Agenda

Deccansoft

Caching

Types of Caching: Output Caching / Fragment Caching / Data Caching / Substitution Caching.

Output Caching: Here the response of the request is cached for a predefined period and the same is reused for all subsequent requests for that page.

Cache.aspx:

```
<%@ OutputCache Duration="10" VaryByParam="txtDemo"  VaryByHeader="referer"
                                    VaryByCustom="Browser" Location="ServerAndClient" NoStore="true"%>
<asp:Label ID="lblCurrentTime" runat="server"></asp:Label>
<asp:TextBox ID="txtDemo" runat="server"></asp:TextBox>
<asp:Button ID="btnDemo" runat="server" Text="Demo" />
```

Cache.aspx.cs

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

Important points:

1. **Duration** and **VaryByParam** are the only required attributes and other attributes are optional.
2. The value of "**VaryByParam**" can be either **QueryString** parameter, **Cookie** Name, **Form** element Name or **Id** of **Control** on the WebForm.
3. For every value of the parameter (parameters) mentioned in VaryByParam, a different cache copy of the response is maintained on the server and same is reused for those values of parameters for subsequent requests.
4. **VaryByParam** can be "**none**" or "*"(create separate cache copy if any parameter changes) or ";" separated list of parameters.
5. **VaryByCustom**="Browser" → For different browser types a different cache is created.
6. **VaryByHeader** = "Accept-language" → Different for every value of request header submitted along with the request.
7. **Location:**

Any	Cache can be located on the browser (client) or Proxy or Server
None	Cache is disabled everywhere
Downstream	Cache can be located either on browser (client) or proxy
Server	Cache is located on the server
Client	Cache is located on the browser (client)
ServerAndClient	Cache on both Server and Client but not on Proxy

8. **NoStore**="true" → page **should not** be cached on the client irrespective of the browser settings (By Default it is false).

Deccansoft

- When Tracing is enabled on the page, response of the page is not cached.
- Browser caches the response of the page based on the response header “Cache-Control” and it’s based on “Location” attribute of OutputCache directive.

To specify cache settings programmatically: (Comment the OutputCache directive in aspx page)

```
protected void Page_Load(object sender, EventArgs e)
{
    lblCurrentTime.Text = DateTime.Now.ToLongTimeString();
    Response.Cache.SetCacheability(HttpCacheability.ServerAndPrivate);
    Response.Cache.SetExpires(new DateTime(2009, 12, 31, 10, 20, 30)); //For setting the absolute Data and Time.
    //Response.Cache.SetMaxAge(new TimeSpan(0, 0, 10)); //For mentioning the relative Data and Time.
    Response.Cache.VaryByHeaders["referer"] = true;
    Response.Cache.SetValidUntilExpires(true);
    Response.Cache.VaryByParams["txtDemo"] = true; //works only if SetValidUntilExpires is set to true
}
```

Web.Config:

Web.config:

```
<system.web>
  <caching>
    <outputCacheSettings>
      <outputCacheProfiles>
        <add name="CacheInConfig" enabled="true" duration="10" location="ServerAndClient"/>
      </outputCacheProfiles>
    </outputCacheSettings>
  </caching>
</system.web>
```

In CacheDemo.aspx:

CacheDemo.aspx:

```
<%@ OutputCache  VaryByParam="txtDemo"  CacheProfile="CacheInConfig"%>
```

- Changes made to profile(i.e OutputCacheProfile) are automatically effected to all the webforms.

Fragment Caching

Here instead of caching the complete response of the page, only a small portion of it will be cached.

Step1 : Add the following to the webform

Server Time: <%= DateTime.Now.ToLongTimeString() %>

Step2: To the Project → Add New item→Web UserControl (CacheControl.ascx)

Step3: In CacheControl.ascx add the following

CacheControl.ascx:

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

Step4: Add to UserControl

Append to UserControl:

```
<%= DateTime.Now.ToLongTimeString() %>
```

Step5: Drag and drop UserControl in WebForm.

Step6: Run the application and Refresh the webform to note that the time rendered from UserControl is not changing but the time rendered from webform is changing.

Substitution Caching

If the response of the page is cached, after first request, for subsequent requests the cached output is rendered. If the webform has a Substitution control, even if the cached copy is rendered, the return value of the MethodName of the substitution control is used by ASP.NET for rendering latest content.

Step1: Cache the response of the page (Default.aspx) using

Cache.aspx:

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

Step2: Add the following to the webform

Cache.aspx:

```
Server Time: <%= DateTime.Now.ToLongTimeString() %>
```

Step3: Drag and Drop **Substitution control** from ToolBox onto the WebForm.

Step4: Set its MethodName property **= "GetServerTime"**

Step5: Add the method as below to the aspx.cs file

Default.aspx.cs:

```
static string GetServerTime(HttpContext context)
{
    return DateTime.Now.ToLongTimeString();
}
```

The return value of the above function is rendered in position of Substitution Control.

Deccansoft

Step6: Run the application to view the webform in webbrowser. Refresh the page to note that overall response of the page is cached but latest time (return value of GetServerTime) is rendered in place of substitution control.

Data Caching

Instead of caching the response of the page we only cache the **data** used in making the response of the page. The cached data can be then reused for subsequent request by any client for any WebForm i.e. the cached data is global. When the memory is insufficient the Items in data cache automatically expires.

