ASP.NET - Introduction

ASP.NET

What is ASP.Net?

- ASP.Net is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web application for PC, as well as mobile devices.
- ASP.Net works on top of the HTTP protocol and uses the HTTP commands and policies to set a browser-to-server two-way communication and cooperation.
- ASP.Net is a part of Microsoft .Net platform.
- ASP.Net applications are complied codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

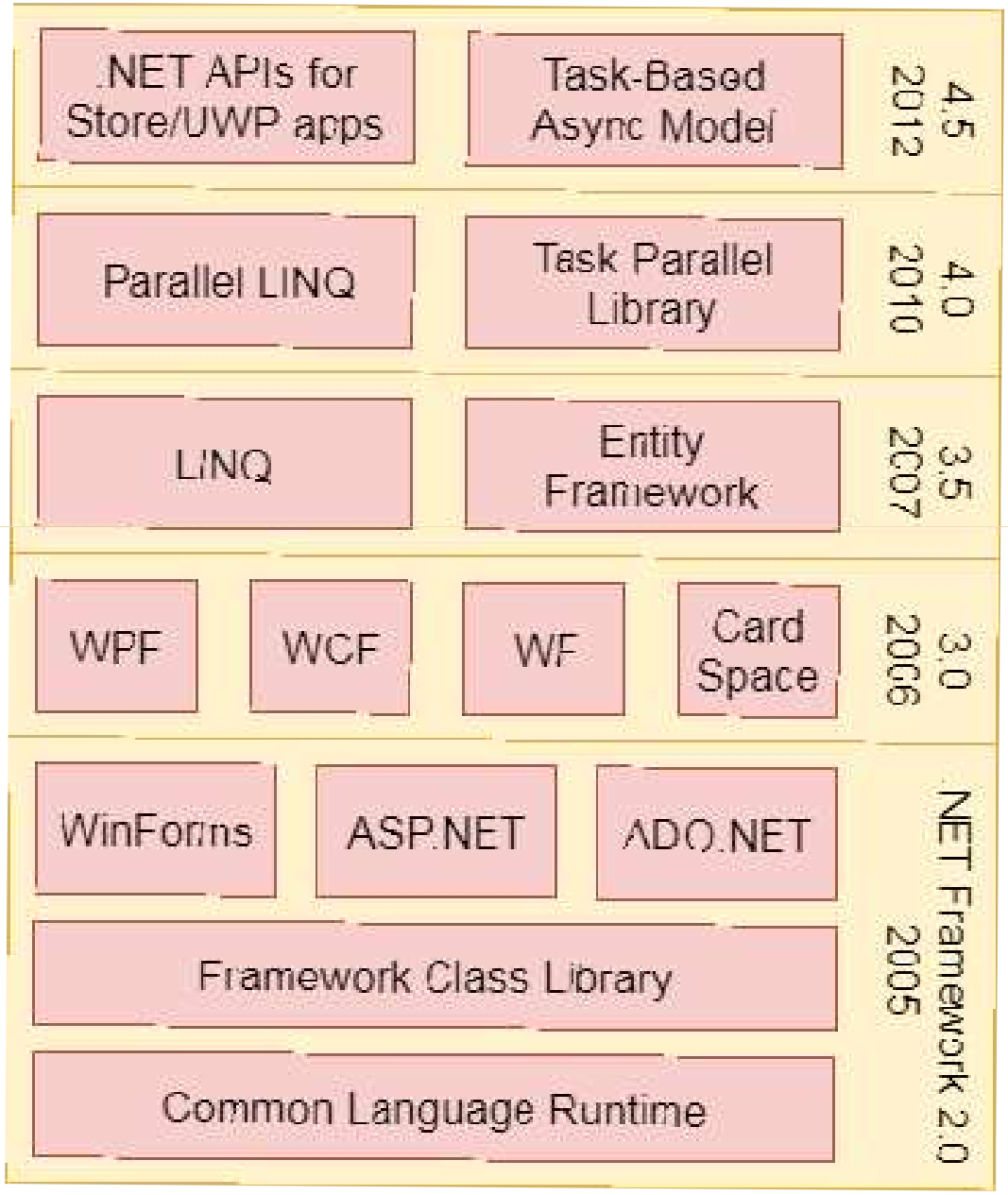
ASP.NET

- The ASP.Net application codes could be written in either of the following languages:

1. C#
2. Visual Basic .Net
3. Jscript
4. J#

- ASP.Net is used to produce interactive, data-driven web applications over the internet.
- It consists of a large number of controls like text boxes, buttons and labels for assembling, configuring and manipulating code to create HTML pages.

Fig: .NET framework components



ASP.NET

- ASP.NET is a web framework designed and developed by Microsoft. It is used to develop websites, web applications and web services.
- It provides fantastic integration of HTML, CSS and JavaScript. It was first released in January 2002.
- It is built on the Common Language Runtime (CLR) and allows programmers to write code using any supported .NET language.
- ASP.NET provides three development styles for creating web applications:
  - Web Forms
  - ASP.NET MVC
  - ASP.NET Web Pages
-

Web Forms

- It is an event driven development framework used to develop application with powerful data access.
- It provides server side controls and events to create web application. It is part of the ASP.NET framework.

ASP.NET MVC

- It gives us a MVC (Model View Controller), patterns-based way to build dynamic websites.
- It enables a clean separation of concerns and that gives you full control over markup for enjoyable, agile development.
- It also provides many features that enable fast development for creating outstanding applications.

