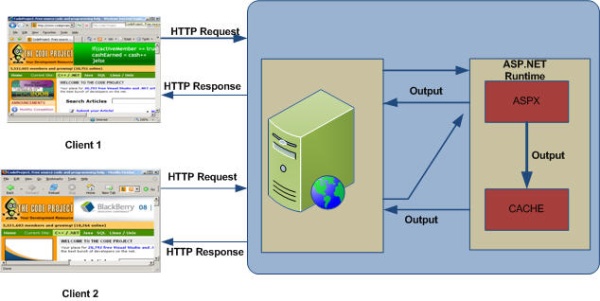
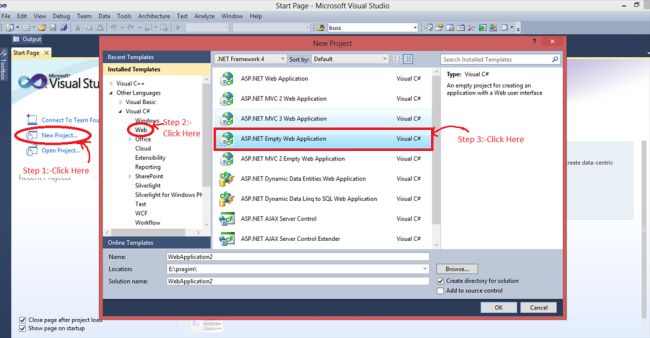
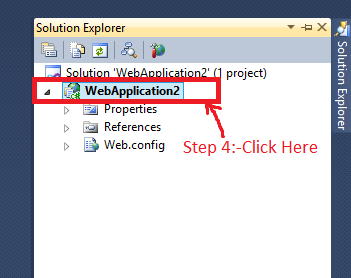
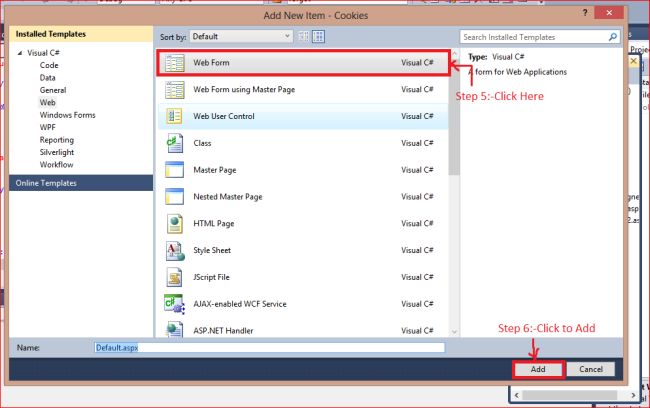
**Caching-**

Any site is developed for a large number of users, in other words for the same request to work the same for a large number of clients so it is hard to provide the best performance for the server so we use caching.  
  
Introduction  
  
Caching is a state management technique that can store a copy of the data in memory. To increase the performance of the application and improve the access time, caching is used. It exists in temporary storage, in other words when the data is no longer used then it expires. Using a cache we can retrieve the data from a database directly.  
  
Mostly, there are 3 types of caching.  
  
1. Output Caching  
  
When we compile our code it is first it compiled into Microsoft Intermediate Language (MSIL) then The MSIL is converted into Native Code using the JIT Compiler. Now if there is a page that changes frequently then the JIT needs to compile it every time. So, rather than generate a page on each request we can cache the page using Page Output Caching so that it can be accessed from the cache itself.  
  
  
  
These pictures show when the first request is generated, the page is cached for the same page request in the future. The page is retrieved in the future. The page is retrieved from the cache rather than regenerating the page.  
  
Now, here I am explaining the Output Cache with an example.  
  
Step 1: Open Visual Studio 2010.  
  
Step 2: Then click on "New Project" > "WEB" > "ASP.NET Empty Web Application".  
  
  
  
Step 3: Now click on Solution Explorer.  
  
  
  
  
Step 4: Now right-click on "Add" > "New Item"> "Web From" and add the name of the web form.   
  
  
  
Step 5: Write the following code and now here I have used a Drop Down Control, Grid Control, Label Control. and set the client-side time using JavaScript.

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Caching.WebForm1" %>

<%@ OutputCache Duration="30" VaryByParam="None" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

<div>    

<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"

onselectedindexchanged="DropDownList1\_SelectedIndexChanged">

</asp:DropDownList>

<br />

<br />

    <asp:GridView ID="GridView1" runat="server">

    </asp:GridView>

<br />

    Server Time:-<asp:Label ID="Label1" runat="server"></asp:Label>

<br />

<br />

Client Time:-

<script type="text/javascript">

    document.write(Date())

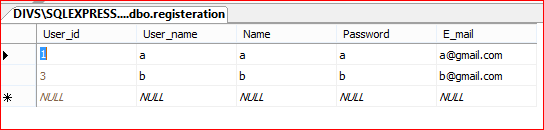
</script>

</div>

</form>

</body>

</html>

Step 6: Now, I used SQL Server 2008 and I created a database table with a name that is registered in the project database.  
  
