

ASP.NET Master Page & State Management

These notes focus on ASP.NET Web Forms with VB.NET, which is important because Master Pages, Themes, Cookies, Session State, and Application State are commonly used in ASP.NET Web Forms.

1. ASP.NET Master Page — Overview

An ASP.NET Master Page is a template that provides a common layout and design for multiple web pages in an ASP.NET Web Forms application.

A Master Page usually contains common elements such as:

Header

Logo

Navigation menu

Sidebar

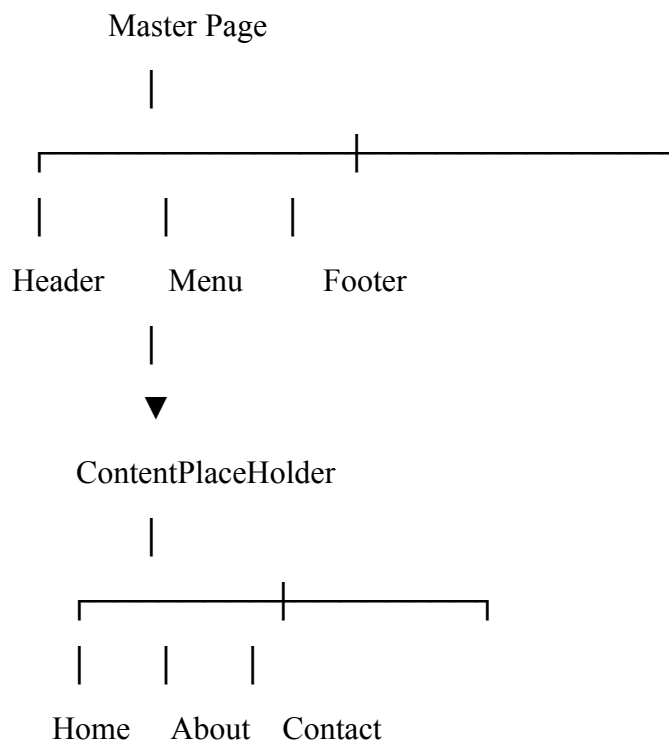
Footer

Common CSS

Common controls

The individual pages are called Content Pages.

Basic Structure



The Master Page defines the common layout, while each Content Page supplies its own content.

1.1 Why Do We Need Master Pages?

Suppose a college website has 20 pages.

Without a Master Page, we might have to write the following on every page:

College Logo

Navigation Menu

Header

Footer

CSS

This creates duplicate code.

With a Master Page, we create these elements once.

Site.master

↓

Common Header

Common Menu

Common Footer

↓

Content Pages

Advantages

Code Reusability

Common layout is written only once.

Consistency

All pages have the same design.

Easy Maintenance

Changing the Master Page can update the layout of associated pages.

Separation of Content and Presentation

Common design is separated from page-specific content.

Reduced Development Time

Developers don't need to create the same layout repeatedly.

2. Creating a Master Page

In an ASP.NET Web Forms application, create a Master Page:

Site.master

The Master Page uses the .master extension.

Basic Master Page

```
<%@ Master Language="VB" %>

<!DOCTYPE html>

<html>

<head runat="server">

    <title>ABC College</title>

</head>

<body>

<form id="form1" runat="server">

    <div>

        <1>ABC College</h1>

        <hr />

        Home | About | Courses | Contact

        <hr />

        <asp:ContentPlaceHolder      ID="MainContent"      runat="server">

            </asp:ContentPlaceHolder>

        <hr />

        <p>Copyright © 2026 ABC College</p>

    </div>

</form>

</body>

</html>
```

Here:

```
<%@ Master Language="VB" %>
```

indicates that the Master Page uses VB.NET.

3. Master Page Layout with CSS

CSS stands for:

Cascading Style Sheets

CSS is used to control the appearance of the Master Page.

For example, we can use CSS to control:

Background color

Font

Text alignment

Width and height

Margins

Padding

Borders

Navigation menu

Header and footer

3.1 External CSS File

Create:

StyleSheet.css

In the Master Page:

HTML

```
<link href="StyleSheet.css" rel="stylesheet" />
```

Meaning

Part

Meaning

link

Connects another resource to HTML

href

Specifies the location of the CSS file

rel

Relationship

stylesheet

Indicates that the linked file is a stylesheet

So:

HTML

rel="stylesheet"

means:

The linked resource is a CSS stylesheet.

