Asp.Net

Ch. 04

Master Pages And Theme Caching, Application Pages AND Data

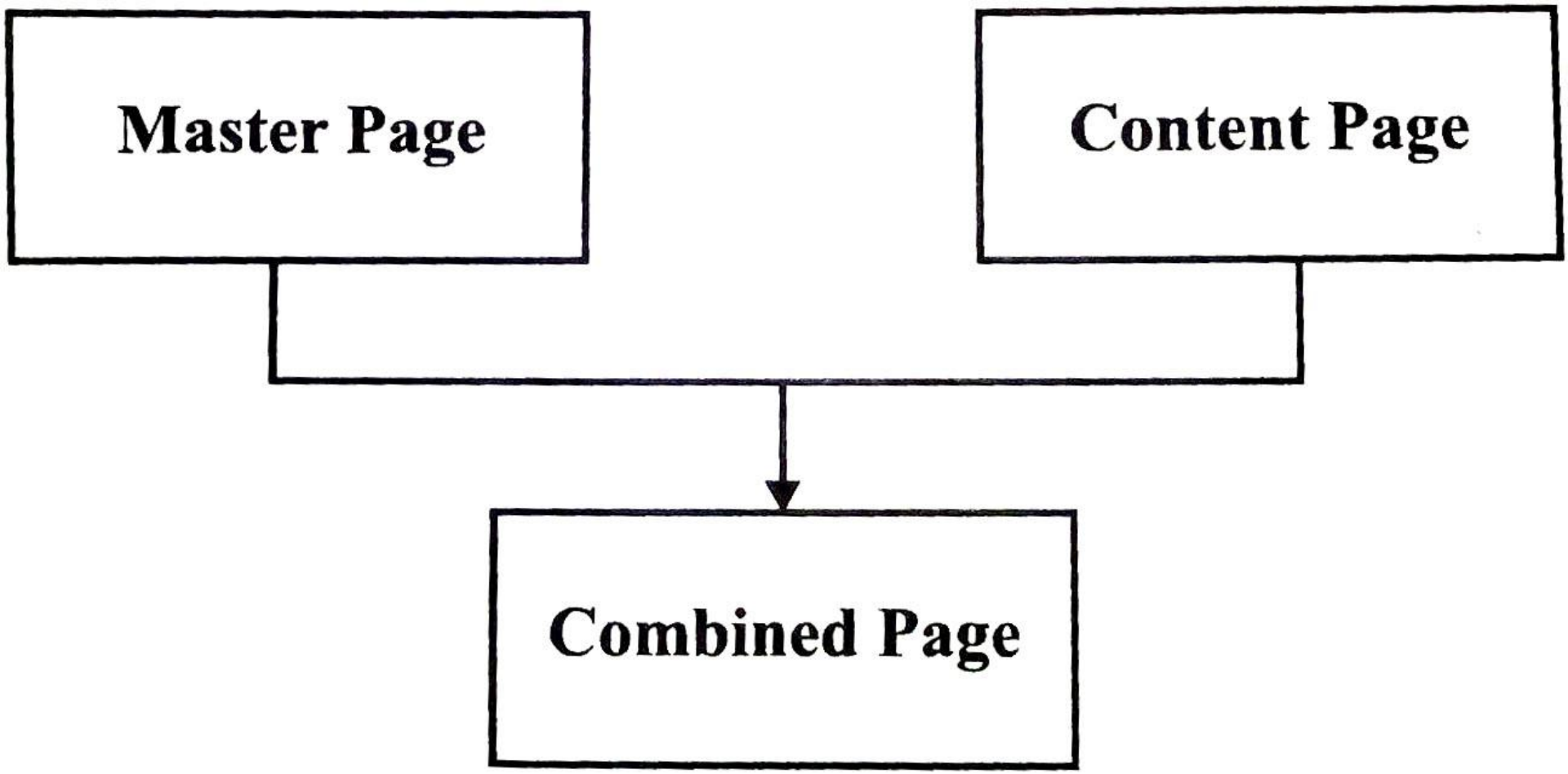
What is Master Page ?

- A Master Page enables you to share the same content among multiple content pages in a website.
- We can use a Master Page to create a common page layout.
- For example,
  - If we want all the pages in our website to share a three-column layout, we can create the layout once in a Master Page and apply the layout to multiple content pages.

Understainding Master Page

- First we need to identify the controls that we need to display on all pages and then add these controls to the master page.
- Then we need to create a ContentPlaceHolder control for the master page to place the content of the Web Page.
- When we execute this website, then the layout of master page and the content in ContentPlaceHolder control are merged to display the out of a web page.

Understainding Master Page



Requirement of Master Page :

- Master page provides centralize common functionality of various pages used in Web Application such that if we make change in one place it would be updated everywhere.
- It makes it easy to create a set of controls and code which applied to a set of a pages.
- For example, We can create a common menu for all pages.

Requirement of Master Page :

- The Master Page Object models allows us to make changes to the master page from individual content page which in turn allows us to have fine grained control over the rendering of the final page.
- Since most of UI code is in the Master Page, the size of content pages will be very small.

Creating Master Page...

- We can create a Master Page by creating a file that ends with the **.master** extension.
- Master page has a predefined layout that includes,
  - Static text
  - HTML elements and
  - Server Controls
- The Master page is identified by a special **@Master** directive. In ordinary **.aspx** page we have **@page** directive.

Creating Master Page…

- The @Master directive looks like the following :

```
<%@Master Language="C#"
  AutoEventWireup="true"
  CodeFile="MasterPage.master.cs"
  Inerits="MasterPage" %>
```

- The above property indicates :
  - Language indicates the language in which the page is written.

Def. Create a Master Page and try to run it.

- Step-1
  - Create a new WEBSite.
- Step-2
  - Add New Master Page.
- Step-3
  - Design the master page with required controls.
- Try to Execute the PROGRAM…!

Creating Content Page

- The content for the Master Page's PlaceHolder controls are created by placing individual content pages, which are ASP.NET pages.
- These page are bounded to a specific master page. The binding is established by using @Page directive of content page.

```
<%@Page Language="C#"
  MasterPageFile="~/MasterPage.Master"
  AutoEventWireup="True"
  CodeFile="Default.aspx.cs"
  Inherits="_Default" %>
```

Def. Create a Content PAGE of a Master Page.

