Structure of an Asp.net page

The main components of an ASP.NET page are:

- Directives
- Code Declaration Blocks
- ASP.NET Controls
- Code Render Blocks
- Server-Side Comments
- Server-Side Include Directives
- Literal Text and HTML Tags.

Trace Directive

If you enable tracing for a page, additional information about the execution of a page is displayed alongwith the content of Page.

(Contd. to next Page)

Directives

starting of Directive → <%@ and %> ← ending of Directive

A directive controls how the page is compiled. It is marked by the tags, <%@ and %>. It can appear anywhere in a page. But, normally it is placed at the top of a page. The main types of directives are:

- Page
- Import

(i) A Page directive is used to specify the default programming language for a page. These are also used to enable tracing & debugging of a page.

```
<%@ Page Language="C#" %>
        OR
<%@ Language="C#" %>
```

(ii) Import Directive

Some namespaces are imported into an ASP.NET page by default. If you wish to use a class that is not contained in the default namespaces, you must import its namespace.

<%@ Import Namespace="System.Data.SqlClient" %>

Trace Directive Contd.

- To enable Tracing for a page :- <%@ Page Trace = "True" %>

After enabling Tracing for a page, you can display Trace messages by using 2 methods of Trace class
- Write() → It displays messages in black Text
- Warn() → It displays messages in red Text.

Trace information is always displayed at the bottom of the page

- Debug Directive → It is used to enable runtime error messages to be displayed on a page.

Syntax :- <%@ Page Debug = "True" %>

→ To enable both the Directives simultaneously, we will combine the directives like :- <%@ Page Debug = "True" Trace = "True" %>

The ASP.NET page in Listing 11 uses these methods to display various trace messages.

Listing 11—Trace.aspx

```
<%@ Page Trace="True" %>
<Script Runat="Server">

Sub Page_Load
 Dim strTraceMessage As String

 Trace.Warn( "Page_Load event executing!" )
 strTraceMessage = "Hello World!"
 Trace.Write( "The value of strTraceMessage is " & strTrace
End Sub

</Script>

<html>
<head><title>Trace.aspx</title></head>
<body>

<h2>Testing Page Trace</h2>

<% Trace.Warn( "Rendering page content!" ) %>

</body>
</html>
```

For example, suppose that you want to send an email from an ASP.NET page by using the Send method of the `SmtpMail` class. The `SmtpMail` class is contained in the `System.Web.Mail` namespace. This is not one of the default namespaces imported into an ASP.NET page.

The easiest way to use the `SmtpMail` class is to add an Import directive to your ASP.NET page to import the necessary namespace. The page in Listing 12 illustrates how to import the `System.Web.Mail` namespace and send an email message.

Listing 12—ImportNamespace.aspx

```
<%@ Import Namespace="System.Web.Mail" %>

<Script Runat="Server">

Sub Page_Load
 Dim mailMessage As SmtpMail

 mailMessage.Send( _
  "bob@somewhere.com", _
  "alice@somewhere.com", _
  "Sending Mail!", _
  "Hello!" )
End Sub

</Script>

<html>
<head><title>ImportNamespace.aspx</title></head>
<body>

<h2>Email Sent!</h2>
```

```
<h2>Email Sent!</h2>


</body>
</html>
```

The first line in Listing 12 contains an import directive. Without the import directive, the page would generate an error because it would not be able to resolve the `SmtpMail` class.

Instead of importing the `System.Web.Mail` namespace with the import directive, you could alternatively use its fully qualified name. In that case, you would declare an instance of the class like this:

```
Dim mailMessage As System.Web.Mail.SmtpMail
```

Code Declaration Blocks

A code declaration block contains all the application logic for a page. It also includes declarations of global variables, and functions. It must be written within the < script runat= "server" > tag. The <script> tag has two optional parameters:

(i) • Language: You can specify the programming language to be used within the <script> tag. Otherwise, the language specified in the Page directive is

```
<script runat="server">
        protected void Page_load(object sender, EventArgs e)
        {
                // Statements;
        }
        protected void btnAdd_Click(object sender, EventArgs e)
        {
                // Statements;
        }
</script>
```

used.

(ii) • SRC: You can specify an external file that contains the code block.

SRC: You can specify an external file that contains the code block. There is no difference in output between writing the code on the same page and including an external script file.

eg. from SS

Example 3.6: Using an External File

```
<script runat="server" src="externalfile.aspx" />
<html>
        <head> <title> Including an External File </title> </head>
        <form runat="server">
                <center>
                        <asp:label id="lblMsg" runat="server" />
                        <p>
                        <asp:button id="btnSubmit" text="click"
onclick="btnSubmit_click"
                        runat="server" />
                </center>
        </form>
</html>

//The external file "externalfile.aspx" is shown below:
protected void btnSubmit_click(Source As Object, E As EventArgs)
{
        lblMsg.text="Hello World!";
}
```

The <Script Runat="Server"> tag accepts two optional attributes. First, you can specify the programming language to use within the <Script> tag by including a language attribute like this:

```
<Script Language="C#" Runat="Server">
```

If you don't specify a language, the language defaults to the one defined by the <%@ Page Language %> directive mentioned in the previous section. If no language is specified in a page, the language defaults to Visual Basic.