  
  
Step 7: Code

protected void Page\_Load(object sender, EventArgs e)

{

    if (!IsPostBack)

    {

       Label1.Text = DateTime.Now.ToString();

       //call the Drop\_down\_Bind method in pageload

       Drop\_Down\_Bind();

    }    

}

private void Drop\_Down\_Bind()

{

    //bind the data in drop down

    SqlConnection con = new SqlConnection(@"server=divs\sqlexpress; Database=Project;uid=sa;password=divya");

    string query = "select User\_name from registeration";

    SqlDataAdapter da = new SqlDataAdapter(query, con);

    DataSet ds = new DataSet();

    da.Fill(ds);

    DropDownList1.DataSource = ds;

    DropDownList1.DataTextField = "User\_name";

    DropDownList1.DataBind();

}

private void databind(string User\_name)

{

    //bind the data in gridview

    SqlConnection con = new SqlConnection(@"server=divs\sqlexpress; Database=Project;uid=sa;password=divya");

    string query = "select Name,Password,E\_mail from registeration where User\_name='"+User\_name+"'";

    SqlDataAdapter da = new SqlDataAdapter(query, con);

    DataSet ds = new DataSet();

    da.Fill(ds);

    GridView1.DataSource = ds;

    GridView1.DataBind();

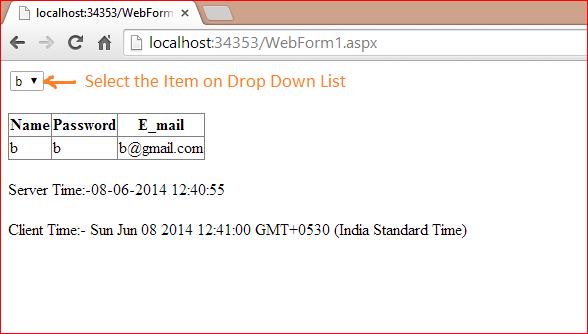
}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

    databind(DropDownList1.SelectedValue.ToString());

}

Output  
  
  
  
And now if the Other Option of Drop Down List is chosen then we can't get the data back. That happens because when I choose the First Option of the Drop Down List the request goes to the server and the page is saved in cached memory so no matter if we choose the other option, To resolve this problem we need to use VaryByParam="\*"and it allows the multiple responses on a single Web Form.  
  
2. Data Caching  
  
Data Caching is basically used for reducing database contention and round-trips, in other words by using Data Caching we store required data in a cache so the Web Server does not send a request to the database for every request so that the database performance will also increased. The data cached is a Full-Featured Cache Engine that enables you to store and retrieve data between multiple HTTP requests and multiple sessions within the same application.  
  
Now here I am explaining data caching with an example.  
  
The remaining steps are the same.  
  
Step 5: Now here I have given 2 web forms with names I have given DataCaching.aspx and DataCaching2.aspx. and in the DataCaching.aspx the following is the code:

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DataCaching.aspx.cs" Inherits="Caching.DataCaching" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

<div>    

User Name:-<asp:TextBox ID="tbUserName" runat="server"></asp:TextBox>

<br />

<br />

    Password:-<asp:TextBox ID="tbpwd" runat="server"></asp:TextBox>

<br />

    <asp:Button ID="Button1" runat="server" onclick="Button1\_Click" Text="Submit" />    

</div>

</form>

</body>

</html>

Code

protected void Button1\_Click(object sender, EventArgs e)

{

    //textbox value is stored in cache

    Cache["UserName"] = tbUserName.Text;

    Cache["Pwd"] = tbpwd.Text;

    Response.Redirect("DataCaching2.aspx");

}

Step 6: Now the following is the code for DataCaching2.aspx:

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DataCaching2.aspx.cs" Inherits="Caching.DataCaching2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

<div>

    User Name:-<asp:TextBox ID="tbUserName" runat="server"></asp:TextBox>

<br />

<br />

    Password:-<asp:TextBox ID="tbpwd" runat="server"></asp:TextBox>

<br />

</div>

</form>

</body>

</html>

Code

protected void Page\_Load(object sender, EventArgs e)