ASP.NET Web Pages

- It is used to create dynamic web pages providing fast and lightweight way to combine server code with HTML along with video, link to the social sites and conforms to the latest web standards.
- We can use any development style to create application. The selection of style is depends on the skills and experience of the programmer.
- Although each framework is independent to other, we can combine and use any of that at any level of our application.
- For example, to develop client interaction module, we can use MVC and for data control, we can use Web Forms.

Page Lifecycle stages

Stage	Description
Page request	This stage occurs before the lifecycle begins. When a page is requested by the user, ASP.NET parses and compiles that page.
Start	In this stage, page properties such as Request and response are set. It also determines the Request type.
Initialization	In this stage, each control's UniqueID property is set. Master page is applied to the page.
Load	During this phase, if page request is postback, control properties are loaded with information.
Postback event handling	In this stage, event handler is called if page request is postback. After that, the Validate method of all validator controls is called.
Rendering	Before rendering, view state is saved for the page and all controls. During the rendering stage, the page calls the Render method for each control, providing a text writer that writes its output to the OutputStream object of the page's Response property.
Unload	At this stage the requested page has been fully rendered and is ready to terminate.at this stage all properties are unloaded and cleanup is performed.

ASP.NET Lifecycle Events

Page Event	Typical Use
PreInit	This event is raised after the start stage is complete and before the initialization stage.
Init	This event occurs after all controls have been initialized. We can use this event to read or initialize control properties.
InitComplete	This event occurs at the end of the page's initialization stage. We can use this event to make changes to view state that we want to make sure are persisted after the next postback.
PreLoad	This event is occurs before the post back data is loaded in the controls.
Load	This event is raised for the page first time and then recursively for all child controls.
Control events	This event is used to handle specific control events such as Button control' Click event.
LoadComplete	This event occurs at the end of the event-handling stage. We can use this event for tasks that require all other controls on the page be loaded.
PreRender	This event occurs after the page object has created all controls that are required in order to render the page.
PreRenderComplete	This event occurs after each data bound control whose DataSourceID property is set calls its DataBind method.
SaveStateComplete	It is raised after view state and control state have been saved for the page and for all controls.
Render	This is not an event; instead, at this stage of processing, the Page object calls this method on each control.
Unload	This event raised for each control and then for the page.

ASP.NET

ASP.Net Web Forms Model:

- ASP.Net web forms extend the event-driven model of interaction to the web applications.
- The browser submits a web form to the web server and the server returns a full markup page or HTML page in response.
- All client side user activities are forwarded to the server for stateful processing. The server processes the output of the client actions and triggers the reactions.
- Now, HTTP is a stateless protocol. ASP.Net framework helps in storing the information regarding the state of the application, which consists of:

1. Page state
2. Session state

ASP.NET

- The page state is the state of the client, i.e., the content of various input fields in the web form.
- The session state is the collective obtained from various pages the user visited and worked with, i.e., the overall session state.
- E.g. lets take an online shopping portal.
  - User adds items to a shopping cart.
  - Items are selected from a page, say the items page, and the total collected items and price are shown in a different page, say the cart page.
  - Only HTTP cannot keep track of all the information coming from various pages.
- ASP.Net session state and server side infrastructure keeps track of the information collected globally over a session.

ASP.NET

- The ASP.Net runtime carries the page state to and from the server across page requests while generating the ASP.Net runtime codes and incorporates the state of the server side components in hidden fields.
- This way the server becomes aware of the overall application state and operates in a two-tiered connected way.

ASP.NET

ASP.Net Component Model:

- The ASP.Net component model provides various building blocks of ASP.Net pages.
- Basically it is an object model, which describes: Server side counterparts of almost all HTML elements or tags, like <form> and <input>.
- Server controls, which help in developing complex user-interface for example the Calendar control or the Gridview control.
- ASP.Net is a technology, which works on the .Net framework that contains all web-related functionalities.
- The .Net framework is made of an object-oriented hierarchy.

ASP.NET

- An ASP.Net web application is made of pages.
- When a user requests an ASP.Net page, the IIS delegates the processing of the page to the ASP.Net runtime system.
- The ASP.Net runtime transforms the .aspx page into an instance of a class, which inherits from the base class Page of the .Net framework.
- Therefore, each ASP.Net page is an object and all its components i.e., the server-side controls are also objects.

ASP.NET - Environment Setup

- ASP.Net provides an abstraction layer on top of HTTP on which the web applications are built.
- It provides high-level entities like classes and components within an object-oriented paradigm.
- The key development tool for building ASP.Net applications and front ends is Visual Studio.
- Visual Studio is an integrated development environment for writing, compiling and debugging the code.
- It provides a complete set of development tools for building ASP.Net web applications, web services, desktop applications and mobile applications.

ASP.NET - Environment Setup

Projects and Solutions:

- A typical ASP.Net application consists of many items: the web content files (.aspx), source files (e.g., the .cs files), assemblies (e.g., the .dll files and .exe files), data source files (e.g., .mdb files), references, icons, user controls and miscellaneous other files and folders.
- All these files that make up the website are contained in a Solution.
- When a new website is created it automatically creates the solution and displays it in the solution explorer.
- Solutions may contain one or more projects. A project contains content files, source files, and other files like data sources and image files.
- Generally the contents of a project are compiled into an assembly as an executable file (.exe) or a dynamic link library (.dll) file.

