book: ASP.NET 2.0 with C#&VB (WROX pub.)
Page No.
Date
ASP.NET 4.0 with C# & VB (WROX pub.) www.ASP.Net

2/1/19

ASP.NET 4.0 with C#

- ASP.NET: Active Server Pages (Server side technology)
- 4.0: version / framework
- C#: language

It is a microsoft technology
Not platform independent
Language independent (32 lang supported)
Uses Pascal coding (Java uses camel coding)

⇒ IDE - Visual Studio 15

⇒ Java enables creating : Console
& .NET Desktop / Stand alone
Web appln
Web services

⇒ C# file extension - .cs
Server - IIS

⇒ Server -

⇒ Java supports pure MVC.
One servlet can serve multiple JSP.

.aspx → UI / HTML
.aspx.cs → Logic / code

ASP.NET MVC 4-0
(may have dependency on .Net 4-0)
↳ V.S 15

Page No.
Date

```
using System;                          //import statement

namespace Mylone
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.Write("Hello World");
         Console.ReadKey();
      }
   }
}
```

In Java, a singer Servlet can serve multiple JSP's. But, in .Net, a single ASP will have corresponding Servlet.
default event: Page_Load

To access multiple ASP, session management needs to be used.

Debugging: Select code lines and press F10

Common life-cycle events

i) Init
ii) InitComplete
iii) PreLoad
iv) Load
v) Control events
vi) LoadComplete
vii) PreRender
viii) SaveStateComplete
ix) Render
x) Unload

⇒ PreInit is called to create or re-create controls. Creates control with default values.

⇒ Init is raised after all controls have been initialized and any skin settings have been modified.

⇒ InitComplete is raised by Page object. Used when certain tasks need to be completed after initialization.

⇒ PreLoad replaces the default value of Init method by the text entered by the user to preserve it over multiple requests. (Ex- text in a textbox) Saved in view state.

⇒ Load actually gives the controls to an element on the page. No control functions until and unless this method is not completed.

⇒ Any control which requires accessing or taking information from the database, Render method is used.
Ex- to load an dropdown list where its values are taken from the database, PreRender is used.

⇒ Unload event occurs for each control and then for the page (Page is the parent class)

⇒ Events which user can overwrite are

Init, PreRender, Unload,

7/19 TextBox properties : • TextMode (what data can be entered)
• ReadOnly (Editable or not)

⇒ Declare and assign a value to a variable before using it.

```
protected void ADD_Click(object sender, EventArgs
{                          ↑ ID of button
    int num1, num2 = 0;
    num1 = Convert.ToInt32(TextBox1.Text);
                           ↑ ID of textbox
                                  ↑ property
    :
    ResultTB.Text = (num1+num2).ToString();
                                ↑ method.
}
```

To avoid 'NULL' exception (or) format exception (input) 'if' loop or validation is req.

```
if (TextBox1.Text != "" && TextBox2.Text != "")
{
    :
}
else
{
    ResultTB.Text = "Please Enter Valid Nos.";
}
```

⇒ ~~Result.Text~~

⇒ Button4.Text = (num1+num2).ToString();
This button's value remains intact until the browser is not closed. This happens because we have modified its property which has no effect due to PageLoad as it is saved in view state. (view state tracks user activity)

TextBox event
protected void TextBox1_TextChanged(...)
{
  Response.Write("You have entered "+
                 TextBox1.Text);
}

But, when we run the above code in browser, the message is not displayed until a button is not clicked. This happens because button event forces the page to go to the server. The textbox change event is stored in view state but, is executed only when the page is sent back to server. For this, set PostBack property of textbox as true

⇒ static int i=0; // if not static, it is reinitialized each time
protected void Page_Load(...)
{
  if(!IsPostBack)
  {
    Response.Write("First Time");
  }

```
    else
    {
        i++;
        Response.Write("Next Request "+i);
    }
}
```

→ ResultTB.BackColor = System.Drawing.Color.Aqua

Or create an object of Color class and set its color (HEX code) and then pass the object here.

