity of the user. For example, Alice logs in with her username and password, and the server uses the password to authenticate Alice.

- ***Authorization*** is deciding whether a user is allowed to perform an action. For example, Alice has permission to get a resource but not create a resource.

- The same dictionary meaning applies to ASP.NET as well. In ASP.NET authentication means to identify the user or in other words its nothing but to validate that he exists in your database and he is the proper user. **Authorization m**eans does he have access to a particular resource on the IIS website. A resource can be an ASP.NET web page, media files (MP4, GIF, JPEG etc), compressed file (ZIP, RAR) etc.

- So the first process which happens is authentication and then authorization. Below is a simple graphical representation of authentication and authorization. So when the user enters 'userid' and 'password' he is first authenticated and identified by the user name.

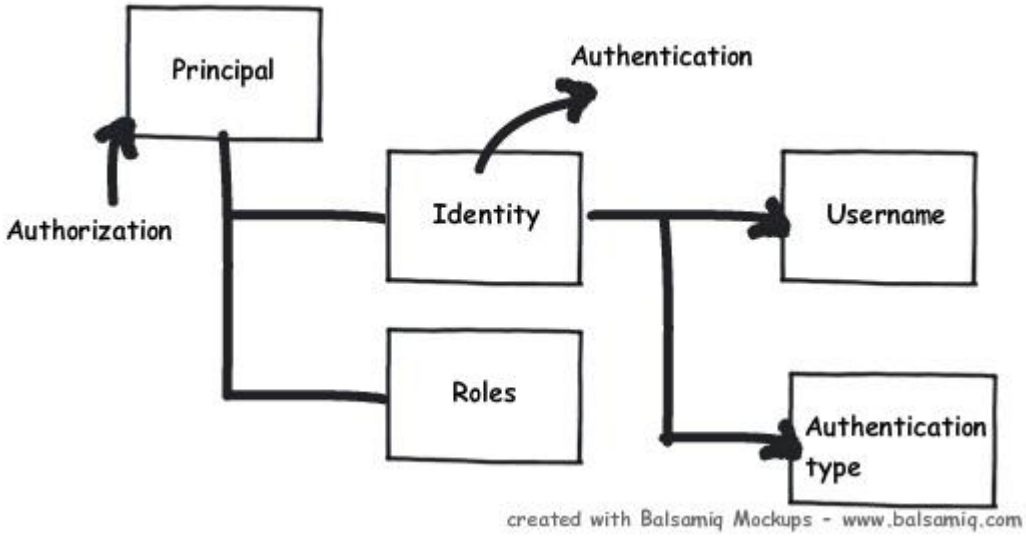
Now when the user starts accessing resources like pages, **ASPDOTNETauthentication**, videos etc, he is checked whether he has the necessary access for the resources. The process of identifying the rights for resources is termed as 'Authorization'.



- To put it in simple words to identify "he is shiv" is authentication and to identify that "Shiv is admin" is authorization.

- **Detecting authentication and authorization: - The principal and identity objects**

➢ At any moment of time if you want to know who the user is and what kind of authentication type he using you can use the identity object. If you want to know what kind of roles it's associated with then we need to use the principal object. In other words to get authentication details we need to the identity object and to know about authorization details of that identity we need the principal object.



- **Types of authentication and authorization in ASP.NET:**

There are three ways of doing authentication and authorization in ASP.NET:-
• **Windows authentication: -** In this methodology ASP.NET web pages will use local windows users and groups to authenticate and authorize resources.

• **Forms Authentication: -** This is a cookie based authentication where username and password are stored on client machines as cookie files or they are sent through URL for every request. Form-based authentication presents the user with an HTML-based Web page that prompts the user for credentials.