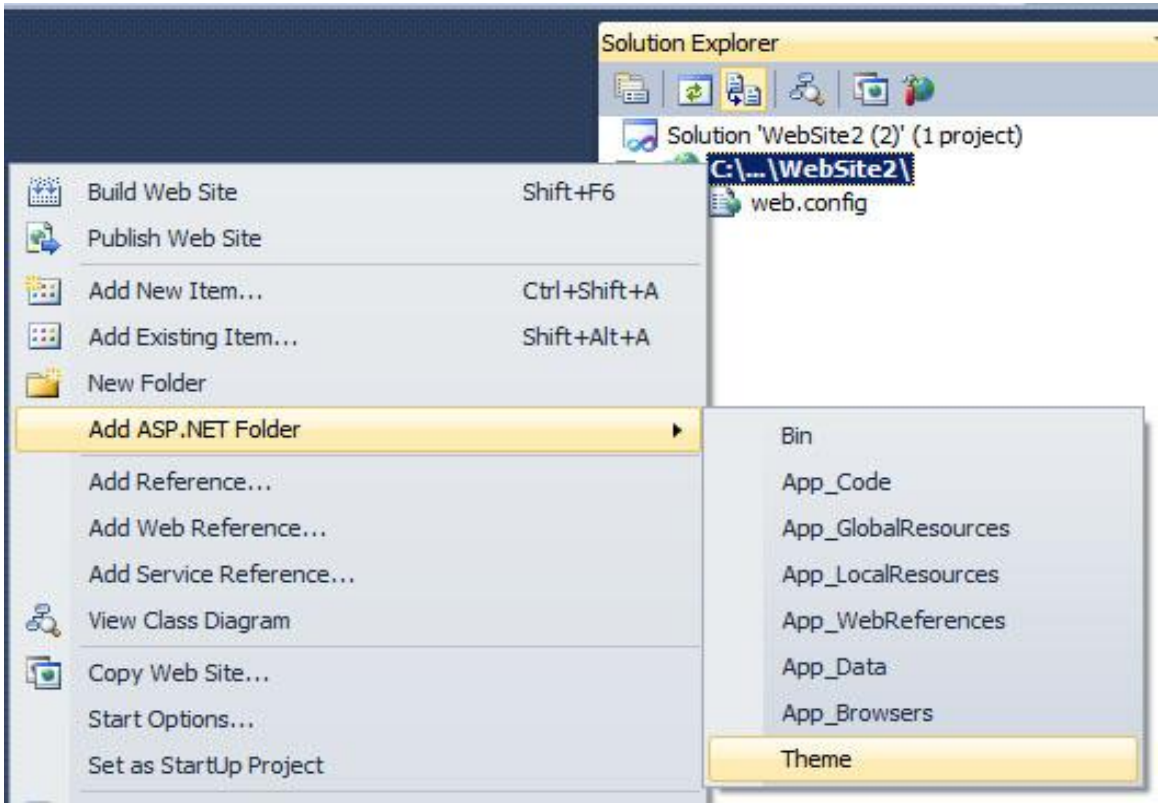
- Step-1
  - Select A MasterPage, then right click on the masterpage. It will display a list of items. Select Content Page.
- Step-2
  - Set required contents in that page.
- Step-3
  - Run the Page and check output.

Understanding Themes :

- A theme is a collection of property settings which allows us to define the look of pages and controls, and then apply that look consistently through all the Web pages in the web application.
- Themes are made of different set of elements such as Skins, Cascading Style Sheet(CSS), images and other resources.
- Themes are mostly defined in special directories.

Create New Theme :

- If there is no Theme folder in solution explorer. Then Right Click on project title.
- Select…
- Add ASP.Net Folder
- Add Theme.



- This ***Theme*** will create a new theme folder in solution explorer.

Elements of Themes :

- Skins
- Casecading Style Sheet (CSS)
- Images and other Resources
  - Skins
    - A skin file has an extension **.skin** and contains property settings for individual web controls of the page.
    - A **.skin** file can have more than one control skins for one or more control types.
    - We can define skins in a separate file for each control or can combine all the skins in a single file.

Elements of Themes : Skin

- Skin for Control
- Example :

```
<asp:button runat="server"
            BackColor="lightblue"
            ForeColor="Black"/>
```

- There are two types of control skins,
  - Default Skin
  - Named Skin

Elements of Themes : Skin

- Default Skin
  - A default skin is automatically applied to all controls of the same type when a theme is applied to a page.
  - A control skin is a default skin.
  - Default skin does not have a SkinID.
- Named Skin
  - A named skin is a control skin with a SkinID property set. Named skins do not automatically apply to controls by type.

Create SKIN : Default SKIN

- Right click on Theme.
- Select Add New ITEM.
- Select SKIN File with ***.skin*** extension.
- Put required formatted (Backcolor, ForeColor etc.) SKIN controls.
  - Like,
    - Label, Textbox, Button (Put it without ID and Text properties.)
- Put these all controls under skin file default <% … %> code.

Apply theme to WEBPAGE.

- To apply any theme to any web page we have to add…
  - Theme="<theme name>"
  - In, Webpage Page Directives.
  - Example :

```
<%@ Page Language="C#"
        Theme="ThemeName"
        AutoEventWireup="true"
        CodeFile="Default.aspx.cs"
        Inherits="_Default" %>
```

NOTE : Theme + Skin

- When we execute the web page then and then only the SKIN effect will be displayed on the controls.

Named SKIN = SkinID

- We can add SkinID in any theme control.
  - Just add new property in control with name **SkinID="IDName".**
  - If your control skin is without SkinID then it will be default skin.
  - If your control skin is with SkinID then it will be Name Skin. We have to add SkinId property in control of WEBPAGE.

In SHORT : Apply SKIN file...

