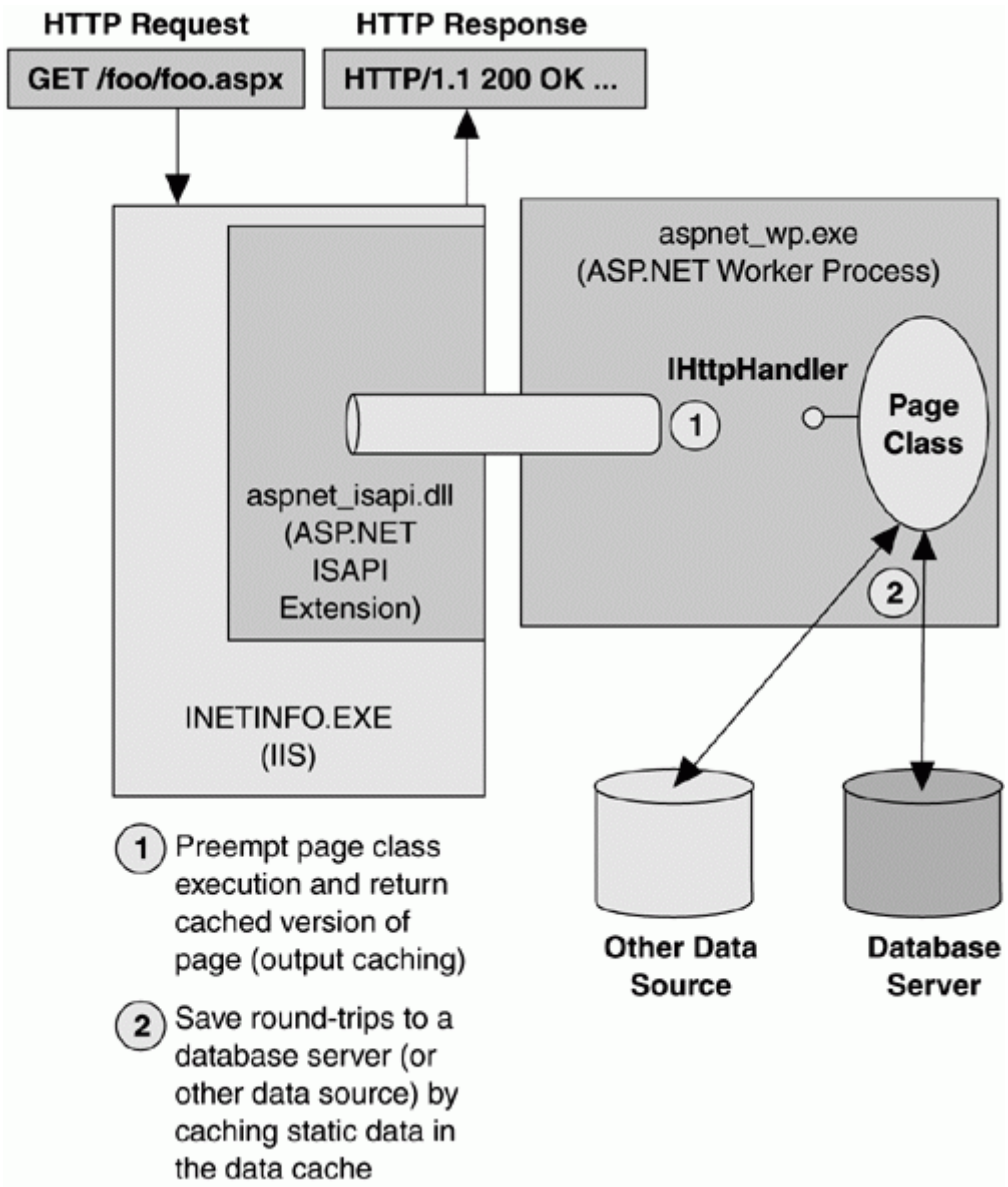
فصل هشتم :

مبحث Caching در ASP.NET

مقدمه :

در حالت عمومی ، Caching ذخیره سازی موقت برای دسترسی سریعتر می باشد. Caching در مکان های مختلفی می تواند انجام شود برای مثال بر روی کامپیوتر کلاینت (Browser caching) ، روی سروری که میان کلاینت و وب سرور قرار گرفته است (Proxy caching) ، و روی خود وب سرور (Page caching or data caching) . هر دو حالت Browser caching و proxy caching ترافیک وب سرور را کاهش می دهند ، زیرا محتویات صفحه به صورت مستقیم از کامپیوتر کلاینت و یا پروکسی تامین می شود ، بنابراین به صورت مستقیم توسط ASP.NET قابل مدیریت نیستند هرچند هنوز متا تگهای ASP کلاسیک نیز اینجا صادق بوده و با اضافه کردن متاتگی مانند CacheControl و Expires می توان نوع Browser caching و Proxy caching را تعیین نمود. حالت های Page caching و Data Caching به صورت مستقیم در ASP.NET قابل مدیریت هستند.