ASP.NET

- Typically a project contains the following content files:

1. Page file (.aspx)
2. User control (.ascx)
3. Web service (.asmx)
4. Master page (.master)
5. Site map (.sitemap)
6. Website configuration file (.config)

ASP.NET - Life Cycle

- ASP.Net life cycle specifies, how:
  - ASP.Net processes pages to produce dynamic output
  - The application and its pages are instantiated and processed
  - ASP.Net compiles the pages dynamically
- The ASP.Net life cycle could be divided into two groups:
- Application Life Cycle
- Page Life Cycle

ASP.NET - Life Cycle

ASP.Net Application Life Cycle:

- User makes a request for accessing application resource, a page. Browser sends this request to the web server.
- A unified pipeline receives the first request and the following events take place:
  - An object of the ApplicationManager class is created.
  - An object of the HostingEnvironment class is created to provide information regarding the resources.
  - Top level items in the application are compiled.

ASP.NET - Life Cycle

- Response objects are created . the application objects: HttpContext, HttpRequest and HttpResponse are created and initialized.
- An instance of the HttpApplication object is created and assigned to the request.
- The request is processed by the HttpApplication class. Different events are raised by this class for processing the request.

ASP.NET - Life Cycle

ASP.Net Page Life Cycle:

- When a page is requested, it is loaded into the server memory, processed and sent to the browser.
- Then it is unloaded from the memory.
- At each of this steps, methods and events are available, which could be overridden according to the need of the application.
- In other words, you can write your own code to override the default code.
- The Page class creates a hierarchical tree of all the controls on the page.

ASP.NET - Life Cycle

- All the components on the page, except the directives are part of this control tree.
- You can see the control tree by adding trace= "true" to the Page directive. We will cover page directives and tracing under 'directives' and 'error handling'.

- The page life cycle phases are:

1. Initialization
2. Instantiation of the controls on the page
3. Restoration and maintenance of the state
4. Execution of the event handler codes
5. Page rendering

ASP.NET - Life Cycle

- Understanding the page cycle helps in writing codes for making some specific thing happen at any stage of the page life cycle.
- It also helps in writing custom controls and initializing them at right time, populate their properties with view-state data and run control behaviour code.

ASP.NET - Life Cycle

Following are the different stages of an ASP.Net page:

- **Page request .** when ASP.Net gets a page request, it decides whether to parse and compile the page or there would be a cached version of the page; accordingly the response is sent

- **Starting of page life cycle .** at this stage, the Request and Response objects are set.
- If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.

- **Page initialization .** at this stage, the controls on the page are assigned unique ID by setting the UniqueID property and themes are applied. For a new request postback data is loaded and the control properties are restored to the view-state values.

ASP.NET - Life Cycle

- **Validation .** Validate method of the validation control is called and if it runs successfully, the IsValid property of the page is set to true.
- **Postback event handling .** if the request is a postback (old request), the related event handler is called.
- **Page rendering .** at this stage, view state for the page and all controls are saved.
- The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Page's Response property.

- **Page load .** at this stage, control properties are set using the view state and control state values.
- **Unload .** the rendered page is sent to the client and page properties, such as Response and Request are unloaded and all cleanup done.

ASP.NET - Life Cycle

ASP.Net Page Life Cycle Events:

- **PreInit .** PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback.
- It sets the themes and master pages, creates dynamic controls and gets and sets profile property values.
- This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.
- **Init .** Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.
- **InitComplete .** InitComplete event allows tracking of view state. All the controls turn on view-state tracking.

ASP.NET - Life Cycle

- **LoadViewState .** LoadViewState event allows loading view state information into the controls.
- **LoadPostData .** during this phase, the contents of all the input fields defined with the <form> tag are processed.
- **PreLoad .** PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.
- **Load .** the Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.

ASP.NET - Life Cycle

- **LoadComplete .** the loading process is completed, control event handlers are run and page validation takes place.
- This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler.
- **PreRender .** the PreRender event occurs just before the output is rendered.
- By handling this event, pages and controls can perform any updates before the output is rendered.
- **PreRenderComplete .** as the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.

ASP.NET - Life Cycle

- **SaveStateComplete .** state of control on the page is saved. Personalization, control state and view state information is saved.
- The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.
- **UnLoad .** the UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself.
- Final cleanup is done and all resources and references, such as database connections, are freed.
- This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

ASP.NET

- An ASP.Net page is an object of the Page Class or inherited from it.
- All the controls on the pages are also objects of the related control class inherited from a parent Control class.
- An ASP.Net page is also a server side file saved with the **.aspx** extension.
- It is modular in nature and can be divided into the following core sections:

1. Page directives
2. Code Section
3. Page Layout

ASP.NET

```
<!-- directives -->
<% @Page Language="C#" %>
<!-- code section -->
<script runat="server">
private void convertoupper(object sender, EventArgs e) {
        string str = mytext.Value; changed_text.InnerHtml = str.ToUpper();
}
</script>
<!-- Layout -->
<html> <head> <title> Change to Upper Case </title> </head>
 <body> <h3> Conversion to Upper Case </h3>
<form runat="server"> <input runat="server" id="mytext" type="text" />
<input runat="server" id="button1" type="submit" value="Enter..."
    OnServerClick="convertoupper"/>
<hr /> <h3> Results: </h3>
<span runat="server" id="changed_text" />
</form> </body> </html>
```

ASP.NET

1. **Page directives:**

- The page directives set up the environments for the page to run.
- The @Page directive defines page-specific attributes used by the ASP.Net page parser and compiler.
- Page directives specify how the page should be processed, and which assumptions are to be taken about the page.
- It allows importing namespaces, loading assemblies and registering new controls with custom tag names and namespace prefixes.