- Add Skin File in current THEME Folder...
- **Insert in page directive theme...**
- Now add required controls to the **.skin** file
- Example 1:

  <asp:Label runat="server" **ForeColor="Red"**/>

  Note :

  *This skin will apply on **all the label control***
- Example 2:

  <asp:Label runat="server" **ForeColor="Red"**
  SkinID="lblLabel" />

  Note :

  *This skin will apply to only those labels which has **SkinID="lblLabel"***

Elements of Themes : CSS

- Casecading Style Sheet (CSS):
  - A theme can also include a CSS file.
  - We can define the CSS files in theme folder.
  - When we put a "**.css**" file in the theme folder, the style sheet is applied automatically as part of the theme.
- Images and Other Resources :
  - We can also include graphics and other resources, such as script files or sound files.
  - Mostly the resource files are located in the same folder as the skin file for that theme.

Types of Themes :

- There are two types of Themes :
  - Page Themes
  - Global Themes
- Page Themes :
  - A page theme is a theme folder with control skins, style sheets, graphics files and other resources which are created as a sub folder of the \App_Themes in our web application.
  - Each theme is a different subfolder of the \App_Themes folder.

Types of Themes :

- Global Themes
  - A global theme is a theme that is applied to all web sites on a server.
  - Global themes allows us to define an overall look for the domain which has multiple websites.
  - Global themes are like page themes they also include control skins, style sheets, graphics file and other resources.
  - Global themes are stored in a folder named themes. This folder is global to the web server.
  - Any website or page in the website on this server can reference global theme.

Applying Themes :

- We can apply themes to various pages or web applications by using theme or StyleSheetTheme attribute.
- Global Themes can be applied setting the <pages> element in **Web.Config** file.
- A theme setting in the **Web.Config** file is applied to all ASP.NET Web Pages in the current web application. It can be done by adding the following code in **Web.Config** file.

Applying Themes :

- Apply Themes to Web.Config

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

Applying Themes :

- Apply styleSheetThemes to Web.Config

```
<configuration>
<system.web>
<pages styleSheetTheme="ThemeName"/>
</system.web>
</configuration>
```

- Page Themes can be applied to a Web Page in Web Application by using StyleSheetTheme or Theme attributes of the Page Directive. It can be given as :

```
<%@ Page Theme="ThemeName" %>
<%@ Page StyleSheetTheme="ThemeName" %>
```

Difference between Theme and StyleSheetTheme

- The StyleSheetTheme attribute works same as the Theme Attribute.
- The difference is when the attributes of a particular control are set locally on the page; the attributes are overridden by the global theme if we use the Theme attribute.
- While the using StytleSheetTheme the attributes of a particular control are not overridden they remain same as that difined in the local page.

Difference between Theme and StyleSheetTheme

- Suppose we have a text box control like the following :

```
<asp:Textbox ID="txt1" Runat="server"
                    ForeColor="#ffffff"/>
```

- In above example, the ForeColor attribute is overridden by the theme if we apply theme using the theme attribute in the Page directive. If, we have applied the theme using the StyleSheetTheme attribute in the Page Directive, the ForeColor attribute remain in place, even if they are explicitly defined in the theme.

Themes & Cascading Style Sheet:

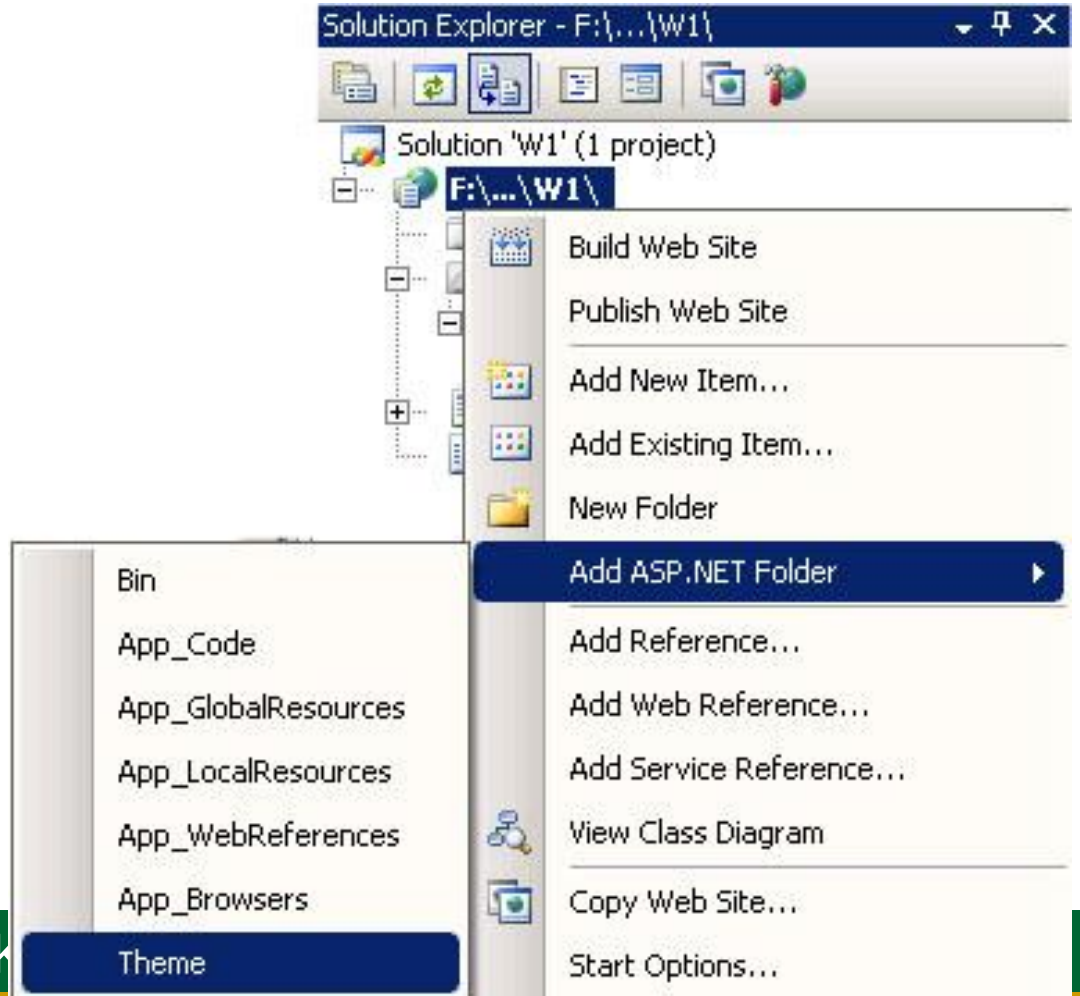
- Themes are quite similar to CSS.
- By default, the themes are defined in CSS file.
- In both themes and CSS we can define a set of common attributes that can be applied to any web page.
- **Themes V/S Cascading Style Sheet :**
  - Themes can be used to define many properties of a control or a web page. Theme just not only define style properties.