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد



شکل ۱– مبحث Caching در ASP.NET

Output caching :

تولید مجدد صفحاتی که محتویات آنها تقریبا استاتیک است (تغییری در آنها حاصل نمی شود) ، به ازای درخواست هر کلاینت مقرون به صرفه نیست. در عوض صفحات را می توان یکبار تولید نمود و

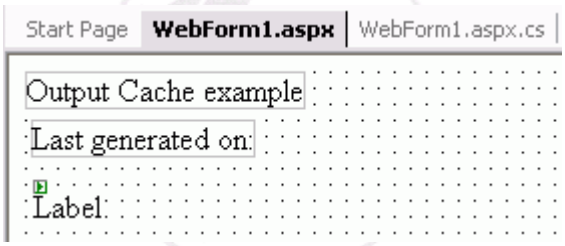
کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

سپس آنها را برای درخواست های بعدی Cache کرد. با استفاده نمودن از OutPutCache برای هر صفحه ی ASP.NET و مشخص کردن مدت زمان Cache شدن آن (به ثانیه) می توان اینکار را انجام داد.

مثال ۱ :

در مثال زیر یک صفحه با استفاده از OutPutCache به مدت یک ساعت از زمان اولین مشاهده ی آن Cache می شود. در این صفحه تاریخ اولین دسترسی به صفحه قابل مشاهده است. تا پایان منقضی شدن مدت Cache این تاریخ تغییری نمی کند.

مطابق شکل زیر یک Label روی فرم قرار دهید.

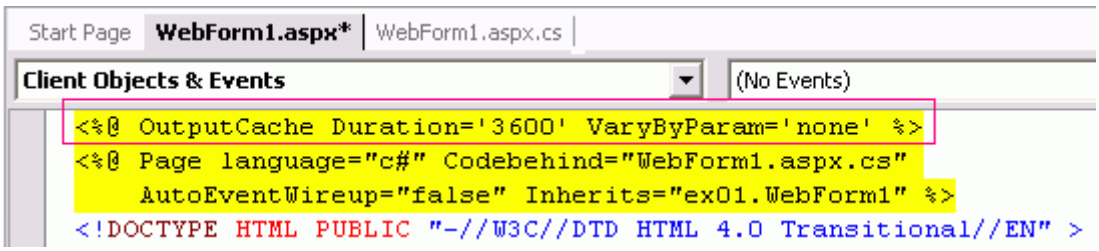


سپس در Page_Load صفحه کد زیر را اضافه نمایید:

```
private void Page_Load(object sender, System.EventArgs e)
{
      lblMsg.Text = DateTime.Now.ToString();
}
```

اکنون به برگه ی HTML صفحه ی aspx مراجعه نموده و کد زیر را به آن اضافه نمایید:

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد



برنامه را اجرا نموده و چند بار صفحه را ریفرش نمایید. مشاهده خواهید کرد که تغییری حاصل نمی شود.

هنگام استفاده از OutPutCache حتما باید حداقل ویژگی های Duration و VaryByParam تنظیم و مشخص گردند. تنظیم VaryByParam به none به این معنا است که یک کپی از صفحه با هرکدام از درخواست های Post ، Head و یا Get حتما Cached می شوند و درخواست های بعدی از Cache خوانده خواهند شد. در جدول زیر لیست کامل ویژگی های قابل اعمال به OutPutCache را می توان ملاحظه کرد.

جدول ۱– **OutPut Cache directive attributes**

ویژگی	مقادیر	توضیح
Duration	عدد	زمانی که صفحه یا کنترل کاربر Cache می شود (به ثانیه).
Location	‘any’ ‘client’ ‘downstream’ ‘server’ ‘none’	هنگامی که any انتخاب شود یعنی تمام حالت های دیگر دربر گرفته شده است. Client به این معنا است که صفحه فقط بر روی کامپیوتر کلاینت ذخیره می شود. Downstream به این معنا است که صفحه روی Downstream server و کلاینت ذخیره می شود. Server به این معنا است که صفحه فقط روی سرور وب ذخیره می شود. None حالت OutPutCaching را غیر فعال می کند.