The second optional attribute of the <Script Runat="Server"> tag is SRC. You can use it to specify an external file that contains the contents of the code block. This is illustrated in Listings 13 and 14.

Listing 13—ExternalFile.aspx

```
<Script Runat="Server" SRC="ApplicationLogic.aspx"/>

<html>
<head><title>ExternalFile.aspx</title></head>
<body>

<form Runat="Server">

<asp:label id="lblMessage" Runat="Server"/>

<asp:Button
 Text="Click Here!"
 OnClick="Button_Click"
 Runat="Server"/>

</form>

</body>
</html>
```

Listing 14—ApplicationLogic.aspx

```
Sub Button_Click( s As Object, e As EventArgs )
 lblMessage.Text = "Hello World!"
End Sub
```

ASP.NET Controls

ASP.NET controls can be mixed with text and static HTML in a page. All controls must appear within a <form runat= "server"> tag. Some controls

such as and the Label control can appear outside this tag. You can have only one form per page in ASP.NET.

Like <ASP: Label runat = "server" />

→ It's the limitation of ASP.Net Page

Code Render Blocks

If you wish to execute code within HTML, you can include the code within code render blocks. There are two types of code render blocks:

(i) • Inline Code: It executes a statement or series of statements. It is marked by the characters <% and %>.

(ii) • Inline Expressions: They display the value of a variable or method. They can be considered as shorthand notation for the Response.Write method. They are marked by the characters <%= and %>.

The ASP.NET page in Listing 15 illustrates how to use both inline code and inline expressions in an ASP.NET page.

Listing 15—CodeRender.aspx

```
<Script Runat="Server">
 Dim strSomeText As String

 Sub Page_Load
  strSomeText = "Hello!"
 End Sub
</Script>

<html>
<head><title>CodeRender.aspx</title></head>
<body>

<form Runat="Server">

The value of strSomeText is:
<%=strSomeText%>

<p>

<% strSomeText = "Goodbye!" %>
The value of strSomeText is:
<%=strSomeText%>

</form>

</body>
</html>
```

FirstPage.aspx → Page Directive

```
<%@ Page Language="C#" %>
<script runat="server">
    Void Page_load()
    {
        lblServerTime.Text = DateTime.Now.ToString()
    }
</script>
<head>
<title>First Page</title></head>
<body>
<form id="form1" runat="server">
<div>
Welcome to ASP.NET 4.0! The current date and time is:
<asp:Label
id="lblServerTime"
Runat="Server" />
</div>
</form>
</body>
</html>
```

→ Code Declaration block

This method is an eg of event handler. It handles the Pageload event. Every time the page loads, the method automatically executes & assigns the current date and time to label control.

→ Method - This method is assigning the current date and time to Text Property of label control named lblServerTime

All Methods, Fxns & sub routines are inside this block.

→ Eg of ASP.Net control. This tag represents an ASP.NET control that executes on the server.

→ Label control is declared with this tag: <asp:Label>

Code Render Block (It is always b/w the opening & closing <html> tags)

, Main things are there in the Code Render Block [Form tag, label control

Server-Side Comments

You can add comments in server-side code using the characters <%-- and --%>. The main use of these comment blocks is to add documentation to a page. These are useful when you are debugging an ASP.NET Page. The code enclosed within these comments can be removed temporarily.

Listing 16—ServerComments.aspx

```
<Script Runat="Server">
 Dim strSomeText As String
 Sub Page_Load
  strSomeText = "Hello!"
 End Sub
</Script>

<html>
<head><title>ServerComments.aspx</title></head>
<body>

<form Runat="Server">

<%--
This is inside the comments
<asp:Label Text="hello!" Runat="Server" />
<%= strSomeText %>
--%>

This is outside the comments

</form>

</body>
</html>
```

Server-Side Include Directives

You can include a file in an ASP.NET page by using a server-side include directive. It is executed before any of the code in the page. If the file is in the same directory or in a sub-directory of the page including the file, this directive is written as:

```
<!--  #INCLUDE  file="includedfile.aspx"  -->
```

You can also specify the virtual path of the file. To include a file located in the directory MyAspx under the wwwroot directory, you will write the directive as:

```
<!--  #INCLUDE  virtual="/MyAspx/includedfile.aspx"  -->
```

Note: It is recommended that you avoid using server-side include directives. It is better to use user controls.

From SS also

HTML Tags and Literal Text

You can build the static part of an ASP.NET page using HTML tags and literal text. The HTML content of your page is also compiled along with the rest of the contents.
The literal text has been made bold and converted to uppercase before being rendered in the browser.

from SS also

Example 3.9: LiteralControl Class

Listing 17—Literal.aspx

```
<Script Runat="Server">

 Sub Page_Load
  Dim litControl As LiteralControl

  For each litControl in Page.Controls
   litControl.Text = strReverse( litControl.Text )
  Next
 End Sub

</Script>

<html>
<head><title>Literal.aspx</title></head>
<body>

<b>This text is reversed</b>

</body>
</html>
```