ASP.NET

```
<!-- directives -->
<%@ Page Language="C#"  AutoEventWireup="true"
   CodeBehind="Default.aspx.cs" Inherits="firstexample._Default"
   %>


<!-- code section -->
<script runat="server">
private void convertoupper(object sender, EventArgs e)
{
}
</script>


<!-- Layout -->
<html> HTML PAGE
</html>
```

ASP.NET

2. **Code Section:**

- The code section provides the handlers for the page and control events along with other functions required.
- We mentioned that, ASP.Net follows an object model. Now, these objects raises events when something happens on the user interface, like a user clicks a button or moves the cursor.
- How these events should be handled? That code is provided in the event handlers of the controls, which are nothing but functions bound to the controls.
- The code section or the code behind file provides all these event handler routines, and other functions used by the developer. The page code could be precompiled and deployed in the form of a binary assembly.

ASP.NET - Life Cycle

3. **Page Layout:**

- The page layout provides the interface of the page.
- It contains the server controls, text, inline JavaScript and HTML tags:
- The previous code snippet provides a sample ASP.Net page explaining page directives, code section and page layout written in C#:

ASP.NET - Event Handling

- Event is an action or occurrence like mouse click, key press, mouse movements, or any system generated notification.
- The processes communicate through events.
- For example, Interrupts are system generated events. When events occur the application should be able to respond to it.
- In ASP.Net an event is raised on the client, and handled in the server.
- For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.

ASP.NET - Event Handling

- The server has a subroutine describing what to do when the event is raised; it is called the event-handler.
- Therefore, when the event message is transmitted to the server, it checks whether the Click event has an associated event handler, and if it has, the event handler is executed.

Event Arguments:

- ASP.Net event handlers generally take two parameters and return void.
- The first parameter represents the object raising the event and the second parameter is called the event argument.

ASP.NET - Event Handling

- **Syntax**

  private void EventName(object sender, EventArgs e);

- **Application and Session Events:**

  The most important application events are:

  - **Application_Start** it is raised when the application/website is started
  - **Application_End** it is raised when the application/website is started / stopped

- The most used Session events are:
  - **Session_Start** . it is raised when a user first requests a page from the application
  - **Session_End** . it is raised when the session ends

ASP.NET - Event Handling

- **Common page and control events are:**

1. **DataBinding** : raised when a control is bound to a data source
2. **Disposed**: when the page or the control is released
3. **Error**: it is a page event, occurs when an unhandled exception is thrown
4. **Init**: raised when the page or the control is initialized
5. **Load** . raised when the page or a control is loaded
6. **PreRender**: raised when the page or the control is to be rendered
7. **Unload**: raised when the page or control is unloaded from memory

ASP.NET - Event Handling

Event	Attribute	Controls
Click	OnClick	Button, image button, link button, image map
Command	OnCommand	Button, image button, link button
TextChanged	OnTextChanged	Text box
SelectedIndexChanged	OnSelectedIndexChanged	Drop-down list, list box, radio button list, check box list.
CheckedChanged	OnCheckedChanged	Check box, radio button

ASP.NET - Event Handling

- Some events cause the form to be posted back to the server immediately, these are called the postback events.
- For example, the click events like, Button.Click.
- Some events are not posted back to the server immediately, these are called non-postback events.
- For example, the change events or selection events, such as, TextBox.
- TextChanged or CheckBox.CheckedChanged.
- The nonpostback events could be made to post back immediately by setting their AutoPostBack property to true.

ASP.NET - Event Handling

Default Events:

- The default event for the Page object is the Load event.
- Similarly every control has a default event. For example, default event for the button control is the Click event.
- The default event handler could be created in Visual Studio, just by double clicking the control in design view.

ASP.NET - Event Handling

Control	Default Event
AdRotator	AdCreated
BulletedList	Click
Button	Click
Calender	SelectionChanged
CheckBox	CheckedChanged
CheckBoxList	SelectedIndexChanged
DataGrid	SelectedIndexChanged
DataList	SelectedIndexChanged

ASP.NET - Event Handling

Control	Default Event
DropDownList	SelectedIndexChanged
HyperLink	Click
ImageButton	Click
ImageMap	Click
LinkButton	Click
ListBox	SelectedIndexChanged
Menu	MenuItemClick
RadioButton	CheckedChanged

ASP.NET - HTML Server Controls

- The HTML server controls are basically the original HTML controls but enhanced to enable server side processing.
- The HTML controls like the header tags, anchor tags and input elements are not processed by the server but sent to the browser for display.
- They are specifically converted to a server control by adding the attribute runat="server" and adding an id attribute to make them available for server-side processing.
- For example, consider the HTML input control:

  <input type="text" size="40">
- It could be converted to a server control, by adding the runat and id attribute:

  <input type="text" id="testtext" size="40“ runat="server">

ASP.NET - HTML Server Controls

Advantages of using HTML Server Controls

- Although ASP.Net server controls can perform every job accomplished by the HTML server controls, the later controls are useful in the following cases:
  - Using static tables for layout purposes
  - Converting a HTML page to run under ASP.Net

ASP.NET - HTML Server Controls

Control Name	HTML tag
HtmlHead	<head>element
HtmlInputButton	<input type=button|submit|reset>
HtmlInputCheckbox	<input type=checkbox>
HtmlInputFile	<input type = file>
HtmlInputHidden	<input type = hidden>
HtmlInputImage	<input type = image>
HtmlInputPassword	<input type = password>
HtmlInputRadioButton	<input type = radio>
HtmlInputReset	<input type = reset>
HtmlText	<input type = text|password>

ASP.NET - HTML Server Controls

Control Name	HTML tag
HtmlImage	<img> element
HtmlLink	<link> element
HtmlAnchor	<a> element
HtmlButton	<button> element
HtmlButton	<button> element
HtmlForm	<form> element
HtmlTable	<table> element
HtmlTableCell	<td> and <th>
HtmlTableRow	<tr> element
HtmlTitle	<title> element
HtmlSelect	<select> element
HtmlGenericControl	All HTML controls not listed

ASP.NET - Server Controls

- Controls are small building blocks of the graphical user interface, which includes text boxes, buttons, check boxes, list boxes, labels and numerous other tools, using which users can enter data, make selections and indicate their preferences.
- Controls are also used for structural jobs, like validation, data access, security, creating master pages, data manipulation.
- ASP.Net uses five types of web controls, which are:
  - HTML controls
  - HTML Server controls
  - ASP.Net Server controls
  - ASP.Net Ajax Server controls
  - User controls and custom controls

ASP.NET - Server Controls

ASP.Net **server controls** are the primary controls used in ASP.Net. These controls again could be grouped into the following categories:

- **Validation controls** - these are used to validate user input and work by running client-side script
- **Data source controls** - these controls provides data binding to different data sources
- **Data view controls** - these are various lists and tables, which can bind to data from data sources for display
- **Personalization controls** - these are used for personalization of a page according to the user's preference, based on user information

ASP.NET - Server Controls

- **Login and security controls** - these controls provide user authentication
- **Master pages** - these provides consistent layout and interface throughout the application
- **Navigation controls** - these helps in navigation, for example, the menus, tree view etc.
- **Rich controls** - these implements special features, for example, AdRotator control, FileUpload control, Calendar control etc.

ASP.NET - Server Controls

Properties of the Server Controls

- The ASP.Net server controls with a visual aspect are derived from the WebControl class and inherit all the properties, events and methods of this class.
- The WebControl class itself and some other server controls that are not visually rendered, e.g., the PlaceHolder control or XML control etc., are derived from the System.Web.UI.Control class.
- ASP.Net server controls inherit all properties, events and methods of the WebControl and System.Web.UI.Control class.

ASP.NET - Server Controls

Property	Description
AccessKey	Pressing this key with the Alt key moves focus to the control
Attributes	It's the collection of arbitrary attributes (for rendering only) that do not correspond to properties on the control.
BackColor	Background colour.
BindingContainer	The control that contains this control's data binding.
BorderColor	Border colour.
BorderStyle	Border style.

ASP.NET - Server Controls

Property	Description
BorderWidth	Border width.
CausesValidation	Indicates if it causes validation.
ChildControlCreated	It indicates whether the server control's child controls have been created.
ClientID	Control ID for HTML markup.
Context	The HttpContext object associated with the server control.
Controls	Collection of all controls contained within the control

ASP.NET - Server Controls

Property	Description
ControlStyle	The style of the Web server control.
CssClass	CSS class
DataItemContainer	Gets a reference to the naming container if the naming container implements IDataItemContainer.
DataKeysContainer	Gets a reference to the naming container if the naming container implements IDataKeysControl.
DesignMode	It indicates whether the control is being used on a design surface.

ASP.NET - Server Controls

Property	Description
DisabledCssClass	Gets or sets the CSS class to apply to the rendered HTML element when the control is disabled.
Enabled	Indicates whether the control is grayed out
EnableTheming	Indicates whether theming applies to the control.
EnableViewState	Indicates whether the view state of the control is maintained.
Events	Gets a list of event handler delegates for the control.
Font	Font .

ASP.NET - Server Controls

Property	Description
Forecolor	Foreground colour.
HasAttributes	Indicates whether the control has attributes set.
HasChildViewState	indicates whether the current server control's child controls have any saved view-state settings.
Height	Height in pixels or %.
ID	Identifier for the control.
IsChildControlStateCleared	Indicates whether controls contained within this control have control state.

ASP.NET - Server Controls

Property	Description
IsEnabled	Gets a value indicating whether the control is enabled
IsTrackingViewState	It indicates whether the server control is saving changes to its view state.
IsViewStateEnabled	It indicates whether view state is enabled for this control.
LoadViewStateById	It indicates whether the control participates in loading its view state by ID instead of index.
Page	Page containing the control.
Parent	Parent control.

ASP.NET - Server Controls

Property	Description
RenderingCompatibility	It specifies the ASP.NET version that rendered HTML will be compatible with.
Site	The container that hosts the current control when rendered on a design surface.
SkinID	Gets or sets the skin to apply to the control.
Style	Gets a collection of text attributes that will be rendered as a style attribute on the outer tag of the Web server control.