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

VaryByCustom	'Browser' Custom string	Out put cache بوسیله ی نام مرورگر و نگارش آن تغییر می کند و یا بوسیله ی یک رشته ی سفارشی تولید شده بوسیله ی GetVaryByCustomString
VaryByHeader	'*' Header name	Header های فرستاده شده توسط کلاینت که توسط لیستی رشته ای ، جدا شده توسط سمی کولون مشخص می شوند.
VaryByHeader	'none' '*' Parameter name	لیستی از رشته های جدا شده توسط سمی کولون که نمایشگر مقادیر query string در یک درخواست get و یا متغیری در درخواست post . این مورد الزامی است.
VaryByControl	Control name	لیستی از رشته های جدا شده توسط سمی کولون بیانگر خواص کنترل کاربر بکار گرفته شده که فقط Output cache را تغییر می دهند (تنها به کنترل های کاربر قابل اعمال است.)

در مورد مفاهیم بکار گرفته شده در جدول فوق در ادامه بیشتر صحبت خواهد شد.

جدول ۲– تاثیر ویژگی Location روی OutPutCaching

Value of Location	CacheControl Header	Expires Header	Page Cached on Server
'Any'	public	Yes	Yes
'Client'	private	Yes	No
'Downstream'	public	Yes	No
'Server'	no-cache	No	Yes
'None'	no-cache	No	No

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

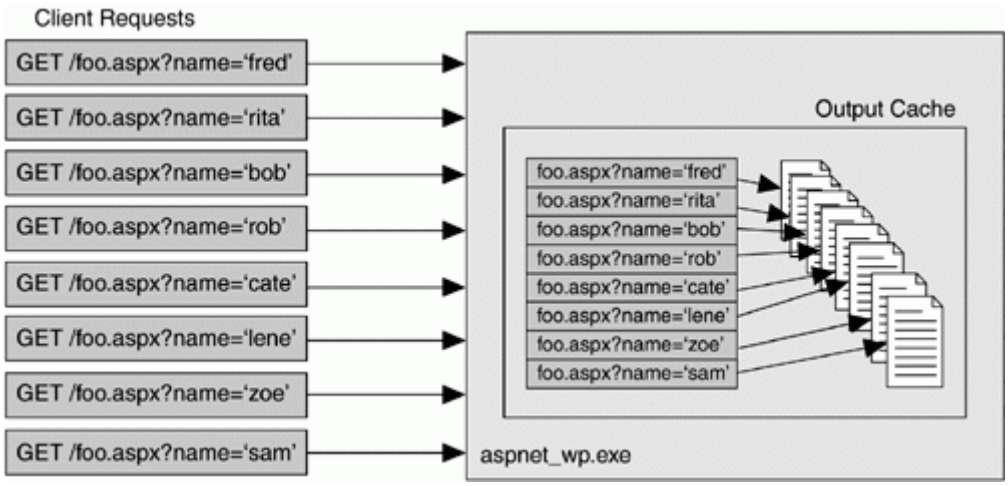
کاشه کردن نگارش های مختلف یک صفحه :

کاربران صفحات را به طرق مختلفی می توانند دریافت کنند: با استفاده از درخواست های get ، head و یا post و یا برای مثال توسط در خواست های get حاوی query string های همراه.

اگر درخواست یک صفحه از طریق get بدون query string ها باشد ساده ترین حالت ممکن بوده و محتویات صفحه هیچگاه تغییر نمی کند. کاشه کردن صفحات با درخواست های post و یا get همراه query string کمی پیچیده می باشند ، زیرا یک نگارش منحصر بفرد به ازای هر درخواست صفحه باید تولید شود.

قبل از تصمیم گیری در مورد استفاده از OutPutCaching یک صفحه ، باید مشخص شود که چند نگارش از یک صفحه باید ذخیره شوند؟ تمام این موارد از طریق ویژگی VaryByParam قابل تنظیم است. اگر مقدار این ویژگی برابر none قرار داده شود ، تنها یک نگارش از هر صفحه در output cache قرار داده می شود. در این حالت اگر کاربری یک صفحه را با روش get به همراه querystring ها درخواست نماید ، Output cache ، این query string ها را ندید خواهد گرفت و تنها یک صفحه ی تکراری کاشه شده را نمایش می دهد.

اگر مقدار VaryByParam مساوی ‘*’ قرار داده شود ، به ازای هر درخواست get و یا post متفاوت یک صفحه ی منحصربفرد کاشه خواهد شد. این روش بسیار ناکارآمد بوده و باید با احتیات به کار گرفته شود ، زیرا منابع زیادی از سرور را هدر خواهد داد.



