DOT NET TECHNOLOGIES

UNIT-3

Rich controls:

In addition to the preceding controls, the ASP.NET page framework provides a few, task-specific controls called rich controls. Rich controls are built with multiple HTML elements and contain rich functionality.

Rich controls in ASP.NET provide additional functionality for creating rich applications. In this article, we shall explore the following controls:

1. AdRotator control
2. MultiView control
3. Panel control
4. Calendar control

1. AdRotater control:

The AdRotator control selects from the banner advertisements. They are added in an XML file. The file is known as advertisement file. The control states the file and the window used for linking the file.

Syntax for AdRotator control:

<asp:AdRotator AdvertismentFile="file1.xml" runat="server" Target="_top" />

The advertisement file consist information about the advertisements to be displayed. The data is saved in a structured format through the formatted tags.
Consider a code snippet demonstrating the XML file.

Code:

```
<STUDENT>
   <STUDNAME>John</STUDNAME>
   <CLASS>12</CLASS>
   <MARKS>50</MARKS>
</STUDENT>
```

Properties of the AdRotator control are:

- **Data Member:** The name of the list of data specified by the user used for binding in absence of the advertisement file.

- **Advertisement File:** The path for the advertisement file is defined
- **DataSource:** The control where the data is accessed
- **Alternate Text Field:** The field name where the alternate text is provided
- **Font:** The font associated with the banner control is defined
- **Target:** The window used for displaying the content of the page
- **NavigateUrlField:** The name of the field where the URL for navigation is defined

Events of the AdRotator control are:

1. **Data Binding:** The server control is used for binding with the data source
2. **Ad Created:** The event is raised when the trip to the server is done, the page is rendered later.
3. **Init:** The server control is initialized during the page life cycle
4. **Load:** The server control is loaded to the Page
5. **Unload:** The server control is unloaded from the memory

2. Multiview control:

The list of View controls forms the multiview control. Every page consists of view control which is used for group management. Only one view is displayed to the user.

Syntax:

Code:

```
<asp:MultiView ID="MultView1" runat="server">
   <asp:View ID="view1" runat="server"></asp:View>
</asp:MultiView>
```


Properties of Multiview control:

1. **ID:** The unique identifier used for identifying the control on the web page
2. **Views:** The group of view controls in a Multiview control
3. **ActiveViewIndex:** It is zero based index used for representing the active view.

Methods of Multiview control:

GetActiveView: The active view is accessed by the user

SetActiveView: The active view is assigned by the user

Events of Multiview control:

Activate: It is raised by the active view

Deactivate: Used by an inactive view

Active View Changed: When the view is modified by the user, it is raised

3. Panel control:

The panel control acts storage for all the controls used in the page. The appearance for the controls is managed.

Syntax:

Code:

```
<asp:Panel ID="Panel1" runat="server">
<asp:Panel>
```

Properties of Panel control:

1. **Direction:** The text direction of the control in the panel is defined
2. **BackImageUrl:** The background image of the post is stated
3. **Grouping Text:** The grouping of the text as a field is defined
4. **Wrap:** The text wrapping is performed
5. **ScrollBars:** The scrollbars are assigned within the panel

4. Calendar control

Calendar control is used for providing a month display to the user. User can view week, month, days of the week, dates, etc.

Syntax:

Code:

```
<asp:Calendar ID="calendar1" runat="server">
</asp:Calendar>
```

Properties of the calendar control:

1. **Cell Padding:** The space between the data and the border of the cell is defined
2. **Caption:** The caption for the control is defined
3. **Day Header Style:** The day of the week is displayed using the style properties
4. **First Day of Week:** The day of the week is displayed
5. **Pre-Month Text:** The text for the previous month navigation control
6. **Selected Dates:** The collection of DateTime objects for the selected dates
7. **Select Month Text:** The text for month element from the selector column
8. **Title Format:** The format for the title section is defined
9. **Visible Date:** The date for the month to be displayed is defined
10. **Today's Date:** The value for today's date is defined

Events for Calendar control:

- **Day Render:** When the data cell of the control is rendered
- **Selection Changed:** When user selects from the calendar, the event is raised