3.2 CSS Example

StyleSheet.css

```
body {  
    margin: 0;  
    font-family: Arial;  
}
```

```
.header {  
    background-color: navy;  
    color: white;  
    padding: 20px;  
    text-align: center;  
}
```

```
.menu {  
    background-color: lightgray;  
    padding: 15px;  
    text-align: center;  
}
```

```
.content {  
    padding: 30px;  
    min-height: 300px;
```

```
}
```

```
.footer {  
    background-color: black;  
    color: white;  
    padding: 15px;  
    text-align: center;  
}
```

Using CSS in Master Page

```
<div class="header">  
    <h1>ABC College</h1>  
</div>
```

```
<div class="menu">  
    Home | About | Courses | Contact  
</div>
```

```
<div class="content">  
  
    <asp:ContentPlaceHolder  
        ID="MainContent"  
        runat="server">  
  
    </asp:ContentPlaceHolder>  
</div>
```

```
<div class="footer">  
    Copyright © 2026 ABC College  
</div>
```

The class attribute connects the HTML element with the CSS rule.

4. Master Page Directive

The Master Page Directive specifies that a file is an ASP.NET Master Page.

Syntax

```
<%@ Master Language="VB" %>
```

A more complete example is:

```
<%@ Master  
    Language="VB"  
    AutoEventWireup="true"  
    CodeFile="Site.master.vb"  
    Inherits="Site" %>
```

Important Properties

Property

Meaning

Language

Specifies programming language

AutoEventWireup

Automatically associates page events

CodeFile

Specifies code-behind file

Inherits

Specifies the class inherited by the Master Page

5. ContentPlaceHolder

A ContentPlaceHolder defines an area in the Master Page where the content of a Content Page will be displayed.

Example

```
<asp:ContentPlaceHolder  
    ID="MainContent"  
    runat="server">  
</asp:ContentPlaceHolder>
```

Here:

asp:ContentPlaceHolder → ASP.NET server control

ID="MainContent" → Unique identifier

runat="server" → Makes the control available on the server

6. Content Page

A Content Page is an ASP.NET page that uses a Master Page.

Suppose we have:

Site.master

Home.aspx

About.aspx

Contact.aspx

Home.aspx can use Site.master.

Home.aspx

<%@ Page

Language="VB"

MasterPageFile="~/Site.master"

AutoEventWireup="true"

CodeFile="Home.aspx.vb"

Inherits="_Default" %>

<asp:Content ID="Content1"

ContentPlaceHolderID="MainContent" runat="server">

<h2>Welcome to ABC College</h2>

<p>

This is the Home Page.

</p>

</asp:Content>

Important Properties

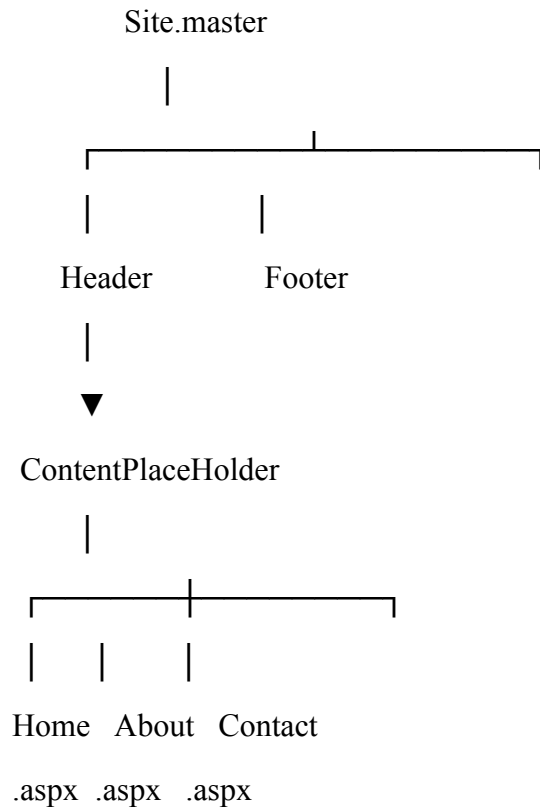
MasterPageFile="~/Site.master"

specifies the Master Page.

ContentPlaceHolderID="MainContent"

specifies which ContentPlaceHolder will contain this content.

7. Master Page and Content Page Relationship



The Master Page controls the common structure.

The Content Page controls the page-specific content.

8. Accessing Master Page from Content Page

Sometimes a Content Page needs to access controls or properties of the Master Page.

Master Page

Suppose the Master Page contains:

```
<asp:Label
    ID="lblCollege"
    runat="server"
    Text="ABC College">
</asp:Label>
```

In the Content Page's VB.NET code:

```
Dim master As Site = CType(Me.Master, Site)
```

```
master.lblCollege.Text = "XYZ College"
```

This allows the Content Page to communicate with the Master Page.

9. ASP.NET Themes

An ASP.NET Theme is a collection of files used to define the visual appearance of an ASP.NET application.

Themes can contain:

CSS files

Skin files

Images

Other resources

Themes help maintain a consistent appearance throughout an application.