شکل ۲ – کاشه کردن کپی های چندگانه از یک صفحه

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

جدول ۳- مقادیر VaryByParam

VaryByParam Values	
VaryByParam Value	**Description**
'none'	One version of page cached (only raw GET or HEAD)
'*'	N versions of page cached based on query string and/or POST body
v1	N versions of page cached based on value of v1 variable in query string or POST body
v1, v2	N versions of page cached based on value of v1 and v2 variables in query string or POST body

ویژگی VaryByParamList همچنین می تواند به نام و یا لیستی از نامها و یا به querystring و یا متغیرهای درخواست Post تنظیم شود. هر چند این موارد نیز خیلی کم بکار برده می شوند ، زیرا متغیرهای querystring و یا post ، هنگامی بکار برده می شوند که می خواهیم یک صفحه را رندر کنیم.

ویژگی VaryByHeader ، هنگامیکه یک Header string و یا مجموعه ای از آنها از یک کلاینت به کلاینت دیگر تغییر می کند ، نگارش های مختلفی از یک صفحه را کاشه می کند. (به صورت ضمنی در مورد اغلب کنترل های ASP.NET اینکار رخ می دهد). برای مثال اگر شما قسمتی از صفحه را بر مبنای Accept-language header فرستاده شده از طرف کلاینت ، رندر کنید ، باید اطمینان حاصل کرد که صفحات کاشه شده به ازای هر زبان متفاوت است.

مثال ۲

یک label روی فرم قرار دهید و سپس به سورس HTML صفحه عبارت زیر را اضافه نمایید:

UserLanguages Property | **WebForm1.aspx** | WebForm1.aspx.cs

Client Objects & Events | (No Events)

```
<%@ OutputCache Location='any'
                VaryByParam='none'
                Duration='120'
                VaryByHeader='Accept-Language' %>
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs"
    AutoEventWireup="false" Inherits="ex02.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
```

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

اکنون کد زیر را به Page_Load صفحه اضافه کنید:

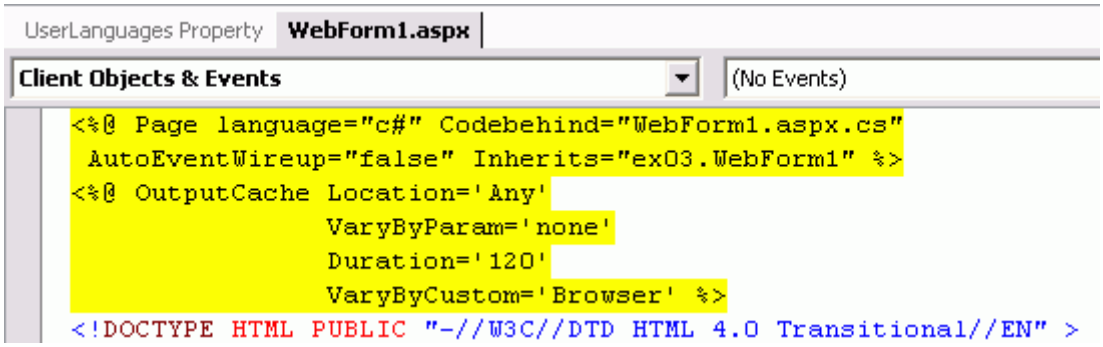
```
private void Page_Load(object sender, System.EventArgs e)
{
      if (!IsPostBack)
      {
            switch( Request.UserLanguages[0])
            {
                  case  "fr":
                        lblMsg.Text = "Bonjour!  Comment allez-vous?";
                        break;
                  case  "de":
                        lblMsg.Text = "Guten Tag!  Wie geht's?";
                        break;
                  default:
                        lblMsg.Text = "Hello!  How are you?";
                        break;
            }
      }

      Response.Write(DateTime.Now.ToString());

}
```

ویژگی VaryByCustom امکان کاشه کردن صفحات را بر مبنای نوع مرورگر و یا ورژن آن و یا هر شرط دیگری میسر می کند. برای مثال اگر شما می دانید که صفحه در مرورگرهای مختلف ممکن است به صورت های مختلفی رندر شود ، لازم است تا براساس مرورگرهای مختلف ، کاشه کردن صورت گیرد. برای مثال اگر VaryByCustom را معادل Browser قرار دهید ، به ازای هر نوع مرورگر دسترسی پیدا کرده به صفحه ، یک نمونه ی منحصر بفرد از صفحه کاشه می شود . (لازم به ذکر است که در ASP.NET اغلب کنترل های سرور وب بر اساس نوع مرورگر به صورت هوشمند ظاهر نهایی خود را ارائه می دهند).