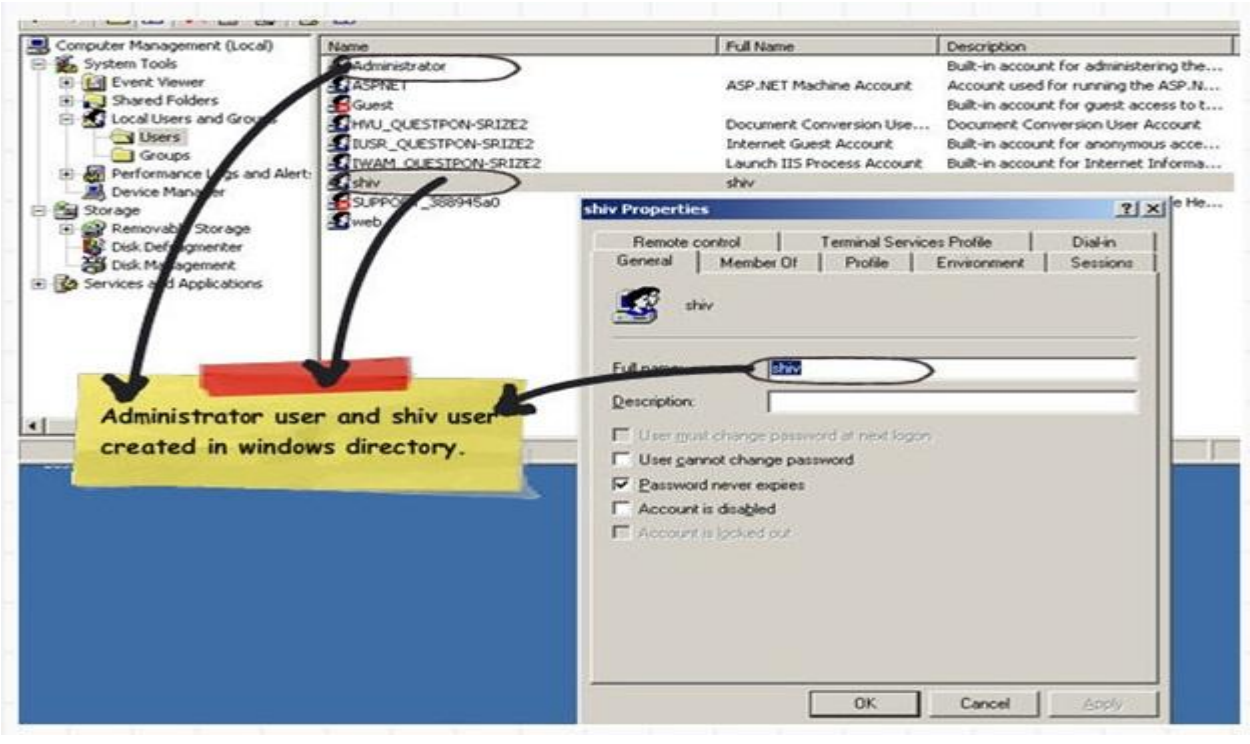
• **Passport authentication :-** Passport authentication is based on the passport website provided
by the Microsoft .So when user logins with credentials it will be reached to the passport website (i.e. hotmail,devhood,windows live etc) where authentication will happen. If Authentication is successful it will return a token to your website.

• **Anonymous access: -** If you do not want any kind of authentication then you will go for Anonymous access.

GenericPrincipal and GenericIdentity objects represent users who have been authenticated using Forms authentication or other custom authentication mechanisms. With these objects, the role list is obtained in a custom manner, typically from a database.
FormsIdentity and PassportIdentity objects represent users who have been authenticated with Forms and Passport authentication respectively.