7/19 Create control at Runtime

```
Protected void Button1_Click(...)
{
    Button b = new Button();   // b becomes ID
    b.Text = "New Button";
    PlaceHolder1.Controls.Add(b);
}
```

Reference of child class can point to object of parent class.

→ Creating table at runtime

```
Table T = new Table();
int row = Convert.ToInt32(TextBox1.Text);
int col = Convert.ToInt32(TextBox2.Text);
for(int i=0; i < row; i++)
{
    TableRow tr = new TableRow();
```

⇒ PlaceHolder control is used to hold any control created at runtime.

```
for (int j=0 ; j<= col ; j++)
{
    TableCell tc = new TableCell();
    tc.Text = " "+i+" : "+j ;
    tr.Cells.Add(tc);
}
T.Rows.Add(tr);
}
PlaceHolder1.Controls.Add(T);   ← Horizontal
T.GridLines = GridLines.Both;     Vertical
                                  None.
```

⇒ To add buttons in each cell.

```
TableCell tc = new TableCell();
Button b1 = new Button();
b1.Text = " "+i+" : "+j;
tc.Controls.Add(b1);
tr.Cells.Add(tc);
```

⇒ Inside Page Load

```
if (!IsPostBack)
{ }
else
{
    PrintTable();
}
```

⇒

```
if (IsPostBack)
{
    PrintTable();
}
```

Table is printed twice.
i) PageLoad
ii) Button Click

Solution: maintain counter

⇒ To add event :

```
:
b1.Text = " " + i + ":" + j;
b1.Click += new EventHandler(b1_Click);
tc.Controls.Add(b1);
:

protected void b1_Click(object sender,
                        EventArgs e)
{
    Button b = sender as Button;
    Response.Write("You have clicked on" +
                   b.Text);
```

⇒ Control can only be added to an element having runat property as Server.
Here, Page.Controls.Add(T) does not work but, form1.Controls.Add(T) works.
(form1 ↑ name of WebForm.)

1/7/19 Validation Controls
ToolBox → Validation

Req. Field Validator
Properties: • ControlToValidate
(On which input control do we need validation)

• InitialValue

• Text (Message to be displayed) (Ex- * have to enter req. field)

- ErrorMessage (message to be displayed when the value is not entered)
  Ex- Please enter the number.

[Remove all text written inside < > tag in web-config file]

- EnableClientScript
  Page is not sent to the server until and unless all the validations are cleared; when this property is set true.

CompareValidator
used when we need to compare value in two input controls.
Ex- Password & Confirm Password.

Properties:
ControlToCompare (TextBox1 - Password)
ControlToValidate (Confirm password).

Operator : =, >=, <=, >, < ...

ValueToCompare (Any specific no., string etc. to be compared) ↑ compared ASCII val.

When ValueToCompare = ABC
B is accepted
AD is accepted.
} based on concept of DFA.

Multiple Validations can be applied on a single control.
But, one validation can control only one input control.

RangeValidator
- ControlToValidate
- Minimum Value } both are inclusive
- Maximum Value
- Type.

18
100
Integer

When min. : A
max. : D
E not accepted. [DD, DE]
BDE accepted.

RegularExpressionValidator
- ControlToValidate
- ValidationExpression
  Built-in: Email, ZIP code.

ValidationSummary
Displays all the error messages in form of a list / bullet list / paragraph

ValidationGroup
Used when we have multiple buttons for activity on same page. (Ex- login & sign up)
When we click on login, it does not show error messages for the fields in the sign up form fields & vice-versa

RequiredValidation (by default true)

⇒ Text property appears beside the textbox and error message in the summary. If text is "" then, by default it takes the error message string as text property too.