ASP.NET - Server Controls

Property	Description
TabIndex	Gets or sets the tab index of the Web server control.
TagKey	Gets the HtmlTextWriterTag value that corresponds to this Web server control.
TagName	Gets the name of the control tag.
TemplateControl	The template that contains this control.
TemplateSourceDirectory	Gets the virtual directory of the Page or control containing this control.
ToolTip	Gets or sets the text displayed when the mouse pointer hovers over the Web server control.

ASP.NET - Server Controls

Property	Description
UniqueID	Unique identifier
ViewState	Gets a dictionary of state information that allows you to save and restore the view state of a server control across multiple requests for the same page.
ViewStateIgnoreCase	It indicates whether the StateBag object is case-insensitive.
ViewStateMode	Gets or sets the view-state mode of this control.
Visible	It indicates whether a server control is visible.
Width	Gets or sets the width of the Web server control.

ASP.NET - Server Controls

Method	Description
CreateControlStyle	Creates the style object that is used to implement all style related properties.
DataBind	Binds a data source to the server control and all its child controls.
DataBind(Boolean)	Binds a data source to the server control and all its child controls with an option to raise the DataBinding event.
DataBindChildren	Binds a data source to the server control's child controls.
Dispose	Enables a server control to perform final clean up before it is released from memory.

ASP.NET - Server Controls

Method	Description
EnsureChildControls	Determines whether the server control contains child controls. If it does not, it creates child controls.
EnsureID	Creates an identifier for controls that do not have an identifier.
Equals(Object)	Determines whether the specified Object is equal to the current Object.
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.

ASP.NET - Server Controls

Method	Description
FindControl(String)	Searches the current naming container for a server control with the specified id parameter.
FindControl(String, Int32)	Searches the current naming container for a server control with the specified id and an integer
Focus	Sets input focus to a control.
GetDesignModeState	Gets design-time data for a control.
GetType	Gets the Type of the current instance.
GetUniqueIDRelativeTo	Returns the prefixed portion of the UniqueID property of the specified control.

ASP.NET - Server Controls

Method	Description
LoadControlState	Restores control-state information.
LoadViewState	Restores view-state information.
OnDataBinding	Raises the data binding event.
OnInit	Raises the Init event.
OnLoad	Raises the Load event.
OnPreRender	Raises the PreRender event.
OnUnload	Raises the Unload event.
OpenFile	Gets a Stream used to read a file
RemovedControl	Called after a child control is removed from the Controls collection of the Control object.

ASP.NET - Server Controls

Method	Description
SaveControlState	Saves any server control state changes that have occurred since the time the page was posted back to the server.
SaveViewState	Saves any state that was modified after the TrackViewState method was invoked.
SetDesignModeState	Sets design-time data for a control.
ToString	Returns a String that represents the current object.

ASP.NET - Server Controls

- Regular ID that's used in the designer
- Unique ID that uniquely identifies it for cases where repeater-type controls are used
- Client ID that uniquely identifies it on the client