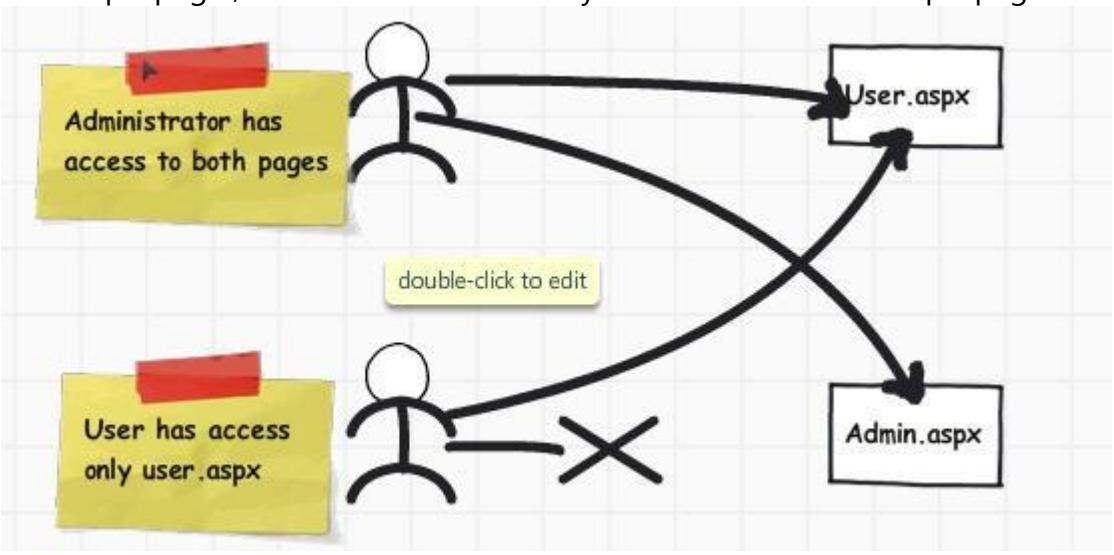
Windows Authentication:

When you configure your ASP.NET application as windows authentication it will use local windows user and groups to do authentication and authorization for your ASP.NET pages. Below is a simple snap shot which shows my windows users and roles on my computer.



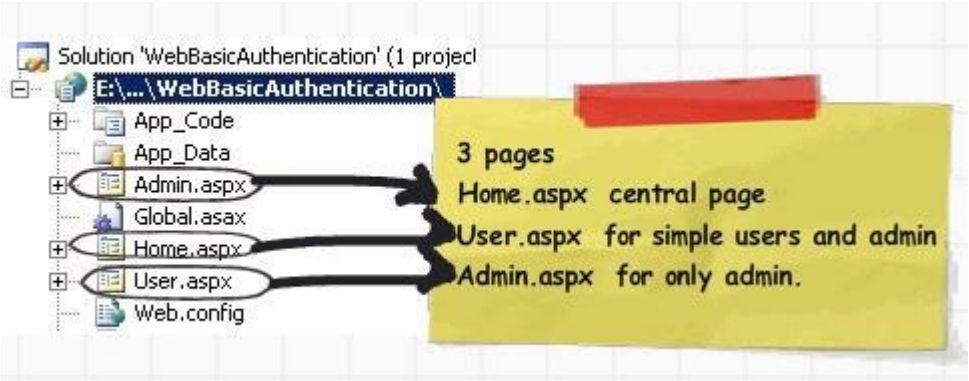
5 steps to enable authentication and authorization using Windows

We will do a small sample to get a grasp of how authentication and authorization works with windows. We will create 2 users one 'Administrator' and other a simple user with name 'Shiv'. We will create two simple ASPX pages 'User.aspx' page and 'Admin.aspx' page. 'Administrator' user will have access to both 'Admin.aspx' and 'User.aspx' page , while user 'Shiv' will only have access to 'User.aspx' page.