10. Creating a Theme

Create a special folder:

App_Themes

Inside it create:

CollegeTheme

Structure:

App_Themes

```
|
└── CollegeTheme
    |
    ├── Style.css
    └── Controls.skin
```

11. CSS in a Theme

Style.css

```
body {
    font-family: Arial;
    background-color: #f2f2f2;
}
```

```
h1 {  
    color: navy;  
}  
  
p {  
    font-size: 16px;  
}
```

The CSS controls the visual appearance of the pages.

12. Skin File

A Skin file is used to define the appearance of ASP.NET server controls.

Create:

Controls.skin

Example:

```
<asp:Button  
    runat="server"  
    BackColor="Blue"  
    ForeColor="White"  
    Font-Bold="true" />
```

Now ASP.NET Button controls can automatically receive this appearance when the theme is applied.

13. Applying a Theme

A theme can be applied at two levels:

1. Page Level

```
<%@ Page  
    Language="VB"  
    Theme="CollegeTheme" %>
```

This applies the theme to that particular page.

2. Application Level

A theme can be specified in Web.config.

```
<configuration>
  <system.web>
    <pages theme="CollegeTheme" />
  </system.web>
</configuration>
```

This applies the theme throughout the application.

14. Themes vs CSS

CSS

Theme

Mainly controls visual styling

Provides overall ASP.NET appearance

Uses .css files

Can include CSS and .skin files

Works with HTML elements

Can style ASP.NET server controls

Client-side styling

ASP.NET framework feature

15. Cookies

A cookie is a small amount of information stored on the user's browser/client computer.

Cookies are useful for remembering information between requests.

Examples

Username

User preferences

Language preference

Shopping cart information

User identification information

16. Creating a Cookie Using VB.NET

```
Dim cookie As New HttpCookie("UserName")
```

```
cookie.Value = "Rahul"
```

```
Response.Cookies.Add(cookie)
```

Here:

```
"UserName"
```

is the cookie name.

```
"Rahul"
```

is the cookie value.

17. Reading a Cookie

If Request.Cookies("UserName") IsNot Nothing Then

```
Dim name As String =
```

```
Request.Cookies("UserName").Value
```

End If

18. Setting Cookie Expiration

```
Dim cookie As New HttpCookie("UserName")
```

```
cookie.Value = "Rahul"
```

```
cookie.Expires = DateTime.Now.AddDays(7)
```

```
Response.Cookies.Add(cookie)
```

The cookie is configured to expire after seven days.

19. Types of Cookies

19.1 Session Cookie

A temporary cookie that generally lasts for the browser session.

19.2 Persistent Cookie

A cookie that remains until its specified expiration date.

Example:

```
cookie.Expires = DateTime.Now.AddDays(30)
```

20. Advantages of Cookies

Easy to implement.

Stored on the client.

Can remember user preferences.

Useful for personalization.

Can reduce repeated data entry.

Disadvantages

Users can disable cookies.

Storage capacity is limited.

Users can potentially inspect/modify client-side data.

Sensitive information should not be stored directly in cookies.

21. ASP.NET Session State

Session State is used to store information associated with a particular user during a session.

Session data is maintained on the server in the normal ASP.NET Session State model.

For example:

User Login

↓

Session Created

↓

User visits Page 1

↓

User visits Page 2

↓

Session information remains available

22. Creating Session Variables in VB.NET

```
Session("UserName") = "Rahul"
```

```
Session("Course") = "MCA"
```

Here:

UserName → Rahul

Course → MCA

23. Reading Session Variables

If Session("UserName") IsNot Nothing Then

Dim name As String =

Session("UserName").ToString()

End If

24. Removing Session Data

To remove one Session variable:

Session.Remove("UserName")

To remove all Session variables:

Session.Clear()

To terminate the entire session:

Session.Abandon()

Difference

Method

Purpose

Session.Remove()

Removes one variable

Session.Clear()

Removes all session variables

Session.Abandon()

Ends the session

25. Session Timeout

A session remains available for a configured period of inactivity.

The timeout can be configured in Web.config.

<configuration>

<system.web>

```
<sessionState timeout="20" />
</system.web>
</configuration>
```

Here:

timeout="20"

means the session timeout is 20 minutes of inactivity.

26. Session State Modes

ASP.NET supports different ways of storing session data.

26.1 InProc

Session information is stored in the memory of the web application process.

```
<sessionState mode="InProc" />
```

Advantages

Fast

Simple

Disadvantage

Session data can be lost if the application process restarts.

26.2 StateServer

Session data is stored in the ASP.NET State Service.

```
<sessionState mode="StateServer" />
```

It can be useful when session information should be kept outside the web application's process.