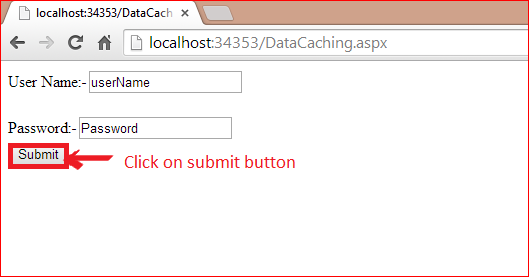
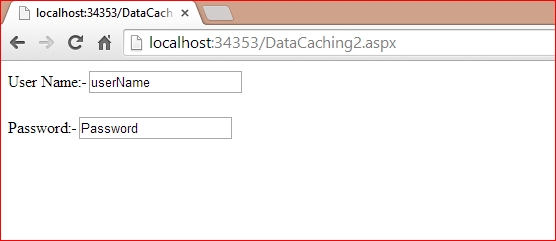
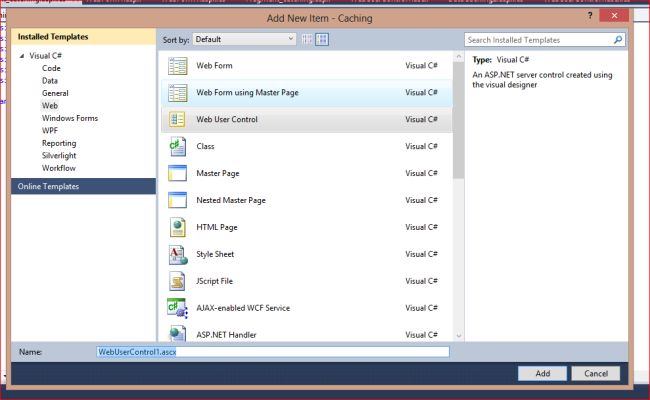
{

    //cache value is assign on the text box

    tbUserName.Text = Cache["UserName"].ToString();

    tbpwd.Text = Cache["Pwd"].ToString();

}

The output is:  
  
  
  
  
  
3 Fragment Caching  
  
Caching of the entire page is not good because some part of the page is common for the entire application. So Fragmentation Catching is used for some portion of the page to be cached and we can do Fragmentation Caching using a User Control.  
  
Now here, I am explaining Fragmentation Caching with an example.  
  
The remaining steps are the same.  
  
Step 4: Now right-click on "Add" > "New Item" > "Web User Control" and add the name.  
  
  
  
Add the code inside the User Control.

<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="WebUserControl1.ascx.cs" Inherits="Caching.WebUserControl1" %>

<%@ OutputCache Duration="30" VaryByParam="None" %>

<div>

<asp:GridView ID="GridView1" runat="server" BackColor="White"

BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px" CellPadding="3"

ForeColor="Black" GridLines="Vertical">

<AlternatingRowStyle BackColor="#CCCCCC" />

<FooterStyle BackColor="#CCCCCC" />

<HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#999999" ForeColor="Black" HorizontalAlign="Center" />

<SelectedRowStyle BackColor="#000099" Font-Bold="True" ForeColor="White" />

<SortedAscendingCellStyle BackColor="#F1F1F1" />

<SortedAscendingHeaderStyle BackColor="#808080" />

<SortedDescendingCellStyle BackColor="#CAC9C9" />

<SortedDescendingHeaderStyle BackColor="#383838" />

</asp:GridView>

<br />

    Server Time:-<asp:Label ID="Label1" runat="server"></asp:Label>

<br />

<br />

    Client Time:-

<script type="text/javascript">

    document.write(Date())

</script>

</div>

    Code:- Write the Code on Code Behind.

protected void Page\_Load(object sender, EventArgs e)

{

    if (!IsPostBack)

    {

          //in the label show the server time

          Label1.Text = DateTime.Now.ToString();    

          //connection string

          SqlConnection con = new SqlConnection(@"server=divs\sqlexpress; Database=Project;uid=sa;password=divya");

          //It's a query

          string query = "select User\_name from registeration";

         //it's a command to be execute in a disconnected mode

         SqlDataAdapter da = new SqlDataAdapter(query, con);

         //create a obj of data set

         DataSet ds = new DataSet();

        //fill the data in dataset

        da.Fill(ds);

        //bind the data in gridview

        GridView1.DataSource = ds;

        GridView1.DataBind();

    }

}

Step 5: Now right-click on the "Add" > "New Item" > "Web From" and add the name of the web form and write the following code:

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Fregment\_catching.aspx.cs" Inherits="Caching.Fregment\_catching" %>

<%@ Register src="WebUserControl1.ascx" tagname="WebUserControl1" tagprefix="uc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

<div>

Server Side Time On Page Level

    <asp:Label ID="Label1" runat="server"></asp:Label>

<br />

Client Side time on Page Level:-

<script type="text/javascript">

    document.write(Date())

</script>    

<br />

<br />

    <uc1:WebUserControl1 ID="WebUserControl11" runat="server" />    

</div>

</form>

</body>

</html>

Code: Write the following code in the Code Behind.