مثال ۳

یک کنترل تقویم را روی صفحه قرار دهید. سپس کد زیر را به سورس HTML صفحه اضافه نمایید:

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد



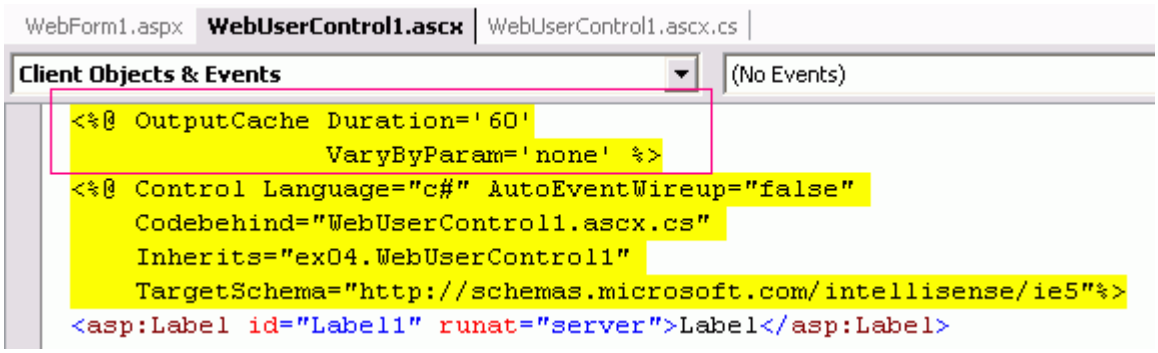
Page fragment caching :

علاوه بر صفحاتی که درون یک وب سایت به ندرت محتویات آنها تغییر می کند ، ممکن است قسمتی از یک صفحه نیز اینچنین باشد. برای مثال منوها ، Header ها و footer های سایت و امثال اینها به ندرت دچار تغییر و تحول می شوند. این قسمت ها موارد ایده آلی برای مبحث Caching به شمار می روند. خوشبختانه ASP.NET مکانیزمی به نام Page fragment caching را برای این منظور ارائه داده است. برای استفاده از این قابلیت ابتدا باید قسمت استاتیک صفحه را که می خواهید Cache کنید به صورت یک کنترل کاربر درآورید. در فایل مربوط به کنترل با اضافه کردن Outputcache و ویژگی Duration و VaryByParam می توان Cache کردن را فعال نمود. به این ترتیب صفحاتی دارای قسمت های استاتیک با سرعت بیشتری در دفعات بعدی مشاهده بارگذاری خواهند شد.

مثال ۴

یک کنترل کاربر به پروژه اضافه نموده و سپس روی آن یک لیبل قرار دهید.
سپس به سورس HTML کنترل عبارت زیر را اضافه نمایید:

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد



سپس در رخداد Page_Load این کنترل عبارت زیر را اضافه نمایید:

```
private void Page_Load(object sender, System.EventArgs e)
{
          Label1.Text = "User control generated at " +
      DateTime.Now.ToString();
}
```

اکنون این کنترل کاربر را با Darg کردن به فرم اصلی برنامه که دارای یک لیبل نیز می باشد اضافه کنید.
سپس در رخداد Page_Load این فرم کد زیر را بنویسید:

```
private void Page_Load(object sender, System.EventArgs e)
{
          Label1.Text = "Page generated at " +
           DateTime.Now.ToString();

}
```

برنامه را اجرا نموده و صفحه را چندبار ریفرش کنید.

Data caching :

کاشه کردن اطلاعات به شدت کارآیی یک برنامه را با کاهش اتصالات به دیتابیس و رفت و برگشت های مربوطه ، افزایش می دهد. بهتر است این مورد را با یک مثال بررسی کنیم.

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

مثال ۵:

در مثال زیر کاشه سازی DataView بررسی شده است. در اولین باری که صفحه توسط کاربر مشاهده می شود ، یک کوئری روی دیتابیس صورت گرفته و dataview ساخته می شود و در ادامه کاشه می گردد. در بازدیدهای بعدی DataView صرفا از کاشه خوانده می شود و زمان کوئری گرفتن مجدد از دیتابیس صرفه جویی می گردد.