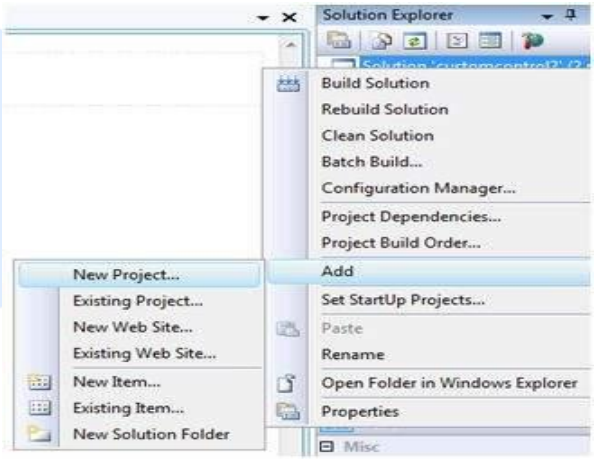
Custom Controls:

Custom controls are deployed as individual assemblies. They are compiled into a Dynamic Link Library (DLL) and used as any other ASP.NET server control. They could be created in either of the following way:

- By deriving a custom control from an existing control
- By composing a new custom control combing two or more existing controls.
- By deriving from the base control class.

To understand the concept, let us create a custom control, which will simply render a text message on the browser. To create this control, take the following steps:

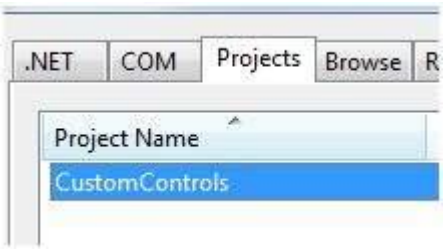
Create a new website. Right click the solution (not the project) at the top of the tree in the Solution Explorer.



In the New Project dialog box, select ASP.NET Server Control from the project templates.

The above step adds a new project and creates a complete custom control to the solution, called ServerControl1. In this example, let us name the project CustomControls. To use this control, this must be added as a reference to the web site before registering it on a page. To add a reference to the existing project, right click on the project (not the solution), and click Add Reference.

Select the Custom Controls project from the Projects tab of the Add Reference dialog box. The Solution Explorer should show the reference.



Validation controls:

Validation is important part of any web application. User's input must always be validated before sending across different layers of the application.

Validation controls are used to:

- Implement presentation logic.
- To validate user input data.
- Data format, data type and data range are used for validation.

There are six types of validation controls in ASP.NET:

1. Required Field Validation Control
2. Compare Validator Control
3. Range Validator Control
4. Regular Expression Validator Control
5. Custom Validator Control
6. Validation Summary

1. RequiredFieldValidation Control

The Required Field Validator control is simple validation control, which checks to see if the data is entered for the input control. You can have a Required Field Validator control for each form element on which you wish to enforce Mandatory Field rule.

2. Compare Validator Control

The CompareValidator control allows you to make comparison to compare data entered in an input control with a constant value or a value in a different control.

It can most commonly be used when you need to confirm password entered by the user at the registration time. The data is always case sensitive.

3. Range Validator Control

The RangeValidator Server Control is another validator control, which checks to see if a control value is within a valid range. The attributes that are necessary to this control are: Maximum Value, Minimum Value, and Type.

4. Regular Expression Validator Control

A regular expression is a powerful pattern matching language that can be used to identify simple and complex characters sequence that would otherwise require writing code to perform.

Using Regular Expression Validator server control, you can check a user's input based on a pattern that you define using a regular expression.

It is used to validate complex expressions. These expressions can be phone number, email address, zip code and many more. Using Regular Expression Validator is very simple. Simply set the Validation Expression property to any type of expression you want and it will validate it.

If you don't find your desired regular expression, you can create your custom one

5. Custom Validator Control

You can solve your purpose with ASP.NET validation control. But if you still don't find solution you can create your own custom validator control.

The CustomValidator Control can be used on client side and server side. JavaScript is used to do client validation and you can use any .NET language to do server-side validation.

I will explain you CustomValidator using server side. You should rely more on server-side validation.

To write CustomValidator on server side you override Server Validate event.

6. Validation Summary

ASP.NET has provided an additional control that complements the validator controls.

The Validation Summary control is reporting control, which is used by the other validation controls on a page.

You can use this validation control to consolidate errors reporting for all the validation errors that occur on a page instead of leaving this up to each and every individual validation control.

The validation summary control will collect all the error messages of all the non-valid controls and put them in a tidy list.

Both Error Message and Text properties are used to display error messages. Text error message have precedence.