Themes V/S Cascading Style Sheet

- Themes can include graphics.
- Themes do not cascade the way the style sheets does.
- We can define the property values locally as well as globally using Theme or StyleSheetTheme attribute.
- We can apply only one theme to each page.
- We cannot apply multiple themes to a page, while multiple style sheets can be applied to one page.

Steps-1 To Set Themes & CSS

- Create a new Web Site
- Right Click on the Solution Explorer and Add ASP.NET Folder, in that select theme.

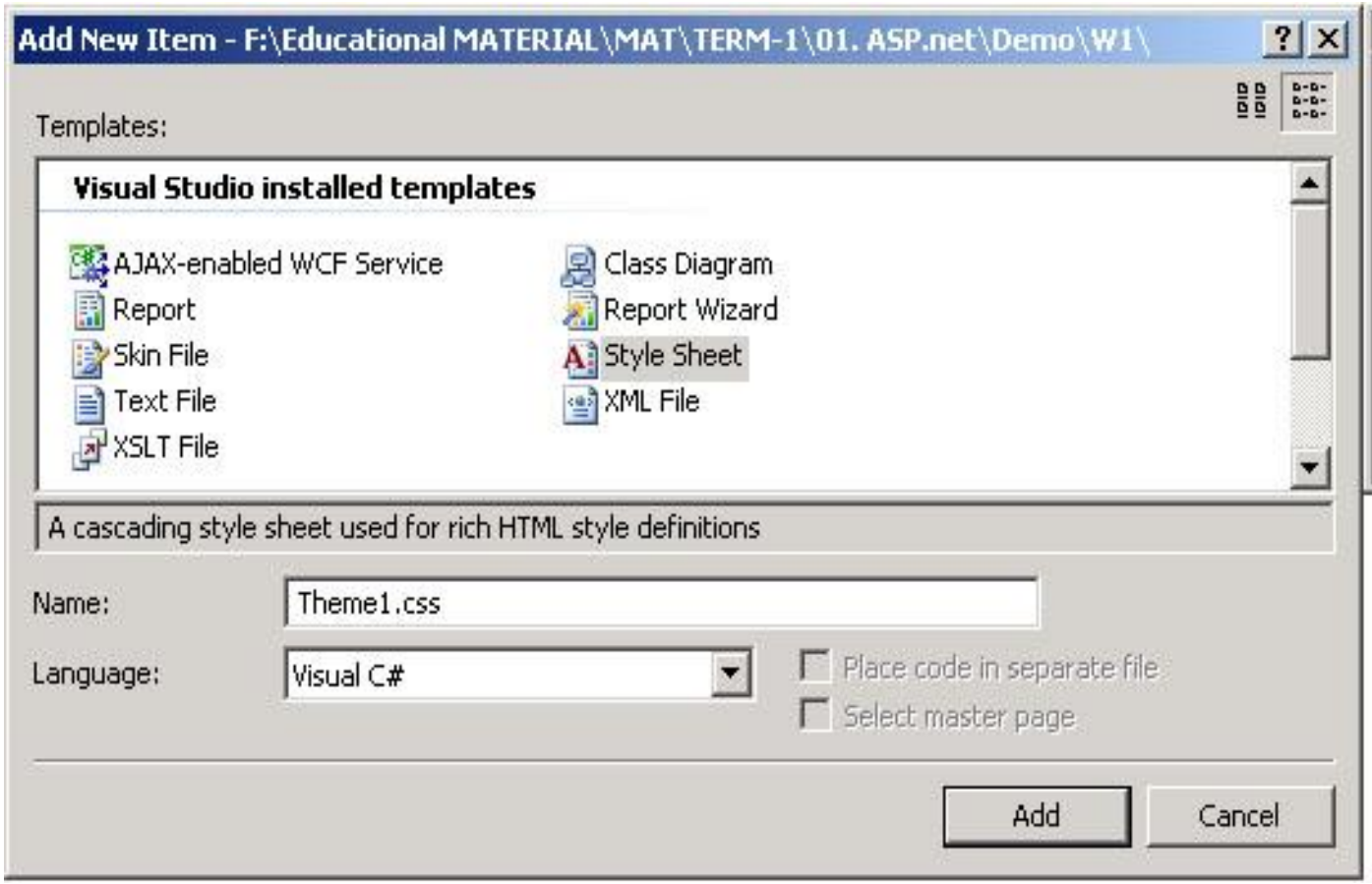


Steps-2 To Set Themes & CSS

- When we click on the Theme option then automatically App_Themes folder is added to Web applicaiton.
- It will also create Theme1 floder In App_Themes folder automatically.