26.3 SQLServer

Session data is stored in SQL Server.

```
<sessionState mode="SQLServer" />
```

This can be useful in environments where session state needs to be maintained in a database.

27. Application State

Application State stores information that is shared by all users of an ASP.NET application.

Important Difference

Session State



Individual User

Application State



All Users

For example, an application can maintain a common visitor count.

28. Creating Application State Using VB.NET

```
Application("Visitors") = 5000
```

Here:

```
Visitors = 5000
```

is available to all users of the application.

29. Reading Application State

```
Dim count As Integer =
```

```
    CInt(Application("Visitors"))
```

30. Application State and Locking

Application State is shared among multiple users.

Suppose two users try to update the same value at the same time.

ASP.NET provides:

```
Application.Lock()
```

and

```
Application.Unlock()
```

Example

```
Application.Lock()
```

```
Application("Visitors") =
```

```
    CInt(Application("Visitors")) + 1
```

```
Application.Unlock()
```

Working

Application.Lock()

↓

Modify shared data

↓

Application.Unlock()

Locking helps prevent conflicting simultaneous updates to shared Application State.

31. Cookie vs Session vs Application State

Feature

Cookie

Session State

Application State

Main location

Client/browser

Server

Server

Scope

Browser/user

Individual user

Entire application

Shared between users

No

No

Yes

Example

Remember preference

Login information

Visitor count

Controlled by

Browser/client

ASP.NET application

ASP.NET application

Typical purpose

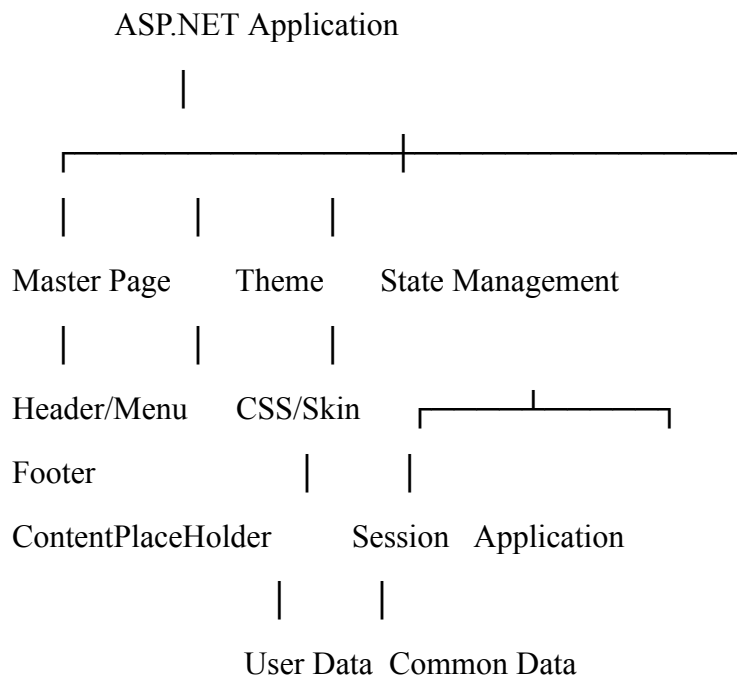
Small client-side data

User-specific data

Common application data

32. Master Page + Theme + Session Example

A typical college application might work like this:



Example

A college website may use:

Master Page

Header

Navigation

Footer

Theme

College colors

Button styles

Fonts

Session

Logged-in student's name

Course

User ID

Application

Total visitors

College name

Common settings

33. Complete VB.NET Example — Session

ASPX

```
<asp:Button  
    ID="btnLogin"  
    runat="server"  
    Text="Login"  
    OnClick="btnLogin_Click" />
```

```
<asp:Label  
    ID="lblMessage"  
    runat="server">
```

```
</asp:Label>
```

VB.NET Code-Behind

```
Protected Sub btnLogin_Click( sender As Object, e As EventArgs)  
    Session("UserName") = "Rahul"  
    lblMessage.Text =  
        "Welcome " & Session("UserName").ToString()
```

```
End Sub
```

34. Complete VB.NET Example — Application State

```
Protected Sub Page_Load( sender As Object, e As EventArgs)
```

If Application("Visitors") Is Nothing Then

Application("Visitors") = 1

Else

Application.Lock()

Application("Visitors") =

CInt(Application("Visitors")) + 1

Application.Unlock()

End If

lblVisitors.Text =

"Visitors: " &

Application("Visitors").ToString()

End Sub

35. Important Differences for Examination

Master Page vs Theme

Master Page controls the structure/layout.

Theme controls the appearance/style.

Master Page → Layout

Theme → Appearance

Session vs Application

Session State is specific to one user.

Application State is shared by all users.

Session

↓

User A → own data

User B → own data

Application

↓

User A —

User B —+→ Shared data

User C —┐