Step1: Add to the project a new webform - DataCacheDemo.aspx

Step2: Add to .aspx file the following two lines:

DataCacheDemo.aspx

```
Server Time: <%= DateTime.Now.ToLongTimeString() %> <br />
Time From Cache: <asp:Label ID="lblTime" runat="server"/>
```

Step3: In Page_Load add the following:

Page_Load

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Cache["dt"] == null)
        Cache["dt"] = DateTime.Now;
    DateTime dt = (DateTime)Cache["dt"];
    lblTime.Text = dt.ToLongTimeString();
}
```

Step4: Run the webform and refresh it multiple times.

Note that the Server time is changing but the cached time is not changing.

To set expiration policy to expire the data cache:

a) Cache Item dependency on file.

Step5: Append to .aspx file the following

DataCacheDemo.aspx

```
<asp:Label ID="lblDemo" runat="server"/>
```

Step6: Project → Right click→ Add New item→Text File (Demo.txt) – Add some text to it

Step7: Top of .aspx.cs file add the following

DataCacheDemo.aspx.cs

```
using System.Web.Caching;
using System.IO;
```

Step8: Append the following to Page_Load

Append to Page_Load

```
if (Cache["file"] == null)
{
    lblDemo.Text = "Fetched from file: ";
    string pp = MapPath("~/Demo.txt"); //To convert Virtual path to physical path
    StreamReader sr = new StreamReader(pp);
    CacheDependency dep = new CacheDependency(pp);
    Cache.Insert("file", sr.ReadToEnd(), dep); //The cache item is dependent on file and if file changed the
cached item immediately expires.
    sr.Close();
}
else
    lblDemo.Text = "Fetched from Cache: ";
lblDemo.Text += (string)Cache["file"];
```

Step9: Run the webform, first time the content if fetched from file. On refresh it fetched from Cache.

Step10: In Studio, open Demo.txt and modify…

Step11: Refresh the webform to note that again the content is fetched from file.

b) Cache Item dependency on Absolute Time.

Step12: Add the following to .aspx file

DataCacheDemo.aspx

```
<asp:Label ID="lblTime1" runat="server"></asp:Label>
```

Step13: Append the following to Page_Load

Page_Load

```
if (Cache["dt1"] == null)
    Cache.Insert("dt1", DateTime.Now, null, DateTime.Now.AddSeconds(10), Cache.NoSlidingExpiration);
lblTime1.Text = ((DateTime)Cache["dt1"]).ToLongTimeString();
```

Step14: Run the WebForm, Keep refreshing…Note that every 10 seconds the label shows new server time.

c) Cache Item dependency on Sliding Expiration

If the data cache remains unused for a pre defined period the data must expire

Step15: Add the following to .aspx file

DataCacheDemo.aspx

```
<asp:Label ID="lblTime2" runat="server"></asp:Label>
```

Step16: Append the following to Page_Load

Page_Load

Deccansoft

```
if (Cache["dt2"] == null)
    Cache.Insert("dt2", DateTime.Now, null, Cache.NoAbsoluteExpiration, new TimeSpan(0, 0, 10));
lblTime2.Text = ((DateTime)Cache["dt2"]).ToLongTimeString();
```

Step17: Run the WebForm, Keep refreshing…Note that even after 10 seconds the label shows cached time.

Step18: Refresh the form after waiting for 10 secs…this time the label shows server time because after 10 seconds from last request the cached item “dt2” expired.

Additional available methods:

```
Response.AddCacheItemDependency("dt");  //Response of the page in output cache expires if the data item "dt" expires
Response.AddCacheItemDependencies; // Cache item is dependent on Array of data items
Response.AddCacheDependency;
Response.AddFileDependencies; // Output Cache dependent on array of files
Response.AddFileDependency;  // Output Cache item is dependent on single file
```

Polling based SQL Cache Invalidation

Step1: Go to command prompt via Microsoft Visual Studio

Run the command for enabling the database for SQL cache dependency

aspnet_regsql.exe -S “.\sqlexpress” -E -d “DBName” –ed

Step2: Command for enabling the table for SQL cache dependency

aspnet_regsql.exe -S “.\sqlexpress” -E -d “DbName” –t “Emp” -et

Step3: In web.config add the following code

```
<system.web>
  <caching>
   <sqlCacheDependency enabled="true" pollTime="2000">
    <databases>
     <add name="DBNameInConfig" connectionStringName="csDemoDb"/>
    </databases>
   </sqlCacheDependency>
  </caching>
</system.web>
```

Step4: Add the following directive to the page

<%@ **OutputCache** Duration="9999" VaryByParam="none" **SqlDependency**="**DBNameInConfig**:**Emp**" %>

Or For caching only the data fetched by the SqlDataSource

<asp:**SqlDataSource** **CacheDuration**="Infinite" **EnableCaching**="true" **SqlCacheDependency**="MSNETDB:Emp"

Deccansoft

Notification based Cache Invalidation

Step1: Run the command for enabling the database for SQL cache dependency

```
aspnet_regsql  -S  ".\sqlexpress"  -E  -d  "DBCDemoDb" -ed
```

Step2: In Global.asax add the following code

```
void Application_Start(object sender, EventArgs e)  {
        System.Data.SqlClient.SqlDependency.Start (
                ConfigurationManager.ConnectionStrings["csDemoDb"].ConnectionString);
}
```

Step3: Add the following directive to the page

```
<%@ OutputCache Duration="99999" VaryByParam="none" SqlDependency="CommandNotification" %>
```

Or For caching only the data fetched by the SqlDataSource

```
<asp:SqlDataSource CacheDuration="Infinite" EnableCaching="true"
SqlCacheDependency="CommandNotification”>
```

Note:

- All Table Names and Column Names are CaseSensitive
- In Select commands “*” should not be used, column names must be mentioned explicitly.
- All the occurances of the table in different SqlDataSources on that page must be qualified by owner name **eg: dbo.Emp.**

Deccansoft