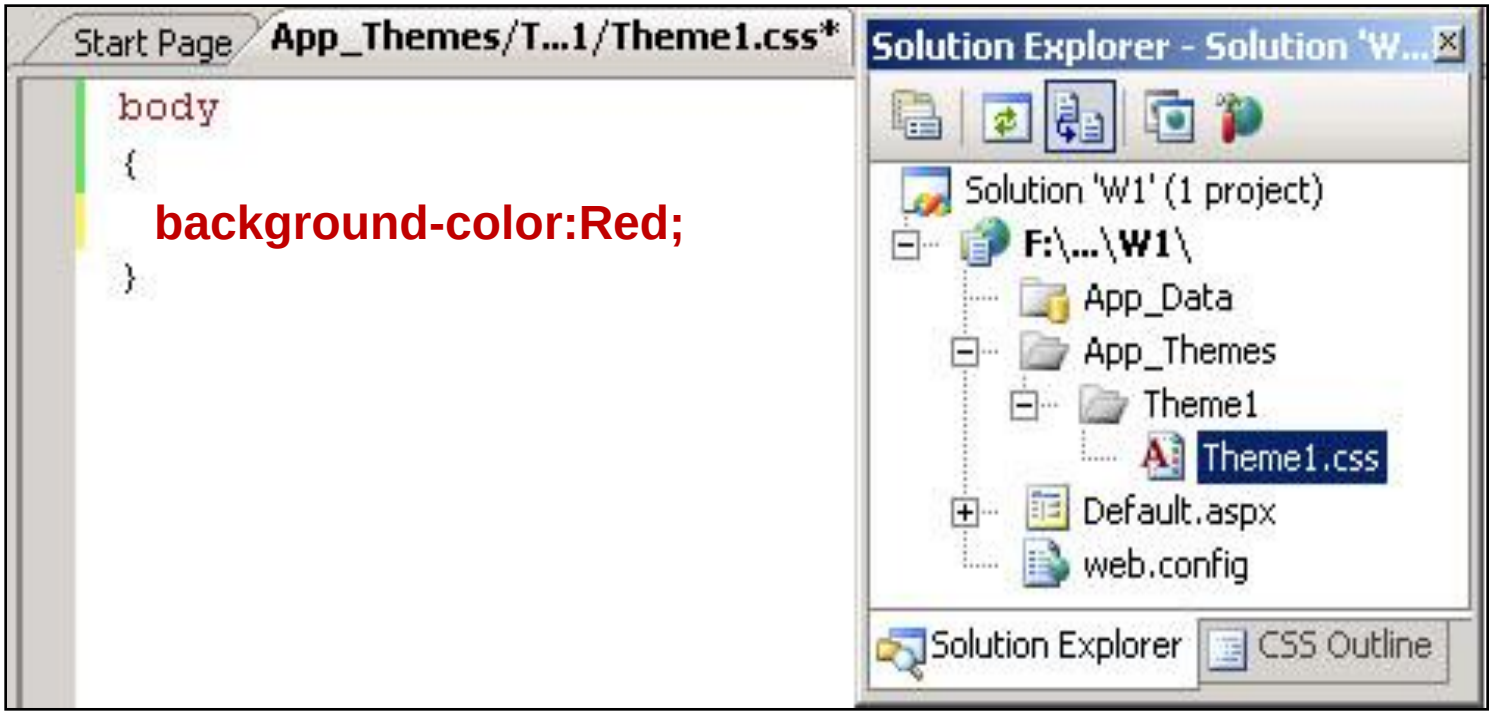
Steps-3 To Set Themes & CSS

- Right click on the Theme1 Floder and add New Item and select StyleSheet and rename it as Theme1.css



Steps-4 To Set Themes & CSS

- Double click on Theme1.css
- It will display a **Theme page**...



- Add following code in the **Theme1.css** file.

```
background-color:Red;
```

Steps-5 To Set Themes & CSS

- Select .aspx file and goto page directive then add **Theme="Theme1"**
- Add your name as Label Text and run the application.
- This will display your required output....

Link a Style Sheet to ASP.NET page

- We add a theme to ASP.NET page by setting the Theme or StyleSheetTheme attribute of @Page directive.
- We can also use Link element to add link a style sheet with Web Page.
- Link element is used to link the style sheet to individual ASP.NET Web Pages and apply the styles to the elements on the page.

Link a Style Sheet to ASP.NET page

- To link a cascading style sheet to an ASP.NET Web page :
  - From the Design view, **from Solution Explorer**, ***just drag the style sheet file****
  - In source view, from Solution Explorer, drag the style sheet file (.css) and drop it within the <head> tag on the Web page. A new link element would be inserted inside the <head> tags, which might look like the following code ***example***.

```
<link href="/Theme1.css" rel="stylesheet"
        type="text/css" />
```

Class and ID Selectors :

- We can also define style for Web Page by using Class and ID selectors.
- The benefit of using them is that same HTML element can be presented differently depending on its Class or ID.
- Class Selector :
  - We can define a class selector by name preceded by a full stop (.).
  - In the class selector block we can define the style we want.

```
.intro{      color:red;
             font-weight: bold; }
```

Class and ID Selector :

- To use this class selector, class tag would be used in HTML tags.
- For example,
  - If we want to use the above class tag in tag, then it can be ginve as :

     This is for testing
- ID Selector :
  - We can define an ID selector by name preceded by a hash character (#). ID tag is also used to set the style of the web page.

Class and ID Selector :

- It is used inside a div tag.
- An example of ID selector can be given as

```
#top{       background-color: #ccc;
            padding: 20px }
```

- New to use this ID selector we need to define a div and assign its ID as this name
- It can be given as below :

```
<div id="top">
  <h1>Test Page</h1>
</div>
```

Class and ID Selector :

- NOTE :
  - ID and Class selector are usually defined inside the StyleSheet file (.CSS file) and are used on the Web Page. They can also be defined on the web page using script tag.
  - The difference between an ID and a Class is that and ID as the name suggest can be used to identify one element, whereas a class can be used to identify more than one.

Asp.Net

Caching Application Pages and DATA

Caching Introduction :

- Caching is used by applications to store data temporarily either on Web Server or on the Client Side.
- Caching is mostly used in situations where several programs continuously access the same set of data. So make data access faster, this data is stored in a cache and can be used for future purpose.
- Usually the data accessing in ASP.NET is quite slow and time consuming as it involves procedure of connecting to the database and retrieving the data.

Caching Introduction :

- Caching is one of the best alternatives to improve the performance of Web Application, as it allows to access data from cache itself, not from the Server Database or from the application.
- Caching not reduces the processing time but it also reduces network traffic, hereby improving the performance of the Web Application and enhancing user experience.

Caching in ASP.NET

- The execution of applications in ASP.NET is quite faster than the applications written in classic ASP.
- With ASP.NET, caching techniques has improved dramatically.
- Caching is the technique for storing the frequently used data in memory for immediate access to the request program calls.

Caching in ASP.NET

- Caching is one of the important features available to improve performance of web applications.
- It minimizes the usage of server resources for fetching frequently used data, and thus by improving the performance of the application.

Reason to USE Caching :

- Every time the page request some information, it has to go through cycle consisting of Client, Server and Database.
- Again in the response it has to go through another cycle consisting of database, server and client.
- This might increase the time, which in turn affect the performance, that’s why to improve performance caching is used.

