بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

ذخیره سازی داده ها / ASP.NET -Data Caching

مدرس : مهندس افشین رفوآ

ذخیره سازی داده ها / ASP.NET -Data Caching

Caching ها عبارتند از ذخیره سازی داده/اطلاعات پرکاربرد (اطلاعاتی که به طور مکرر مورد استفاده قرار می گیرند) در حافظه. تکنیک مزبور برای این پیاده می شود که در صورت نیاز پیدا کردن به اطلاعات پرکاربرد دیگر نیازی نباشد که داده ها مجددا توسط برنامه (application) ایجاد شود بلکه بتوان آن ها به طور مستقیم از حافظه بازیابی کرد.

از آن جایی که کنترل ها و صفحات دربردارنده ی آن ها در ASP.NET به صورت پویا ایجاد می گردند (تولید می گردند)، ذخیره سازی موقت اطلاعات در حافظه به افزایش سرعت و کارایی کمک چشمیگری می کند.

Caching به خصوص در رابطه با بازیابی اطلاعات مربوط به تراکنش ها بسیار مفید و کارامد محسوب می شود، زیرا که این عملیات از لحاظ مدت زمانی که طول می کشد تا response ارائه گردد بسیار سنگین و زمان بر محسوب می شوند.

Caching دادهایی که به طور مکرر مورد استفاده قرار می گیرند را در وسیله های در دسترس مانند RAM رایانه جای گذاری و ذخیره می کند. ASP.NET runtime دربردارنده ی یک نما یا نقشه ی جفت کلید-مقدار از اشیا CLR می باشد که Cache (حافظه ی نهان) خوانده می شود. cache در application مقیم بوده و از طریق کلاس HttpContext و System.Web.UI.Page قابل دسترسی می باشد.

از جهاتی، caching مشابه ذخیره سازی state object ها است. با این حال ذخیره سازی اطلاعات در state object ها قطعی است به این معنا که شما از ذخیره ی اطلاعات درون state object ها مطمئن هستید اما caching (ذخیره سازی موقت) کاملا قطعی نیست.

در موارد زیر فهرست شده، داده قابل دسترسی نمی باشد:

در صورتی که چرخه ی حیات (lifetime) آن تمام (منقضی) شده باشد

در صورتی که application حافظه ی خود را آزاد کند

در صورت رخ ندادن caching به هر دلیلی

می توانید با استفاده از indexer (شاخص زن) به آیتم های ذخیره شده در cache (حافظه ی نهان) دسترسی پیدا کرده و نیز چرخه ی حیات اشیا را مدیریت کنید. همچنین این امکان برای شما وجود دارد که بین اشیا ذخیره شده در حافظه ی نهان (cached objects) و منبع فیزیکی آن ها لینک ایجاد کنید.

ذخیره سازی داده ها به صورت موقت در ASP.NET

ASP.NET از انواع مختلف caching پشتیبانی می کند که زیر برای شما شرح داده شده:

Output Caching: یک نسخه ی عینی (copy) از صفحات یا قسمتی از صفحات نهایی و render شده ی HTML که به سرویس گیرنده (client) ارسال شده را ذخیره می کند. هنگامی که کلاینت بعدی صفحه را درخواست می کند، بجای بازسازی (تولید مجدد) صفحه، یک نسخه ی ذخیره شده از صفحه فرستاده می شود و از این رو در زمان صرفه جویی قابل توجهی صورت می گیرد.

Data Caching: به معنای ذخیره ی موقت اطلاعات از یک منبع داده است. مادام اینکه cache منقضی نشده، داده ی درخواست شده از حافظه ی نهان (cache) بازیابی می شود. هنگامی که cache منقضی شده یا مدت آن کاملا سر می آید، داده های جدید از منبع داده گرفته شده و حافظه ی نهان مجددا از داده پر می شود.

Object Caching: به ذخیره سازی اشیا در یک صفحه گفته می شود، از جمله ی آن می توان به کنترل های data-bound اشاره کرد. داده هایی که به طور موقت ذخیره می گردند (cached data)، در حافظه ی سرور نگه داری می شوند.

Class Caching: صفحات یا سرویس های وب هنگامی که برای اولین بار اجرا می گردند داخل assembly ،درون یک کلاس Page کامپایل شده، سپس اسمبلی به طور موقت داخل سرور ذخیره می شود. دفعه ی بعدی که صفحه یا سرویس مورد نظر درخواست می شود، به assembly ذخیره شده در سرور ارجاع داده می شود. به محض اینکه source code تغییر می کند، CLR اسمبلی را مجددا کامپایل می کند.

Configuration Caching: اطلاعات پیکربندی مربوط به کل application داخل فایل پیکربندی (configuration file) ذخیره می شود. Configuration caching در واقع اطلاعات پیکربندی را داخل حافظه ی server ذخیره می کند.