یک لیست باکس را روی صفحه قرار دهید. سپس کد زیر را در Page_Load فرم برنامه بنویسید:

```
private void Page_Load(object sender, System.EventArgs e)
{
      // Look in the data cache first
      DataView dv = new DataView();

      dv = (DataView)Cache["EmployeesDataView"];

      if (dv == null) // wasn't there
      {
            SqlConnection conn  =
new SqlConnection("server=localhost;uid=sa;pwd=;database=Northwind");
            SqlDataAdapter da =
                  new SqlDataAdapter("select * from Employees", conn);
            DataSet ds = new DataSet();
            da.Fill(ds, "Employees");
            dv = ds.Tables["Employees"].DefaultView;
            dv.AllowEdit = false;
            dv.AllowDelete = false;
            dv.AllowNew = false;
            // Save employees table in cache
            Cache["EmployeesDataView"] = dv;
            conn.Close();
      }
      else
      {
            Response.Write("<h2>Loaded from data cache!</h2>");
      }

      ListBox1.DataSource = dv;
      ListBox1.DataTextField = "LastName";
      ListBox1.DataValueField = "FirstName";
      DataBind();

}
```

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

چند بار پس از اجرا ، صفحه را ریفرش کنید تا نتیجه حاصل گردد.

Data cache در سطح application مهیا است اما دوتفاوت عمده با مبحث application state که پیش تر در مورد آن بحث شد ، دارد. اگر داده ای درون کاشه قرار گیرد هیچ تضمینی وجود ندارد که هنگام دسترسی بعدی تغییر نکرده باشد. کلاس cache از System.Threading.ReadWriterLock برای اطمینان حاصل کردن از اینکه در آن واحد تنها یک thread می تواند محتویات کاشه را تغییر دهد استفاده می نماید. این روش تنها در سطح سرور انجام می شود و کلاینت ها نمی توانند توسط آن انجام قفل کردن کاشه و تغییر داده ها را انجام دهند. مطلب دیگری را که باید بخاطر داشت این است که کاشه کردن داده ها صرفا روی یک ماشین و یا پروسه انجام می شود و اگر از چندین سرور استفاده شود انجام همزمانی صورت نخواهد گرفت.

بنابراین بهتر است که Data caching به صورت readonly انجام شود تا از تغییرات ناخواسته توسط سایر کاربران جلوگیری گردد. به مثال مهم زیر دقت فرمایید:

مثال ۶ :

یک لیست باکس را روی صفحه قرار دهید. سپس کد زیر را در Page_Load فرم برنامه بنویسید:

```
private void Page_Load(object sender, System.EventArgs e)
{
      // Look in the data cache first
      ArrayList al = (ArrayList) Cache["MyList"];

      if (al == null)  // wasn't there
      {
            al = new ArrayList();
            // Save ArrayList in cache
            Cache["MyList"] = al;
      }

      // Manipulate the ArrayList by adding the time this
      // request was made (bad! may be accessed concurrently!)
      al.Add(DateTime.Now.ToString());
      ListBox1.DataSource = al;
      DataBind();

}
```

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

در مثال فوق یک نمونه از کلاس ArrayList در کاشه ذخیره می شود. این نمونه هر بار که صفحه مشاهده می شود با اضافه کردن زمان دسترسی به آن ، تغییر می یابد. این مساله خطرناک می باشد زیرا احتمال دارد که چندین در خواست دقیقا در یک زمان رخ دهد و کلاس ArrayList کلاسی ThreadSafe نمی باشد.

مشاهده ی داده های cache شده :

مثال ۷ :

```
private void Page_Load(object sender, System.EventArgs e)
{

      // display all of the items stored in the ASP.NET cache
      Response.Write("<b>Data cache contains:</b><br/>");
      Response.Write("<table>");
      Response.Write("<tr><td><b>Key</b></td>");
      Response.Write("<td><b>Value</b></td></tr>");

      foreach(DictionaryEntry objItem in Cache)
      {
            Response.Write("<tr><td>");

            Response.Write(objItem.Key.ToString());
            Response.Write("</td><td>");
            Response.Write(objItem.Value.ToString());
            Response.Write("</td></tr>");
      }

      Response.Write("</table>");
}
```

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

ویژگی های مرتبط با data caching :

تاکنون ملاحظه کرده اید که data caching شبیه application state object می باشد و تفاوت های آن در مورد طول عمر و قابلیت آپدیت آن است. هر گونه کش کردن داده ها سبب ایجاد یک نمونه ی مخفی از کلاس cacheEntry می شود که تنها ویژگی های زیر آن قابل دسترسی و تنظیم می باشد.