Reason to USE Caching :

- When multiple users want to access the same information again and again at that time caching is considered as one of the important aspect.
- Here, data would be cached in memory and thus avoiding the database access with every pages request, which increases the performance of the applications.

Advantages / Disadvantages of Caching

- Advantages :
  - Cache improves the performance of the Website.
  - It takes heavy load or execution from the server for the repeated operations.
  - It reduces load on Database or Web services.
  - Storing data using cache is quite reliable.

Advantages / Disadvantages of Caching

- Disadvantages :
  - It will delay while accessing Website as the data needs to written before accessing.
  - It would be an overhead for the Client to carry out this activity again and again.
  - It would lead to an increase in maintenance, as the cached data need to get updated regularly.
  - Again there would also scalability issues. **(Scalability= માપનીયતા)**

Different types of Caching :

- ASP.NET provides different kinds of caching techniques to cache a completely rendered Web Page.
- Caching Techniques are :
  - Page Output Caching
  - Partial Page Caching
  - Data Caching

Page Output Caching :

- The page output caching is one of the simplest technique of caching in ASP.NET.
- In this technique complete HTML output of a rendered ASP.NET page is stored in cache and all the subsequent requests of the page are sent from the cache itself.

Page Output Caching :

- The cache engine checks the cache for the existing page requests. This engine is also known as **cache server**, as it acts as a server that stores web pages and other internet content of request locally.

Page Output Caching :

- If the engine finds a request for the requested page in the cache, the cached HTML output is sent to the client.
- If no request is found, then the engine starts rendering the page again and saves a copy of the HTML output in cache.

Page Output Caching :

- Page Output Caching is very useful when we have static pages.
- As with static pages the content of the pages does not change frequently. So cache server would cache the HTML output once and for subsequent requests, the engine sends cached output.

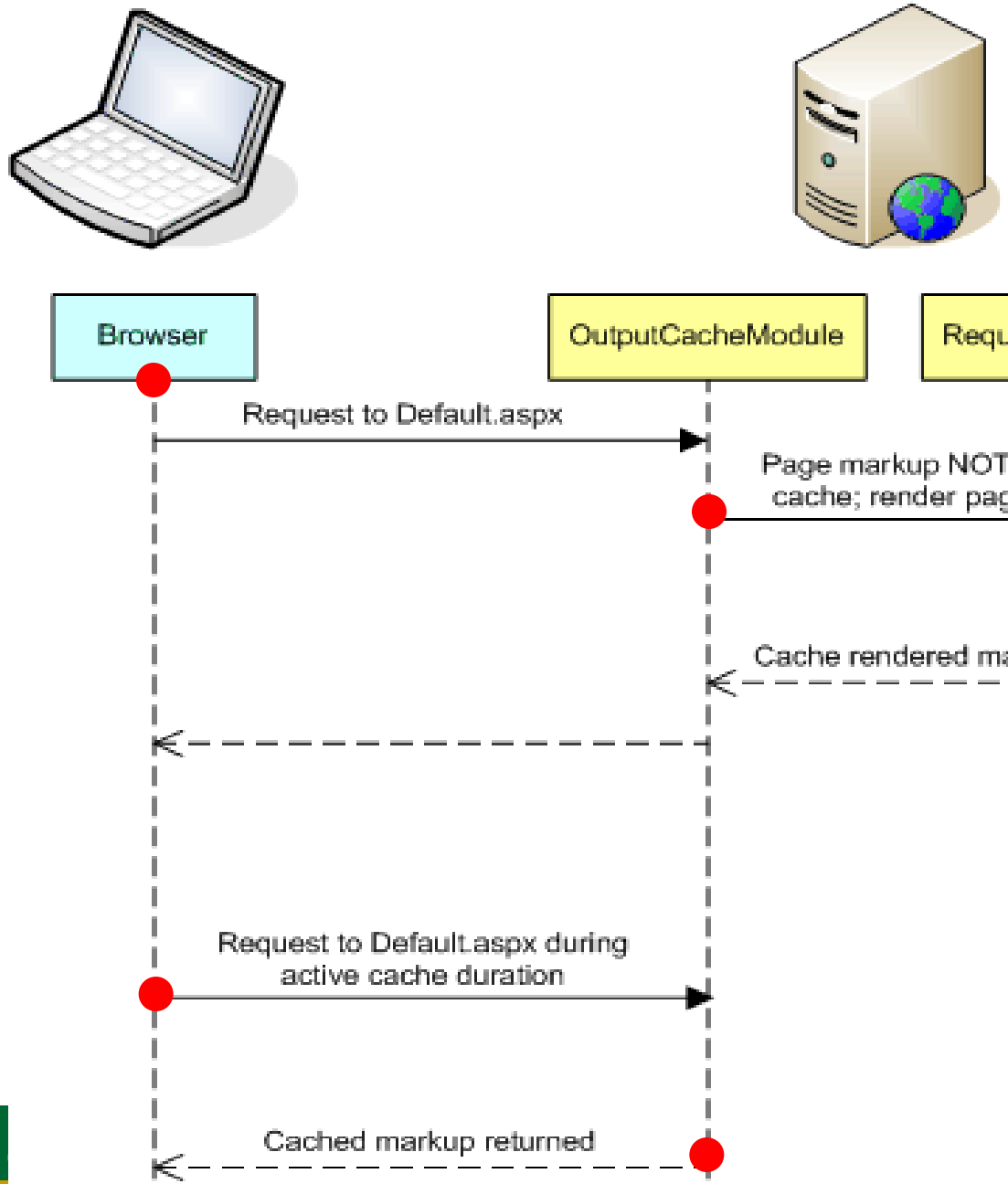
Page Output Caching :

- Because of static page, it reduces the load on the web server as the page is not rendered each time a request is made; it is cached from the cache server.
- In Output Caching the first time Browser makes a request to ASP.NET web page.

Page Output Caching :

- Since the web page is not stored in the cache for the first time, the cache server passes the request to requested ASP.NET page.
- Now the request page would be cache and stored on the cache server.
- For the next request to the same page, the output would be rendered from the Cache Server it would not go to the Requested ASP.NET page.

