**ASP.NET Directives**

ASP.NET directives are instructions to specify optional settings, such as registering a custom control and page language. These settings describe how the web forms (.aspx) or user controls (.ascx) pages are processed by the .Net framework.

The syntax for declaring a directive is:

<%@ directive\_name attribute=value [attribute=value] %>

1. **The Application Directive**

The Application directive defines application-specific attributes. It is provided at the top of the global.aspx file.

The basic syntax of Application directive is:

<%@ Application Language="C#" %>

1. **The Control Directive**

The control directive is used with the user controls and appears in the user control (.ascx) files.

The basic syntax of Control directive is:

<%@ Control Language="C#" EnableViewState="false" %>

1. **The Implements Directive**

The Implement directive indicates that the web page, master page or user control page must implement the specified .Net framework interface.

The basic syntax for implements directive is:

<%@ Implements Interface="interface\_name" %>

1. **The Import Directive**

The Import directive imports a namespace into a web page, user control page of application. If the Import directive is specified in the global.asax file, then it is applied to the entire application. If it is in a page of user control page, then it is applied to that page or control.

The basic syntax for import directive is:

<%@ namespace="System.Drawing" %>

1. **The Master Directive**

The Master directive specifies a page file as being the mater page.

The basic syntax of sample MasterPage directive is:

<%@ MasterPage Language="C#" AutoEventWireup="true" CodeFile="SiteMater.master.cs" Inherits="SiteMaster" %>

1. **The OutputCache Directive**

The OutputCache directive controls the output caching policies of a web page or a user control.

The basic syntax of OutputCache directive is:

<%@ OutputCache Duration="15" VaryByParam="None" %>

1. **The Page Directive**

The Page directive defines the attributes specific to the page file for the page parser and the compiler.

The basic syntax of Page directive is:

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" Trace="true" %>

The attributes of the Page directive are:

1. **The PreviousPageType Directive**

The PreviousPageType directive assigns a class to a page, so that the page is strongly typed.

The basic syntax for a sample PreviousPagetype directive is:

<%@ PreviousPageType attribute="value"[attribute="value" ...] %>

1. **The Reference Directive**

The Reference directive indicates that another page or user control should be compiled and linked to the current page.

The basic syntax of Reference directive is:

<%@ Reference Page ="somepage.aspx" %>

1. **The Register Directive**

The Register derivative is used for registering the custom server controls and user controls.

The basic syntax of Register directive is:

<%@ Register Src="~/footer.ascx" TagName="footer" TagPrefix="Tfooter" %>

**Code Declaration blocks**

Defines member variables and methods compiled in the dynamically generated [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page(v=vs.71).aspx) or [UserControl](https://msdn.microsoft.com/en-us/library/system.web.ui.usercontrol(v=vs.71).aspx) classes that represent the ASP.NET page and the user control, respectively.

<script language="C#" runat="server">

void EnterBtn\_Click(Object Src, EventArgs E) {

Abc.Text = "Hi " + Name.Text + ", welcome to ASP.NET!";

}

</script>

<form id="Form2" runat="server">

Enter your name: <asp:textbox id="Name" runat=server/>

<asp:button ID="Button1" text="Enter" Onclick="EnterBtn\_Click" runat="server"/>

<p>

<asp:label id="Abc" runat=server/>

</form>

**Attributes**

1. **language**

Specifies the language used in this code declaration block. This value can represent any .NET-compatible language, such as Visual Basic (VB), C#, or JScript .NET. If no language is specified, this value defaults to that specified in the **@ Page** or **@ Control** directive. If no language is specified in the directive, the default is **VB**, unless you have changed the default in your application's Web.config file.

1. **src**

Specifies the path and file name of a script file to load. When this attribute is used, any other code in the declaration block is ignored.

**Code Render Blocks**

Defines inline code or inline expressions that execute when the page is rendered. There are two styles:

**1. Inline code:** Use inline code to define self-contained code blocks or control flow blocks.

<% inline code %>

1. **Inline expressions:** Use inline expressions as a shortcut for calling the [HttpResponse.Write](https://msdn.microsoft.com/en-us/library/system.web.httpresponse.write(v=vs.71).aspx) method.