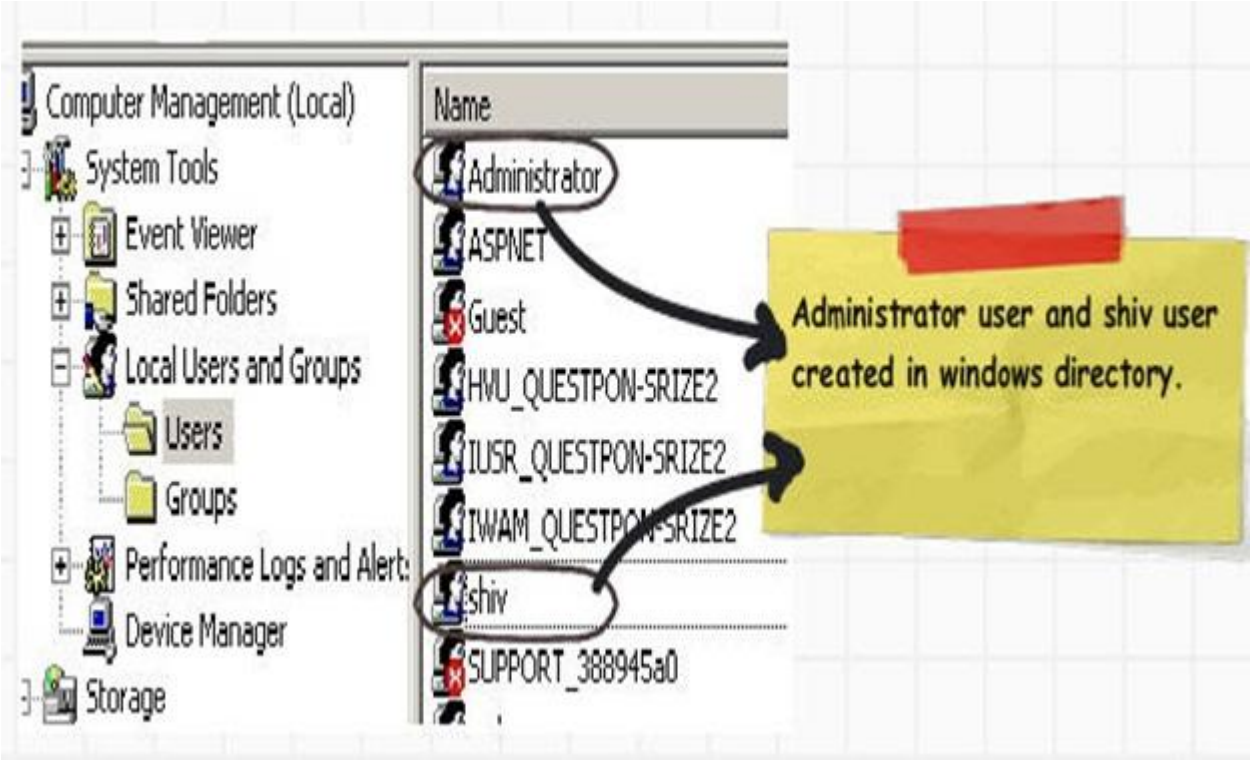
Step 1:- Creation of web site.

The next step is to create a simple web site with 3 pages (User.aspx, Admin.aspx and Home.aspx) as shown below.



Step 2:- Create user in the windows directory

The next step is we go to the windows directory and create two users. You can see in my computer we have 'Administrator' and 'Shiv'.



Step 3:- Setup the 'web.config' file

In 'web.config' file set the authentication mode to 'Windows' as shown in the below code snippets.

```
<authentication mode="Windows"/>
```

We also need to ensure that all users are denied except authorized users. The below code snippet inside the authorization tag that all users are denied. '?' indicates any unknown user.

```
<authorization>
<deny users="?"/>
</authorization>
```