جدول ۴– خواص کلاس **CacheEntry**

خاصیت	نوع	توضیح
key	**string**	کلیدی است منحصربفرد بکار گرفته شده برای مشخص نمودن این ورودی در کاشه
Dependency	**Cache dependency**	وابستگی ورودی به کاشه به فایل یا دایرکتوری و یا سایر داده های ورودی به کاشه .هنگامیکه تغییر نماید ، ورودی flushed می شود.
expire	**DateTime**	مدت زمانی که پس از آن داده های ورودی به کاشه flushed می شوند.
SlidingExpiration	**TimeSpan**	زمانی بین هنگام آخرین دسترسی به شیء و هنگامیکه شیء باید از کاشه flushed شود.
Priority	**Cache Item priority**	در مقایسه با سایر داده های موجود در کاشه چقدر اهمیت دارد که این داده در کاشه نگهداری شود (هنگامیکه تصمیم گرفته می شود چگونه باید اشیاء کاشه در هنگام Scavenging حذف شوند)
OnRemoveCallBack	**CacheItemRemovedCallback**	Delegate ایی است که می تواند با داده ی ورودی به cache رجیستر شده و هنگام حذف داده خبر رسانی نماید.

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

هنگامیکه ایندکسر پیش فرض data cache برای اضافه کردن آیتم ها بکار برده می شود ، مقادیر کلاس Cache Entry به مقادیر پیش فرض تنظیم می شوند.

این مورد بدین معنا است که زمان expire شدن به infinite تنظیم شده است و SlidingExpiration=0 و cacheItemPriority=normal و CacheItamRemoveCallback=null است. اساسا تا زمانی که عملیات Scavenging رخ نداده است ، شیء شما در کاشه باقی خواهد ماند (عموما بدلیل مصرف حافظه ی زیادی) و همچنین شما آنرا صریحا حذف نکرده اید.

اگر می خواهید کنترل بیشتری روی خواص Cache entry ایجاد شده برای شیء کاشه داشته باشید ، می توانید از ورژن های Overloaded مختلف متد Insert استفاده نمایید. پرکاربرد ترین ورژن Insert تمام خواص Cache entry را بعنوان پارامتر ، بعلاوه شیء ایی که باید کاشه شود دریافت کرده و سپس آنرا به سازنده ی کلاس CacheEntry می فرستد. برای مثال کد نشان داده شده در زیر ، رشته ها را به data cache اضافه نموده و زمان expire شدن آنها را به نیمه ی شب اول دسامبر سال ۲۰۰۳ تنظیم می نماید.

مثال ۸ :

```
object obj  = //retrieve obj to place in cache somehow

DateTime dt  = new DateTime(2001, 12, 31, 23, 59, 59);

Cache.Insert("MyVal", obj, null, dt,
          Cache.NoSlidingExpiration,
          CacheItemPriority.Default,
          null);

// Cache.Insert :
// Meaning of parameters:
// Insert(key, object, dependencies, absolute expiration,
//        sliding expiration,
//        priority,
//        callback delegate)
```

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

حذف کردن شیء Cache :

برای حذف کردن داده های موجود در کاشه به چندین روش می توان عمل کرد. می توان آنرا به صورت صریح با بکار گیری متد Cache.Remove حذف کرد و یا حذف خواهد شد چون طول عمر آن expire شده است و یا می تواند به صورت غیرصریح از کاشه حذف شود تا حافظه ی مصرف شده آزاد گردد (Scavenging). شما روی دو مورد اول ، کنترل مستقیم ندارید. در مورد scavenging کنترل مستقیمی وجود نداشته و بیشتر بر اساس priority آیتم های کاشه شده ، حذف صورت می گیرد. در جدول زیر مقادیر CaheItemPriority توضیح داده شده اند ، هر چند می توان تنظیمات را طوری انجام داد که عملیات scavenging داده ها را حذف نکند. اما عاقلانه تر این است که کار را به الگوریتم های هوشمند Scavenging واگذار نماییم.

جدول – مقادیر **cache Item priority**

مقدار cache Item priority	توضیح
AboveNormal	در زمان Scavenging احتمال حذف شدن آنها از حالت Normal کمتر است.
BelowNormal	در زمان Scavenging احتمال حذف شدن آنها از حالت Normal بیشتر است.
Default	معادل Normal است.
High	در زمان Scavenging احتمال حذف شدن آنها بسیار کم است.
Low	در زمان Scavenging احتمال حذف شدن آنها بسیار زیاد است.
Normal	در طی عملیات Scavenging پس از حذف حالتهای Low و BelowNormal ، از کاشه حذف خواهند شد.
NotRemovable	هرگز از کاشه به صورت غیرصریح حذف نمی شوند.