<%=inline expression %>

#### Remarks

A compilation error occurs if you try to include the character sequence **%>** anywhere inside a code render block. That sequence can only be used to close the code render block. For example, the following code fragment will cause an error in C#:

<%@ page language="C#" %>

<%

Response.Write(" %>");

%>

[Visual Basic]

<%@ page language="VB" %>

<%

Response.Write("%>)

%>

# ASP.NET - Server Controls

Controls are small building blocks of the graphical user interface, which include text boxes, buttons, check boxes, list boxes, labels, and numerous other tools. Using these tools, the 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, and 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 are the primary controls used in ASP.NET. These controls can be grouped into the following categories:

* **Validation controls** - These are used to validate user input and they 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 displaying.
* **Personalization controls** - These are used for personalization of a page according to the user preferences, based on user information.
* **Login and security controls** - These controls provide user authentication.
* **Master pages** - These controls provide consistent layout and interface throughout the application.
* **Navigation controls** - These controls help in navigation. For example, menus, tree view etc.
* **Rich controls** - These controls implement special features. For example, AdRotator, FileUpload, and Calendar control.

The syntax for using server controls is:

<asp:controlType ID ="ControlID" runat="server" Property1=value1 [Property2=value2] />

# Server-Side Comments

Server-side comments allow developers to embed code comments in any part of an ASP.NET application file (except for within **<script>** code blocks). Any content between opening and closing tags of server-side comment elements, whether ASP.NET code or literal text, will not be processed on the server or rendered to the resulting page.

<%-- Content of comments, or commented out server controls --%>

## Example

The following code example demonstrates a [Button](https://msdn.microsoft.com/en-in/library/system.web.ui.webcontrols.button(v=vs.85).aspx) control that is commented out using server-side comments.

<%--

<asp:button runat="server" id="MyButton"

OnClick="MyButton\_Click" />

--%>

**Literal Text**

Text property is used to specify or determine the caption displayed in the [Literal](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.literal(v=vs.110).aspx) control. This allows you to programmatically change the caption at run time.

Syntax in C#

[BindableAttribute(true)]

public string Text { get; set; }

The caption is HTML-decoded depending on how you assign a value to this property. If you assign a value through an attribute of the [Literal](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.literal(v=vs.110).aspx)control, the value is HTML decoded before it is displayed. For example, <asp:Literal id="DisplayLiteral" Text="A&nbsp;B"/> is rendered as "A B" on the browser. However, if you set this property programmatically or by placing the text between the opening and closing tags of the control, the caption is not HTML decoded. For example, <asp:Literal id="DisplayLiteral"> A&nbsp;B </asp:Literal> is rendered as "A&nbsp;B".

**ASP.NET State View**

View state is the method that the ASP.NET page framework uses to preserve page and control values between round trips. When the HTML markup for the page is rendered, the current state of the page and values that must be retained during postback are serialized into base64-encoded strings. This information is then put into the view state hidden field or fields.

This topic contains the following sections:

## Scenarios

View state is used automatically by the ASP.NET page framework to persist information that must be preserved between postbacks. This information includes any non-default values of controls.

You can also use view state to store application data that is specific to a page.

## Features

View state is a repository in an ASP.NET page that can store values that have to be retained during postback. The page framework uses view state to persist control settings between postbacks.

You can use view state in your own applications to do the following:

* Keep values between postbacks without storing them in session state or in a user profile.
* Store the values of page or control properties that you define.
* Create a custom view state provider that lets you store view state information in a SQL Server database or in another data store.

For example, you can store information in view state that your code can access during the page load event the next time that the page is sent to the server.

1. **Background**

A Web application is stateless. A new instance of the Web page class is created every time that the page is requested from the server. This would ordinarily mean that all information in the page and in its controls would be lost with each round trip. For example, by default if a user enters information into a text box on an HTML Web page, that information is sent to the server. However, it is not returned to the browser in the response.

To overcome this intrinsic limitation of Web programming, the ASP.NET page framework includes several state-management features to preserve page and control values between round trips to the Web server. One of these features is view state.

#### Data Types You Can Store in View State

You can store objects of the following types in view state:

* Strings
* Integers
* **Boolean** values
* **Array** objects
* **ArrayList** objects
* Hash tables
* Custom type converters

1. **Class Reference**

|  |  |
| --- | --- |
| [ViewState](https://msdn.microsoft.com/en-us/library/system.web.ui.control.viewstate.aspx) | Provides a dictionary object for retaining values between requests for the same page. |
| [PageStatePersister](https://msdn.microsoft.com/en-us/library/system.web.ui.pagestatepersister.aspx) | Provides a means to define a custom mechanism for storing view state information, such as in a SQL Server database. |