Unit 4

ASP.net

Creating a web application
- Open VS
- File menu, New Project
- Template → C# → web
- Choose ASP.Net ~~tobe~~ web Application template
- Name your project
- Select project template as web Form template
- VS creates a new project that includes prebuilt functionality based on web forms template
- It provides Home.aspx, Contact and ~~a~~ About with membership functionality - registers users
- Saves credentials - to log in your website

Features
- Cross-lang. Interoperability - code/interact with another code
- Multi-lang. Support - DUH
- Automatic Resource Management
- Debugging
- Elimination of DLL Hell - allows multiple versions of same DLL to coexist
- Security - lets dev decide - component can ~~acce~~ access sensitive components.
- Support For HTML 5

Maybe Not Important IDK

Structure of an Application

- Includes concepts of application domain, application lifetime, and application directory structure.
- App domain:- virtual boundary inside which application runs.
- App lifetime:- span of time for which the domain exists
- App directory structure :- Specifies directory structure that organizes various entities associated to an application (references, resources, code files etc).

→ App domain: Implemented by CLR (Common Language Runtime). Objective - prevent simultaneously running apps from entering each other's domain.
Entities stored are
Web Pages, Web Services, Code-behind files, Configuration files, Global.asax files

→ App Lifetime: might encounter restarting. Can restart if Excess requests, Excess memory usage, lengthy lifetime etc. Restarts performed - to recycle domain - helps in performance monitoring

→ App Directory Structure: Seggregates resources, enhancing productivity of devs. Some built in directories are
Bin, App-Code, App-GlobalResources,
App-LocalResources, App-Data, App-Browsers. etc.

☆ Global.asax App file - ASP.NET App file contains code which is executed when certain events occur. Events/States specified in this file - can be accessed by any resource in the web app.

- written in same way as Web Forms, difference - does not contain HTML & ASP.NET tags but contains methods with predefined names.

☆ States

- Eliminate drawback - losing data when reloading webpage. Can preserve state at server or client side. States needed in E-com websites to track Req's.

Methods to store state info

→ Hidden fields - This control - not visible - web app viewed in browser

→ Cookies - Text files which store data = user ID, preferences at client end. When browser req. web page - cookie sent too.

→ Query strings - Info strings - added at end of URL - not secure - values exposed.

3 states

Application State

→ Used to store data - corresponding to ALL variables of an ASP.NET web application. Data here is stored once - read several times. Access this data using HttpApplication class property. This class - provides lock method - used to ensure only ~~per~~user has access.

Session State

→ Each client - distinct session with WebServer. Specific info associated with each session, defined in ~~ses~~ ~~section~~ <SessionState> section of the web.config file. Data specific to users - session variables. Different variables created for each user session.

Control = Element (i guess)

View State

→ Stores page specific info - when it is postedback to the server. When page is processed - current state of page hashed into string - saved as hidden field in a page. ViewState property is used to save the view state of each control used in a page. Maintained in web page by default.

★★ Standard Controls

i) Label Control: Used to display info on Web Form.
Properties: Text, Font, ForeColor, Height, BackColor, Width
Syntax
<asp:Label ID='Label1' Text="Hello" > </asp:Label>

ii) Textbox Control: used to input data (Text)
Properties: Text, TextMode (Single, Multiline, Password, Number), Rows, Columns, AutoPostBack, Textchange.
Syntax
<asp:TextBox ID='TextBox1' TextMode="Number"></asp:TextBox>

iii) Button Control: Used to create an Event / Send req to web server
Properties: Text, Click (Event), Command (Event)
On Client Click.
Syntax
<asp:Button ID='Button1' OnClick="Button1_Click / >
in the index.aspx.cs, Button1_Click function will ~~apear~~ appear
protected void Button1_Click(object sender, EventArgs e)
{ Label1.Text = "Hi"; }

★ NavigateUrl Property: Links / Sets URL to a web page control

(iv) DropDownList control: used to select from a list of predefined data items.
Properties: BorderWidth, BorderColor, SelectedIndex
Syntax

```
<asp: DropDownList ID="DropDownList 1" >
      <asp: ListItem Value="Saish> Saish </asp: ListItem>
            :
<asp:DropDownList>
```

To access the selected item use
→ Label1.Text = DropDownList1.SelectedItem.Value

v) CheckBoxList Control / RadioButtonList Control
Used to display a number of CheckBoxes / RadioButtons in a column.
Syntax

```
<asp: RadioButton/CheckBox List ID=" " List1" >
       <asp: ListItem Value="Me"> Me </asp:ListItem>
       <asp:ListItem Value="you"> You </asp: ListItem>
</asp: " " List>
```

★★ Navigation Controls - SiteMapPath, Menu Control, TreeView
1) Site Map Path
→ XML files - used to describe logical structure of web app.
Defines layout of pages - how they relate to each other.
Files are defined with .sitemap extension.
<sitemap> element is the root node of the sitemap file
Has 3 Attributes. Title, URL, Description.
Title : Text desc. of link
URL : location of valid physical file.
Description: used for tooltip of the link

Displays navigation path of current page. Path acts as links to previous page. Some Properties

- → PathSeperator : used to get /Set path seperator text
- → NodeStyle : Set Styles of all displayed Nodes
- → RootNodeStyle : " " Absolute Root Node
- → PathDirection : direction of links in output.