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

طول عمر شیء کاشه شده :

هرگاه که داده ای به کاشه اضافه می شود باید طول عمر آنرا مشخص نمود و یا مقدار پیش فرض (infinite) را پذیرفت. تعیین این امر بسیار مهم است زیرا روی صحت داده های دریافت شده مؤثر بوده و در صورت عدم تنظیم دقیق آن با داده هایی بیات و کهنه (stale) مواجه خواهید شد. تعیین طول عمر داده های قرار گرفته در کاشه ، بیشتر به نوع آن داده وابسته است. داده ای در صورتیکه یک فایل روی سیستم تغییر نماید و یا یک داده ی وارد شده به کاشه غیر معتبر شود ، ممکن است غیر معتبر شود. یا ممکن است پس از مدتی غیر معتبر شود (absolute expiration) و یا ممکن است بخواهید حافظه ی زیادی را که اشغال کرده است آزاد نمایید (sliding expiration items) . در آخر می توان یک callback delegate را رجیستر نمود تا از حذف یک آیتم از کاشه اطلاعاتی را در اختیار قرار داده و مقتضی با آن عملیاتی را انجام داد.

اکثر موارد فوق در مثال زیر بررسی شده اند:

مثال ۹ :

در فایل کد بیهایند global.asax کد زیر را بنویسید:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Web;
using System.Web.SessionState;
using System.Web.Caching;

namespace ex09
{
      /// <summary>
      /// Summary description for Global.
      /// </summary>
      public class Global : System.Web.HttpApplication
      {

            // Generic function to load contents of 'pi.txt' file
        // and store in cache
            public void LoadPi()
            {
                  System.IO.StreamReader sr    =
      new System.IO.StreamReader(HttpContext.Current.Server.MapPath("pi.txt"));
                  string pi = sr.ReadToEnd();
```

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

```
                CacheDependency piDep   =
        new CacheDependency(HttpContext.Current.Server.MapPath("pi.txt"));
                Context.Cache.Add("pi", pi, piDep,
                      Cache.NoAbsoluteExpiration,
                      Cache.NoSlidingExpiration,
                      CacheItemPriority.Default,
                      new  CacheItemRemovedCallback( OnRemovePi));
            }

        public void OnRemovePi(string key , object val , CacheItemRemovedReason r )
            {

                LoadPi();
            }

            public Global()
            {
                InitializeComponent();
            }

            protected void Application_Start(Object sender, EventArgs e)
            {
                LoadPi();

            }

            #region Web Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
            }
            #endregion
        }
}
```

سپس روی فرم اصلی برنامه یک تکست باکس مالتی لاین قرار دهید:

```
            private void Page_Load(object sender, System.EventArgs e)
            {
                if (Cache["pi"] ==null)
                {
                    //Refresh pi in app
                    Global dd= new  Global();
                    dd.LoadPi();
                }
                else
                {
                    TextBox1.Text = Cache["pi"].ToString() ;
                }


            }
```

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

موارد مهمی که باید هنگام Data caching در نظر گرفت:

۱- داده های قابل آپدیت را نباید در سطح application بکار گرفت و آنها را در Cache قرار داد. چون data cache از تغییرات این داده ها توسط سایر کاربران محافظت نمی کند.
۲- مؤثر بودن کاشه کردن داده ها به دو فاکتور وابسته است : داده ها اغلب چگونه و چندبار مورد دسترسی واقع می شوند و اغلب چندبار تغییر می کنند؟ اگر داده ها با هر بار دسترسی کلاینت به آنها ، تغییر داده شود ، کاشه کردن آنها اتلاف منابع است. اگر دادها به ندرت تغییر می کنند اما تقریبا مورد استفاده قرار نمی گیرند ، کاشه کردن آنها نیز اتلاف منابع است. کاشه کردن داده هایی که از دیتابیس خوانده می شوند اغلب سودمند است.
۳- با استفاده از cache dependency می توان از تازه بودن اطلاعات اطمینان حاصل کرد. به این صورت اگر داده های موجود در فایل تغییر کنند ، کاشه در درخواست بعدی آپدیت می شود.
۴- سعی کنید از duration های کمتری در هنگام کاشه کردن داده ها استفاده نمایید تا از اینکه داده ها کهنه نیستند اطمینان حاصل شود.

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد

تمرین :

۱- در مورد HttpCachePolicy Class تحقیق نمایید.

۲- در مورد Response.Cache تحقیق نمایید.

۳- از عبارت زیر در یک مثال استفاده نمایید:

```
<%@ OutputCache  Duration='120'  Location='Client'  VaryByParam='none' %>
```

۴- در مورد نحوه ی استفاده از GetVaryByCustomString در فایل global.asax تحقیق نمایید.

کلیه حقوق این جزوه آموزشی متعلق به سایت آموزش الکترونیکی پرشیا میباشد