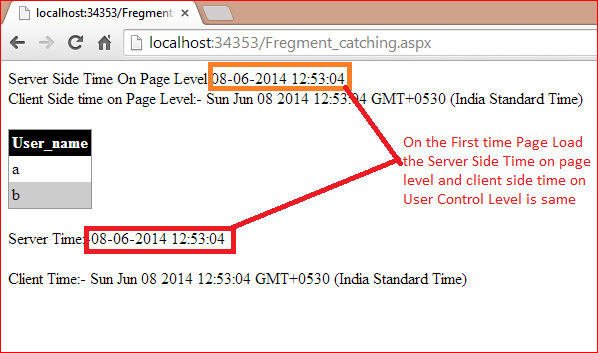
protected void Page\_Load(object sender, EventArgs e)

{

    //show the server time

    Label1.Text = DateTime.Now.ToString();

}

Output  
  
  
  
In this output the Page Side Server time and user Control Side Server Time is the same but if we again refresh the button within 30 seconds then the user control server time is the same but all the time will be changed, that means the particular portion of the page is cached.  
  
And now the main point is if the same User Control is used in a different browser then it will also have the shared property so that the same cached user control is shared in various clients.

<%@ OutputCache Shared="true" Duration="30" VaryByParam="None" %>

Cache Dependency  
  
By using Cache Dependency we can set the dependency on some data and entities. In ASP.NET Caching Dependency are one of the following 5 types:

File-Based Dependency: Generally File-Based Dependency is used when the cached data is dependent on the file. We can set the dependency on multiple files also.

Key-Based Dependency: Items in the Application Cache are stored in key/value pairs. The key dependency allows an item to be dependent on the key of another item in the Application Cache. When the original item is removed, the Item that has the key dependency is also removed. For example you can add a cache an item named reports Valid and then cache several reports that are dependent on the ReportValid Key. When the ReportValid key item is removed, all the dependent cached reports are similarly removed from the cache.

SQL-Based Dependency: Items of the cache are dependent on the row of a table in a SQL Server Database

Aggregate Dependency: Items of the cache are dependent on multiple elements; if any of the dependencies change then the item is removed from the cache.

Custom Dependency: An Item in the cache is configured with the dependency you create in your own code. For example, you can create a custom Web Service Cache Dependency that removes data from the cache when a call to the web service results in a particular value.

We can add the object into the cache and create it using the cache.insert() and cache.add() methods. The cache.insert() method has 4 overload versions whereas cache.add() has no overloads and most of the common properties for those methods are:

Key: It's type is String type and a unique key is used to identify these entries in the cache.

Dependency: It's type is cachedependency and it shows the dependency of cache like file or directory and when changed it should cause this entry to be flushed.

Expires: It's type is DateTime and a fixed date and time after which this cache entry should be flushed.

Sliding Expiration: It's type is TimeSpan and the time between when the object was Last accessed and when the object should be flushed from the cache.

Priority: It's the type CacheItemPriority and it sets the priority of the cache.

OnRemoveCallBack: It's type is CacheItem RemovedCallback and a delegate that can be registered with something.

And we can write the dependency as in the following:  
  
Cache.Insert("Name", strName,new CacheDependency(Server.MapPath("name.txt"),DateTime.Now.AddMinutes(2), TimeSpan.Zero);  
  
Caching Location  
  
In which location caching can be done so ASP.NET has the following 4 caching locations where we can do the cache:

Client Caching: The cache data is stored inside the client's browser on the local disk as a temporary file or in the browser's internal memory. So it reduces the network traffic and the client can be accessed easily but it is totally browser dependent so it is not shareable.

Proxy Caching: It stores the caching information between the client and the web server in a shared location so that all clients can use the same shared data.

Reverse Proxy Caching: The web server has a Proxy Server in the front so that it reduces the number of requests that they receive. This allows the Proxy Server to respond to frequently received requests and only passes other requests to the web server. This is called a reverse proxy and it reduces the number of requests and can easily be accessed but it increases the network traffic because the Proxy Server is the front of the web server.

Web Server Caching: The cache data is stored inside the web server.

Advantage of the Caching

It reduces the traffic of the server because the copy of the data is cached on the server and for the same request the copy of the data is given by a response without hitting the web server.

Reduces bandwidth consumption