2) Menu Control

→ Used to display menu in web pages. Used with SiteMap Data Source Control for Navigating the Website. Display two types of Menus : Static menu : Displayed by default, root menu always displayed. Dynamic Menu : When mouse pointer moves over the parent menu that contains a sub menu.

Properties

- → CSS Class : Specify CSS class for the control
- → ImgUrl : Specify image apperaring next to menu item
- → Orientation : Horizontal/Vertical
- → Tooltip : ~~Shows~~ ~~to~~ Specify tooltip while hovering over items
- → Text : Text to display
- → NavigateUrl : Specify target location

3) ~~T~~ TreeView Control

→ Another nav control - display data in heirarchical list manner. When displayed first time, displays all nodes. Can be controlled by setting ExpandDepth

Properties

- → DataSourceID : specify data source to be used
- → ShowLines : Specify lines to connenct individual items
- → CssClass : Specify CSS class for control
- → ExpandDepth : Specify level at which items in tree are expanded

★★ Validation Controls

i) RequiredField Validation Control
Simple validation control, checks if data is entered for the input control. Mush be present if wish to enforce MandatoryField rule.

→ Syntax
<asp: RequiredFieldValidator ID="RequiredFieldValidator"
~~Error mess~~ ErrorMessage = "Invalid"
ControlToValidate = "TextBox1" >
</ asp: " >

ii) CompareValidator Control
Allows for comparison of data entered in an input control with a constant value or another control. Used to confirm passwords, case Sensitive.

→ Syntax
<asp: ComparValidator ID="CompareValidator1"
Error Message = "Do not Match" ControlToValidate="TextBox1"
ControlToCompare = "TextBox2" / "Hello" > </asp: " >
Optional: Can Specify Operator Property are Equal, LessThan, GreaterThan.
Type Property = "Text"/"Integer" etc.

iii) RangeValidator Control
Checks to see if control value within valid range. necessary values MaximumValue, MinimumValue, Type

→ Syntax
<asp: RangeValidator ID= " "
ControlToCompare = "TextBox1" MaximumValue = "100"
MinimumValue = "10" Type = "Integer" > </asp: "" >

iv) RegularExpressionValidator - DUH

v) ~~Sin~~ CustomValidator - Uses a function to validate through a property
OnServerValidate = "UserCustomValidate" ← function.
if false, returns error. use foreach (char ch in str)
where str = args.Value. return a Bool value
Set args.IsValid = true; when validation complete

★★ Database Controls

i) Gridview Control
→ used to display values of datasource in a table.
Column = field, Row = Record. (Represents)
Features
→ Binding to data source, SqlDataSource
→ Built in sorting capabilities
→ Built in update Delete ~~feat~~ capabilities.
→ " row selection "
→ Multiple key fields.
Each column is represented by a DataControlField.
Default Autogenerate columns set to true. Boundfield is used to display value of field in a data source. -
Others Button field, Imagefield, CheckBox Field etc.

ii) Datalist Control

- [illegible] Databound Control to display and manipulate data in a web app.

→ Appearance controlled by template. (content too)

→ ~~At minimum Item template~~

→ Display direction of a Datalist can be vertical or horizontal.

→ The function Datalist() initializes a new instance of the DataLists class.

→ At minimum, ItemTemplate needs to be defined to display items in ~~a~~ the List.

iii) Details View

→ used to display a single record from a data source in a table. Each field of record - displayed in a row.

→ can be used with Gridview for master-detail

→ features are

- Binding to datasource

→ Built in inserting capabilities

→ " updating and deleting "

→ " paging "

→ Customize appearance - themes and styles

iv) Form View

→ used to display " ~~Sam~~ similar to Details View, But uses user defined templates instead of row fields

→ Flexibility in controlling how data is displayed.

Features - Same as Details View.

v) ListView

→ used to display " similar to Gidview. But uses user defined templates instead of row fields.

→ flexibity similar to Form View

→ features same as Gridview

vi) Repeater

→ Basic templated data bound list

→ has no built in layout or styles, - need to be declared

→ Allows to split markup tags across templates

→ To create table include
<table> ~~beig~~ begin table tag in HeaderTemplate
<tr> single table row & " ItemTemplate
</table> end " " " FooterTemplate.

→ Has no Selection ~~cap~~ /editing capabilities

→ Use Item Command event to process events (control).

vii) Datapager

- used to page data, display navigation controls for data-bound controls.

→ Associate NavigationControl with DataBound control with PageControlID.

→ Customize no. of items displayed per datapage - Page Size

→ Also can change how a page is submitted to the server

→ To display navigation controls, must add page field.

viii) ~~SQ~~ SqlDataSource

- → represents datain a SQL relational database to data bound controls.
- → Can also be used to retrieve data from a database
- → Display, edit, Sort etc with little to no code
- → To connekt to a database, ConnectionString must be set to a valid connection string
- → Supports any database- using an ADO.NET provider.
- → retrieves data when Select method is called.