24/7/19 Calender control

Properties: TitleFormat (Month Year)
PrevMonthText (<) [<']

To know which date is clicked:
Response.Write("" + Calender1.SelectedDate.ToString());

For only date:	does not give time. Hence, 12:00:00 AM is displayed.
ToShortDateString()	

SelectionMode: Day (by default)
DayWeek (selects entire week)
DayWeekMonth (selects entire month)

```
for(int i=0;i<Calender1.SelectedDates.Count;i++)
{
    Response.Write("" + Calender1.SelectedDates[i].
                        ToShortDateString()+"<br>");
}
```

Only first: Calender1.SelectedDates[0]
Only last: Calender1.SelectedDates[Calender1.SelectedDates.Count-1]

Event is Calender1_SelectionChanged

⇒ calender does not have AutoPostBack property

ii) Calender1_DayRender
fired for every day of the month

for image button
EventArgs → DayRenderEventArgs
EventArgs → ImageButtonEventArgs

protected void Calender1_DayRender(object sender,) {
{ if(e.Day.Weekend) → cannot select weekend dates
{
e.Day.isSelectable = false;
}
}

⇒ if (e.Day.Date < DateTime.Now.Date)
{
e.Day.isSelectable = false; ↑ cannot select dates before today.
}

⇒ if (e.Day.DayNumberText == "25")
{
e.Cell.Text = TextBox1.Text;
}
↑ 25 in the cell is replaced with TextBox1 text.

⇒ if (e.Day.DayNumberText == "25")
{
LiteralControl lc = new LiteralControl();

Applies to the 25th date of every month
DayRender

```
lc.Text = TextBox1.Text;
e.Cell.Controls.Add(lc);
```

Appends text with

⇒ ~~e.Cell~~ lc.Text = " " + TextBox1.Text + " "
for printing text below 25

⇒ DayOfWeek gives the day (Monday, Tuesday, ...)
DayOfMonth gives the date
DayOfYear gives no (1 - 356)

⇒ DateTime d = new DateTime(2019, 8, 25);
Year Month Date
Calendar1.TodaysDate = d;
DateTime d1 = DateTime.Now;
Response.Write("Date Object is " + d1.ToLongDateString());
d.AddDays(5);
Response.Write(d1.ToLongDateString());
Adds temporarily
Hence, not printed.
d1.AddDays(5).ToLong.. works correctly
O/P: Wednesday 25-Jul-19

⇒ d1.ToUniversalTime()
prints time relative to GMT.

⇒ if(e.Day.Date.AddDays(5)
returns DateTime obj
Compare with today's / specified date & set IsSelectable T/F as req.

Applies to the 25th date of every month because DayRender

```
lc.Text = TextBox1.Text;
e.Cell.Controls.Add(lc);
```

Appends Add text with 25

=> e.Cell lc.Text = " " + TextBox1.Text + " ";
for printing text below 25

=> DayOfWeek gives the day (Monday, Tuesday, ...)
DayOfMonth gives the date
DayOfYear gives no (1-356)

=> DateTime d = new DateTime(2019, 8, 25);
Year month Date

```
Calendar1.TodaysDate = d;
DateTime d1 = DateTime.Now;
Response.Write("Date Object is " + d1.ToLongDateString());
d1.AddDays(5);
Response.Write(d1.ToLongDateString());
```

Adds temporarily. Hence, not printed.

d1.AddDays(5).ToLongDateString() works correctly

O/P: Wednesday 25-Jul-19

=> d1.ToUniversalTime()
prints time relative to GMT.

=> if(e.Day.Date.AddDays(5)
returns DateTime Obj
compare with today's / specified date & set as selectable T/F as req

⇒ TimeSpan t = d - d1;
Response.Write(t.TotalDays.ToString());
considers time as well
because both d & d1 are
DateTime Obj.

t.Days.ToString() returns an integer indicating days only

⇒ DateTime d = DateTime.UtcNow;
Calendar1.TodaysDate = d;
DateTime d1 = DateTime.Now;
TimeSpan t = d - d1;
Response.Write(t.Hours.ToString() + ":" +
t.Minutes.ToString() + ":" +
t.Seconds.ToString());

⇒ e.Day.Date < d1 && e.Day.Date >
d1.Day.Date.AddDays(1);

Static DateTime d1; ☹
↑ Else does not remember when page comes back to server.