Page Output Caching :



Page Output Caching :

- Here the cached output would be active for some duration only. So if in that duration if the output is requested then the output would be rendered from the Cache Server, else it would go the Requested Web Page.
- We can implement Page Output caching by using @OutputCache directive.

Page Output Caching : Directive

Attribute	Description
CacheProfile	It contains various settings of the cache which are associated with a Web Page. It is an optional attribute and by default has an empty string.
Duration	It defines the time duration for the Web Page to be cached. It is required attribute. It is usually defined in seconds.
Location	It specifies the location where the cache is created. The values for this attributes are any, Client, Downstream, Server, ServerAndClient and None.

Page Output Caching : Directive

Attribute	Description
Nostore	It is Boolean value which determines whether to prevent secondary storage of the sensitive information.
Shared	It is a Boolean value which allows sharing of a user control output with multiple pages.
SqlDependency	It contains a string value that identifies a set of database and table name pairs on which the output cache of a control or web page depends.
VaryBy Control	It contains a semicolon separated list of strings used to vary the output cache. This attribute is used with user controls only.

Page Output Caching : Directive

Attribute	Description
VaryBy Custom	It defines the custom strings for custom output caching requirements.
VaryBy Header	It defines a list of HTTP headers used to vary the output cache.
VaryBy Param	It is used to cache different copies of a Web Page when it is generated dynamically, based on the parameters received from HTTP POST or GET request. It is required field.

Page Output Caching :

- To implement Page Output Caching, add the following code at top of the Web Page to implement Page Output Caching :

```
<@OutputCache Duration="10"
                    VaryByParam="none"%>
```

- We can also store the cached data, on client side or on the server side or proxy by using Location attribute in the @OutputCache directive.

```
<@OutputCache Duration="90"
  Location="Any|Client|Downstream|Server|
  None" VaryByParam="none" %>
```

Def. Demonstrate use of output caching.

- Step-1
  - Add a new web page.
  - Put 2 Labels on that page
  - Label1 with TEXT="Output Caching 9Sec"
  - Label2 with Date&Time

    Label2.Text = DateTime.Now.ToString("hh:mm:ss");
- Step-2
  - Add the following code below <!DOCTYPE>

<%@OutputCache Duration="9" VaryByParam="none"%>

- Step-3
  - Execute and Refresh the page again & again and check the output.

Partial Page Caching **or** Fragment caching

- Caching of a complete requested web page is not required always.
- We may need implement caching on specific part of a web page. The remaining content of the web page would be refreshed each time the page is requested. This is called Partial Page Caching **or** fragment caching.

Partial Page Caching **or** Fragment caching

- **Step to apply partial page caching :**
  - Identify the parts of web page that requires more processing than the other parts, while rendering the page.
  - Create user controls and decide the caching policies for them. Caching policies would check the validity of the cached data on each user request.

Partial Page Caching **or** Fragment caching

- Add the user controls in the web application. Now caching of the defined user controls would only be done. While other controls would be refreshed for each request.

Def. Demonstrate Partial Page Cach

- Step-1
  - Create a new empty website.
- Step-2
  - Right click on the Website Name on the Solution Explorer and **add** a web user control (**.ascx** file).
- Step-3
  - Check **.ascx** page directives.
  - Ad a Label control in **.ascx** file & set property like, ID = lbltime, EnableViewState=False, ForeColor=Red

Def. Demonstrate Partial Page Cach

- Step-4
  - In the **WebUserControl.ascx** add given code.

```
<%@ OutputCache Duration="5"
                    VaryByParam="None" %>
```

- Step-5
  - In the code behind file of **WebUserControl.ascx.cs** add given code

```
protected void Page_Load(object sender, EventArgs e)
{ lblTime.Text = DateTime.Now.ToString();
}
```

Def. Demonstrate Partial Page Cach

- Step-6
  - Add new Web Form (**.aspx**)
  - Add WebUserControl.**ascx** file using **@Register** tag.

```
<%@ Register
  Src="~/WebUserControl.ascx"
  TagPrefix="uc" TagName="UserControl"
  %>
```

Def. Demonstrate Partial Page Cach

- Step-7
  - Add Following code in default webpage.

```
<div>
    Try to refresh this page, the Date time
  doesn't change till 5 second.
  <uc:UserControl runat="server"
      ID="uc1" />
  <asp:Label ID="lblTime1"
      EnableViewState="false"
      ForeColor="Blue" runat="server"
      Text="Label"></asp:Label>
</div>
```

Def. Demonstrate Partial Page Cach

- Step-8
  - Add a new label in Default.aspx and set given properties.
    - Id=lblTime1, EnableViewState=False, Folrecolor=Blue
  - Add following code in Default.aspx file.

```
protected void Page_Load(object sender, EventArgs e)
{
        lblTime1.Text =
              DateTime.Now.ToString();
} Now, execute and refresh the file.
```

Data Caching :

- Data caching is a technique used to store the data in the Cache Memory.
- Here we would retrieve the data from the database and store the data in the Data Cache.
- It is quite efficient to retrieve data from the data cache rather than from the database or other sources.
- For this we need to use **System.Web.Caching** namespace available in ASP.NET.

Data Caching :

- The scope of this data cache is within the application domain. Every user who accesses the website can have access to this object.
- If the requests come again then data would be cached from Cache Memory (IIS).

Data Caching :

- Here also we can set the expiration period time after which the data would be again cache from database and stored in the cache memory.
- For creating a data cache, we have to use Cache Keyword. It is available in the **System.Web.Caching** namespace.
- Basic Syntax :

```
Cache["VariableName"]="Value";
```

Data Caching :

- There are three main things needed to perform data caching :
  - Inserting Value in Cache
  - Retrieve the Cached Value
  - Deleting the Cached Value

Data Caching :

- Inserting value in Cache : For inserting a value in Cache, ASP.NET provides 4 methods.

  Cache.Insert(Key, Value)

  - This method inserts a value into cache with **Key** name and **Value** with default priority and expiration.

Data Caching :

Cache.Insert(Key, Value, Dependencies)

- This method inserts a value into cache with **Key** name and **Value** with default priority and expiration, with CacheDependency. (અવલંબન)
- When any of the dependency changes, then the object becomes invalid and is removed from the cache.
- If there are no dependencies, then this parameter has a null value.