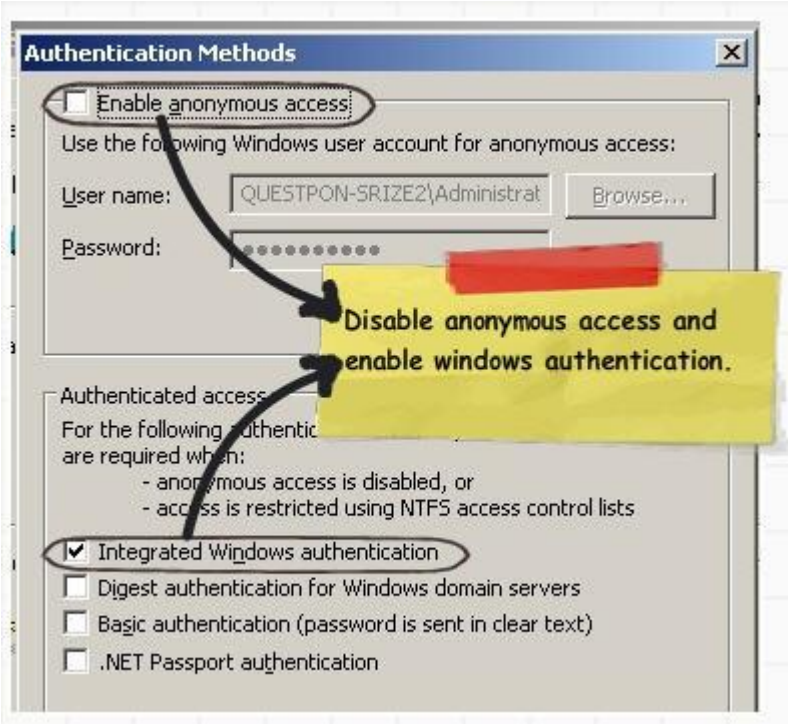
Step 4:- Setup authorization

We also need to specify the authorization part. We need to insert the below snippet in the 'web.config' file stating that only 'Administrator' users will have access to 'Admin.aspx' pages.

```
<location path="Admin.aspx">
<system.web>
<authorization>
<allow roles="questpon-srize2\Administrator"/>
<deny users="*"/>
</authorization>
</system.web>
</location>
```

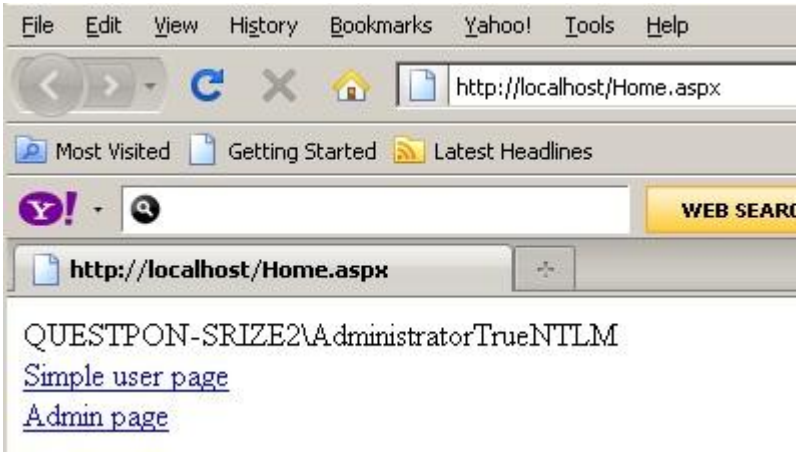
Step 5:-Configure IIS settings

The next step is to compile the project and upload the same on an IIS virtual directory. On the IIS virtual directory we need to ensure to remove anonymous access and check the integrated windows authentication as shown in the below figure.