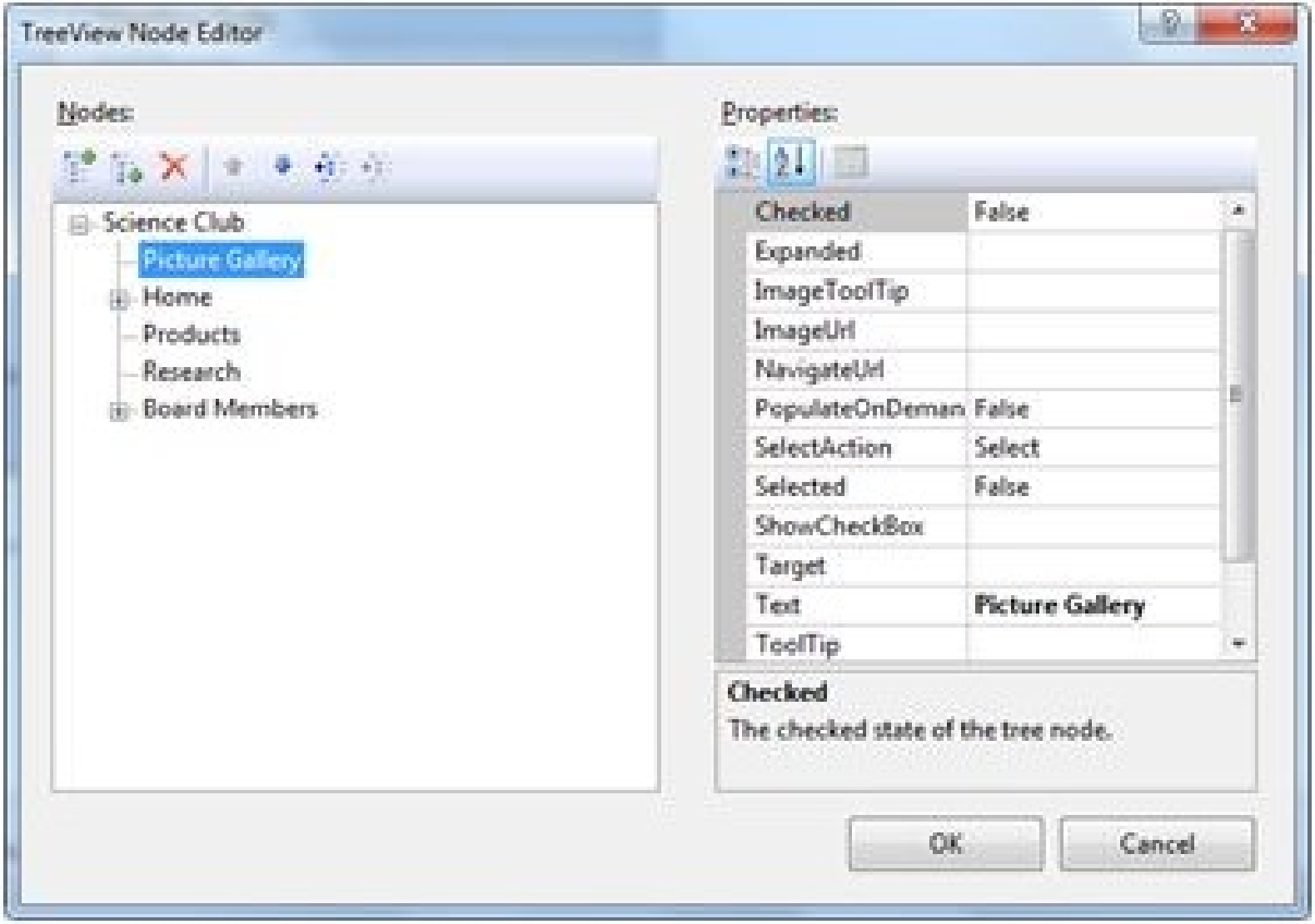
ASP.NET - Server Controls

- **Differnece between Clientid and uniqueid**
- ClientID is used for javascript and UniqueId for server side and that ClientID uses an underscore (_) and UniqueId uses a dollar sign ($) in asp.net 2.0.
- You probably know UniqueID is used with name attribute and ClientId with id attribute of rendered HTML tag.
- UniqueID uses colon as separator.
- On the other hand ClientId uses underscore as separator, because colon is not allowed in JavaScript variable names.
- ClientID is indeed also unique on the Page as UniqueID is, but ClientID is targeted at client-side processing and UniqueID for server-side (pretty obvious),the latter especially to route for postback data and events with composite controls