Data Caching :

Cache.Insert(Key,Value, Dependencies, AbsoluteExpiration, SlidingExpiration)

- This method adds an expiration policy along with all other things mentioned above.

Cache.Insert(Key,Value, Dependencies, AbsoluteExpiration, SlidingExpiration, Priority)

- This method adds a priority to inserted value in the cache. The **Key** name with low priority is deleted before the Key name with higher priority.

Data Caching :

- Priority is an object of CacheItem priority Enumeration.
- It has the following possible Values :
  - AboveNormal
  - BelowNormal
  - Default
  - High
  - Low
  - Normal
  - NotRemovable

Data Caching :

- Retrieve the Cached Data :
  - Retrieving the Cached Data from Web Page is one of the simplest tasks in ASP.NET. We need to simply specify the key and value that represent the cached data.
  - For this, we need to write the following line of the code :

```
value=Cache("Key");
```

Data Caching :

- Here value would now contain the data cached from the variable named as Key.
- Each time before using the retrieved value from the Cache we need to check whether the retrieved value is null or not.

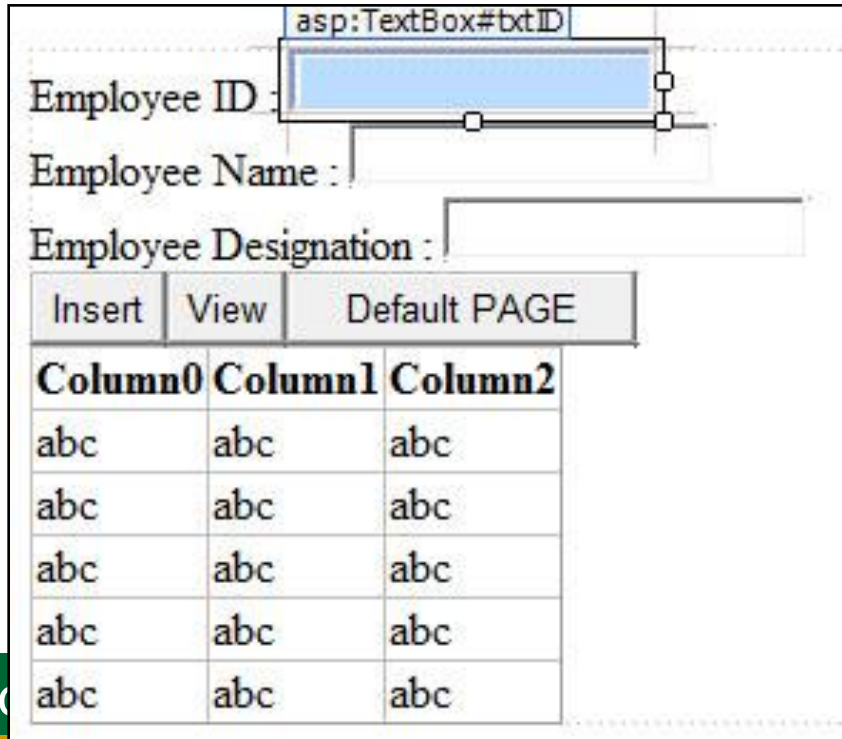
Data Caching :

- Deleting the Cached Value :
  - Basically data would be automatically deleted from the cache on the basis of its expiration policy.
  - To delete the data from cache, we need to write the following line of code:

    Cache.Remove("Key");
  - Here Key indicates the Cached Value to be deleted from the cache.

Def. Demonstrate Data Caching...

- Step-1
  - Create a new website and new web form.
- Step-2
  - Rename the web form with **insert.aspx**
  - Design the forms as given.



Def. Demonstrate Data Caching...

- Step-3
  - Create a new database table with name **employee** with fields id, name, designation.
- Step-4
  - Proper Code for buttons **INSERT** (to add new record), **View** (To display records in gridview), **Default page** (To redirect page to Default.aspx page).
- Step-5
  - Create **Default.aspx** page.

Def. Demonstrate Data Caching...

- Step-6
  - In **Default.aspx** insert Gridview control.
- Step-7
  - Code following for Data Caching.
    - Define Connection
    - Open Connection
    - Define Dataset
    - Then CODE FOR **DATA CACHE.**

Def. Demonstrate Data Caching...

```
con = new SqlConnection(@“”);
con.Open();
DataSet ds;
if (Cache["employee"] == null)
{ ds = new DataSet();
  SqlDataAdapter da;
  da = new SqlDataAdapter("select * from
                                    employee", con);
  da.Fill(ds, "employee");
```

Def. Demonstrate Data Caching...

```
  Cache.Insert("employee", ds, new
      System.Web.Caching.CacheDependency(
      Server.MapPath("Master.xml")),
      DateTime.Now.AddSeconds(45),
      TimeSpan.Zero);
  con.Close();
}
else
{ ds = (DataSet)Cache["employee"]; }
  GridView1.DataSource = ds;
  GridView1.DataBind();
}
```

Cache Expiration Policy :

- ASP.NET cache provides a great feature through which we can increase the performance of the website by reducing the server round trips to the database.
- We can also cache any serializable data into cache memory.

Cache Expiration Policy :

- There are many ways to cache data; one of the simplest way is to use an insert method for storing data in caching.
- After data is stored in Cache we need to periodically validate cache and also set the expiration time for cache as data may be updated on the database.

Cache Expiration Policy :

- So for that we need to define an expiration time after this defined time the cache will expire and then it will again get the new fresh data from the database into the cache. This method is called as Time Base Expiration.
- There are two ways of expiration policy used
  - Absolute Cache Expiration
  - Sliding Cache Expiration

Cache Expiration Policy :

- Absolute Cache Expiration
  - Absolute Cache Expiration means that the data from cache will be removed after a fixed amount of time, starting form the time to cache is activated.
  - Generally we use this technique when we are displaying data which are not caching frequently.

Cache Expiration Policy :

- Sliding Cache Expiration :
  - Sliding Cache Expiration means that it expire cache after time period at the time of activating cache and also if no requests are made during this time period.
  - This type of expiration is useful in situation where there are many data to cache.
  - In this method only those data would be cached which are frequently used in the application. So there is not unnecessary wastage of memory.