If you are using Validation Summary than only Error Message and Text property is used.

Debugging:

Debugging allows the developers to see how the code works in a step-by-step manner, how the values of the variables change, how the objects are created and destroyed, etc.

When the site is executed for the first time, Visual Studio displays a prompt asking whether it should be enabled for debugging:

When debugging is enabled, the following lines of codes are shown in the web.config:

```
<system.web>
  <compilation debug="true">
    <assemblies>
    ..............
    </assemblies>
  </compilation>
</system.web>
```


Breakpoints

Breakpoints specifies the runtime to run a specific line of code and then stop execution so that the code could be examined and perform various debugging jobs such as, changing the value of the variables, step through the codes, moving in and out of functions and methods etc.

- To set a breakpoint, right click on the code and choose insert break point.
- Next when you execute the code, you can observe its behavior.
- At this stage, you can step through the code, observe the execution flow and examine the value of the variables, properties, objects, etc.
- You can modify the properties of the breakpoint from the Properties menu obtained by right clicking the breakpoint glyph.

- The location dialog box shows the location of the file, line number and the character number of the selected code. The condition menu item allows you to enter a valid expression, which is evaluated when the program execution reaches the breakpoint.
- The Hit Count menu item displays a dialog box that shows the number of times the break point has been executed.
- Clicking on any option presented by the drop-down list opens an edit field where a target hit count is entered. This is particularly helpful in analyzing loop constructs in code.
- The Filter menu item allows setting a filter for specifying machines, processes, or threads or any combination, for which the breakpoint will be effective.
- The When Hit menu item allows you to specify what to do when the break point is hit.

The Debug Windows:

Visual Studio provides the following debug windows, each of which shows some program information. The following table lists the windows:

Window	Description
Immediate	Displays variables and expressions.
Autos	Displays all variables in the current and previous statements.
Locals	Displays all variables in the current context.
Watch	Displays up to four different sets of variables.
Call Stack	Displays all methods in the call stack.
Threads	Displays and control threads.

Deploying projects with Business objects:

The entities are mapped to the database tables and object-relational mapping (ORM) frameworks like Entity Framework, NHibernate, etc. are used to retrieve and save the data into a database. The business object contains both state (data) and behavior that is logic specific to the business. Let us understand how to use Business Objects as Model in ASP.NET MVC Application.

Steps:

1. Create the Required Database

Please use the below SQL script to create and populate the Employee table with some test data. Also, we are creating one stored procedure to retrieve the employee data.

2. Add a Class Library project with Name="BusinessLayer" to the Solution

Right-click on the Solution Folder => Add => New Project as shown in the below image.

From the new project window, select Visual C# from Installed Template from the left pane and then select Class Library Template from the middle pane.

Now it will add the BusinessLayer class library project to our existing solution.

3: Adding Models to the Class Library Project

Right-click on the business layer class library project and add a class file with the name Employee.cs. Once you created the Employee class then copy and paste the following code into it. The following class is very straightforward. We simply created the class with 6 properties.

4: Adding Required References

Right-click on the "References" folder of the business layer class library project and add a reference to the "System.Configuration" assembly. This is required as we want to read the connection string from the web config file using the ConfigurationManager class and this class belongs to System.Configuration namespace.

5: Adding EmployeeBusinessLayer class

Right-click on the business layer class library project and add a class file with the name EmployeeBusinessLayer.cs. Once you created the EmployeeBusinessLayer class then copy and paste the following code into it. In the following class, we define one method i..e. GetAllEmployess(). This method is used to get the employee details from the database. The following code is self-explained, so please go through the comment lines.

6: Adding a Reference to class Library Project in ASP.NET MVC Application:

Right-click on the "References" folder of the "MVC_DEMO" project and add a reference to the "BusinessLayer" class library project. Then Include a connection string with name = "DBCS"

7: Creating Controller

Right-click on the "Controllers" folder and add a Controller with the name "EmployeeController" and then copy and paste the following code into it. In the below controller we have only one action method i.e. Index. This method creates an instance of EmoloyeeBusinessLayer class and then calls the GetAllEmployees method which will return the list of employees. The list of employees is then handed over to the Index view.

8: Adding Index View

Right-click on the Index() action method in the "EmployeeController" class and then select "Add View" from the context menu. Set.

CODECHAMP
CREATED WITH ARBOK