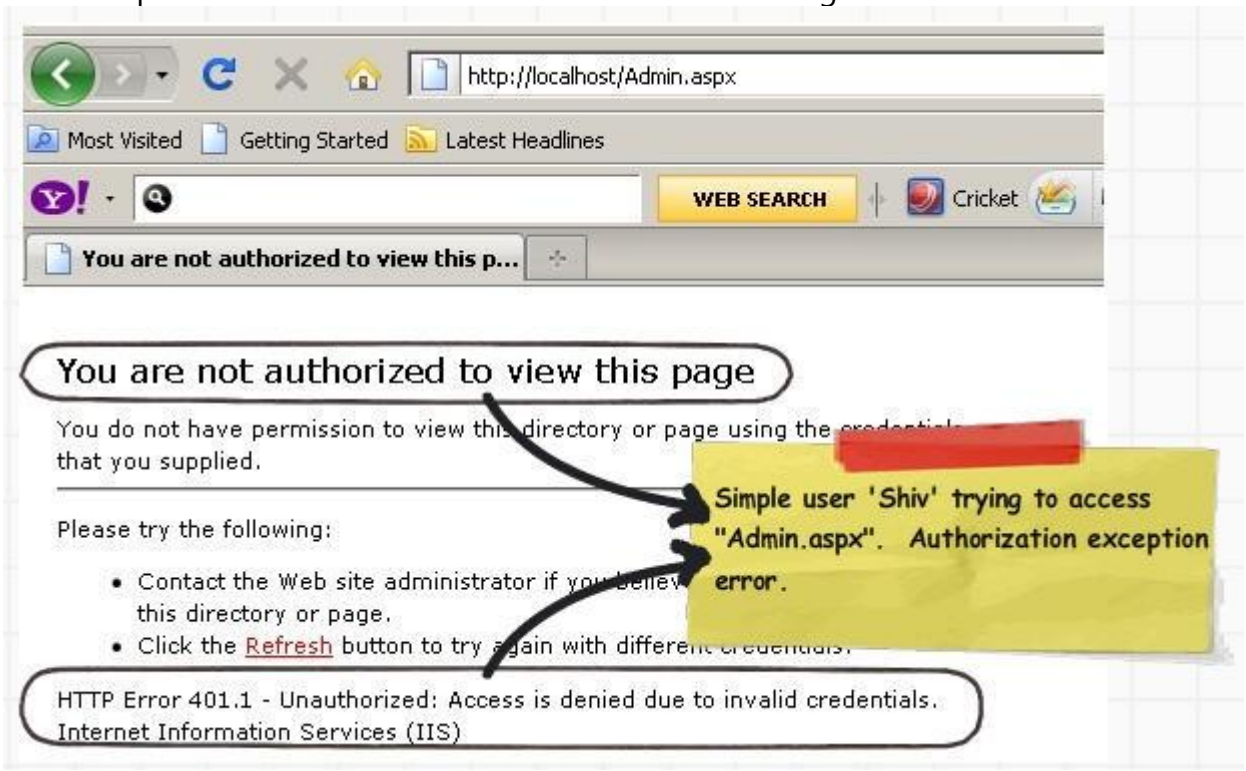


Now if you run the web application you will be popped with a userid and password box as shown below.

Once you enter credentials you should be able to see home.aspx as shown below.



In case you are not an administrator (i.e in this case its 'shiv') and you navigate to 'Admin.aspx' it will throw an error as shown in the below figure.



In case you want to read who the user is and with what authorization rights has he logged in you can use 'WindowsPrincipal' and 'WindowsIdentity'. These two objects represent users who have been authenticated with Windows authentication. You can also get the roles these users have.

Web Services:

- A Web Service is programmable application logic accessible via standard Web protocols. One of these Web protocols is the Simple Object Access Protocol (SOAP). SOAP is a W3C submitted note (as of May 2000) that uses standards based technologies (XML for data description and HTTP for transport) to encode and transmit application data.

- Consumers of a Web Service do not need to know anything about the platform, object model, or programming language used to implement the service; they only need to understand how to send and receive SOAP messages (HTTP and XML).

Soap Message:

- A SOAP message consists of several elements, most notably an envelope. The envelope encapsulates the data transmitted within the SOAP message. Below is a simple SOAP message complete with HTTP headers:

- Web Services are simple and easy to understand. It is possible, in fact, to author a simple application that surfaces data as XML conforming to the SOAP specification. It would also be relatively straightforward to build an application capable of receiving SOAP messages over HTTP and deriving meaningful value out of it. For those of you familiar with PERL, this could simply be a matter of using RegEx to parse the value out of the XML result; it's just another string.

- However, just as we use frameworks such as ASP and ASP.NET to build Web applications, we would much rather use a framework for building Web Services. The reasoning is quite logical. We don't need to reinvent the plumbing—that is, at a high level, the capability to serialize our data as XML, transport the data using HTTP, and de-serialize the XML back to meaningful data. Instead, we want a framework that makes building Web Services easy, allowing us to focus on the application logic not the plumbing. ASP.NET provides this framework for us.

- From a developer's point of view, if you have ever written application logic, you have the required skills to author ASP.NET Web Services. More importantly, if you're at all familiar with ASP or ASP.NET application services, (application state memory, and so on) you can also leverage these skills when you build ASP.NET Web Services.