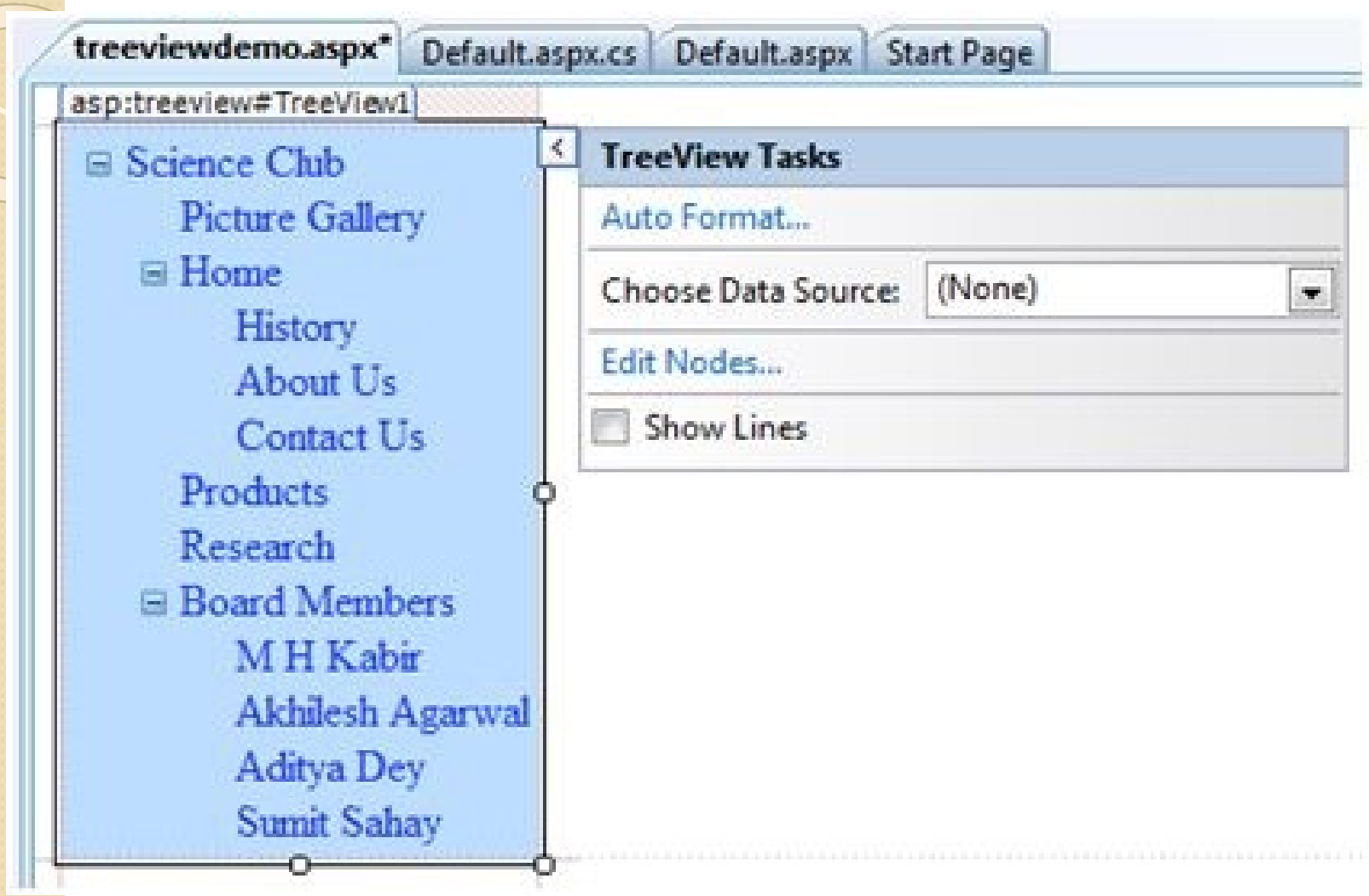
ASP.NET - Server Controls

- Differnece between ID and UniqueId
- For example, if you include an ASP.NET Label Web server control in a Repeater server control, and assign the Label control an ID property value of MyLabel, and the Repeater an ID of MyRepeater.
- If you bind data to the Repeater to an ArrayList object with three entries, the resulting UniqueID properties for each instance of the Label server controls are
- MyRepeater:ctl0:MyLabel, MyRepeater:Ctl1:MyLabel, and MyRepeater:Ctl2:MyLabel.

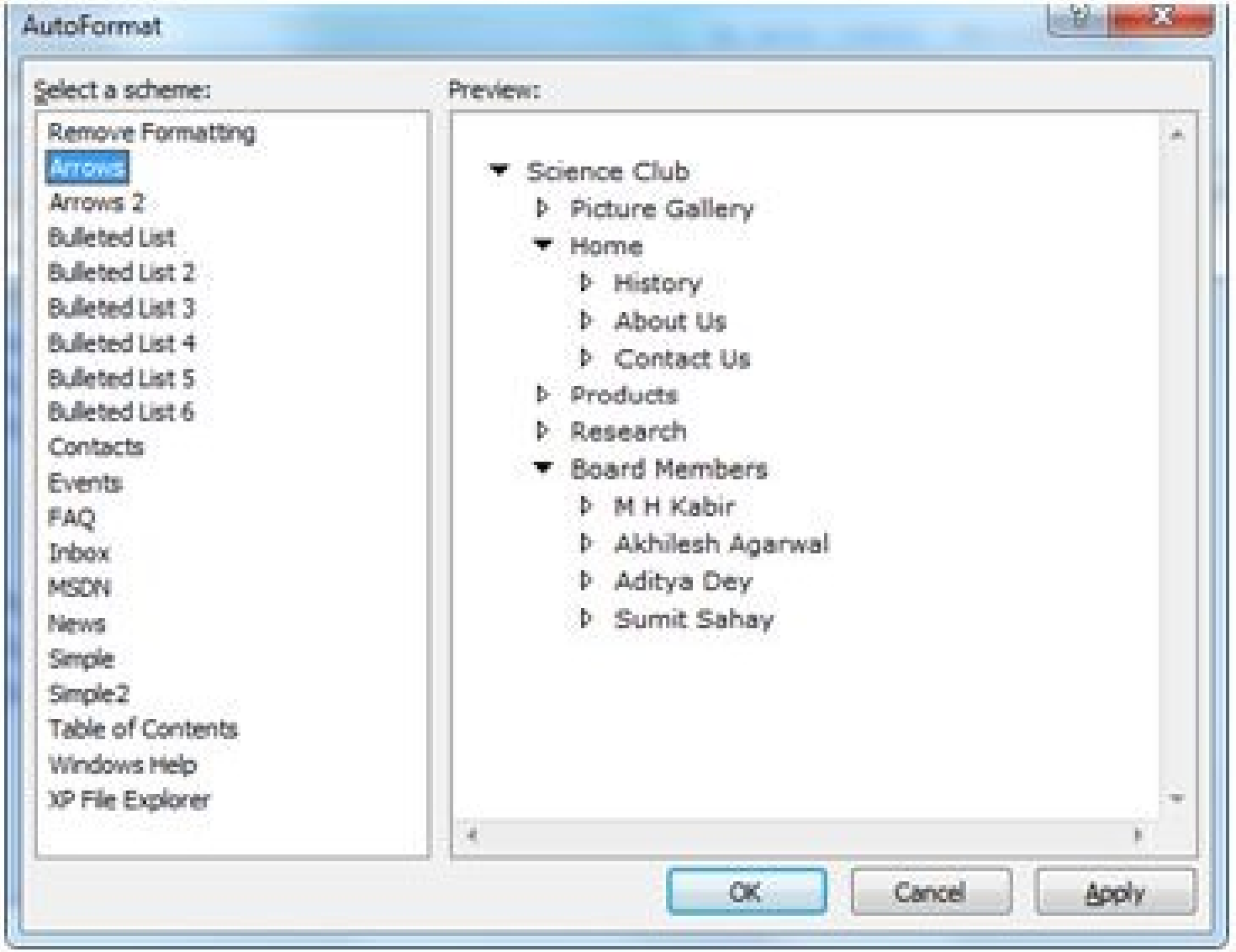
ASP.NET – Server Controls



ASP.NET – Server Controls



ASP.NET – Server Controls



ASP.NET – Server Controls