در این مبحث تنها به تشریح output caching، data caching و object caching می پردازیم.

Output Caching

rendering (ارائه ی ماشینی) یک صفحه ممکن است شامل فرایند های سنگین و پیچیده ای همچون دسترسی به پایگاه داده، render کردن کنترل های مرکب و پیچیده باشد. output caching با ذخیره سازی موقت اطلاعات داخل حافظه از دانلود یا درخواست مجدد اطلاعات از سرور (درخواست تکراری برای دریافت اطلاعات به سرور) جلوگیری می کند. حتی امکان ذخیره ی کل یک صفحه نیز وجود دارد.

OutputCache directive مسئول output caching می باشد. directive ذکر شده قابلیت output caching را فعال سازی کرده و به برنامه نویس امکان می دهد عملکرد آن را تا حدی کنترل کند.

دستور نگارش (syntax) directive به شرح زیر می باشد:

```
<%@ OutputCache Duration="15" VaryByParam="None" %>
```

Directive فوق را زیر page directive درج کنید. با این کار به محیط می فهمانید که باید صفحه را به مدت 15 ثانیه نگه دارد (ذخیره کند). با استفاده از event handler زیر برای رخداد page load می توان بررسی کرد آیا صفحه به درستی ذخیره می شود یا خیر.

```
protected void Page_Load(object sender, EventArgs e)
{
    Thread.Sleep(10000);
    Response.Write("This page was generated and cache at:" +
    DateTime.Now.ToString());
}
```

متد Thread.Sleep() نخ (thread) جاری را به مدت زمان مشخصی (برای چند هزارم ثانیه) به حالت تعلیق در می آورد. در این مثال نخ به مدت 10 ثانیه متوقف می شود، بنابراین هنگامی که صفحه برای اولین بار بارگذاری می شود، چیزی حدود 10 ثانیه بارگذاری آن طول می کشد. اما دفعه ی بعدی که صفحه را بروز رسانی (refresh) می کنید، صفحه بلافاصله (بدون اتلاف وقت) بالا می آید، زیرا که صفحه بدون نیاز به بارگذاری مجدد و مستقیما از سرور بازیابی می شود.

OutputCache directive دارای خصیصه های (attribute) زیر می باشد که در مدیریت و کنترل عملکرد output caching کمک شایان توجهی می کند:

خصیصه	مقدار	شرح
DiskCacheable	true/false	تعیین می کند آیا خروجی می تواند روی دیسک به صورت موقت ذخیره شود یا خیر.
NoStore	true/false	این خصیصه تعیین می کند آیا "no store" مربوطه به cache control header ارسال شود یا خیر.
CacheProfile	String name	اسم cache profile ای که در فایل web.config ذخیره می شود را تعیین می کند.
VaryByParam	None * Param- name	لیستی از رشته های تعیین حدود (delimit) شده توسط نقطه ویرگول که مقادیر query string را در Get request یا متغیر را در Post request تعیین می کند.
VaryByHeader	* Header names	به شما امکان می دهد محتوا را بر اساس یک یا چند سرآیند (header) که توسط مرورگر ارسال می شود تغییر دهید.
VaryByCustom	Browser Custom string	به ASP.NET دستور می دهد خروجی (output) را بر اساس اسم مرور گر و ورژن آن و یا یک رشته ی سفارشی تغییر دهد.
Location	Any	Any: صفحه ممکن است هر جایی ذخیره شده

	Client	باشد. Client: محتویات ذخیره شده در مرورگر باقی می ماند. Downstream: محتویات ذخیره شده هم در downstream و هم در سرور ذخیره می گردند. Server: محتویات ذخیره شده در حافظه ی پنهان (cached content) فقط بر روی سرور ذخیره می شوند. None: caching را به طور کلی غیر فعال می کند.
Duration	Number	مدت زمانی (تعداد ثانیه ها) که صفحه یا کنترل مورد نظر به طور موقت ذخیره می شود.

حال یک کنترل textbox و button به مثال قبلی اضافه می کنیم. در مرحله ی بعد event handler زیر را برای کنترل button بکار می بریم.

```
protected void btnmagic_Click(object sender, EventArgs e)
{
    Response.Write("<br><br>");
    Response.Write("<h2> Hello, " + this.txtname.Text + "</h2>");
}
```

OutputCache directive را اصلاح کنید:

```
<%@ OutputCache Duration="60" VaryByParam="txtname" %>
```

به هنگام اجرای صفحه، ASP.NET صفحه را بر مبنای اسمی که داخل textbox وارد شده بود موقتا ذخیره (cache) می کند.

Data Caching

مهمترین جنبه ی data caching ذخیره سازی کنترل های داده (data source control) در حافظه ی نهان می باشد. همان طور که پیشتر ذکر شد کنترل های داده (data source controls) نمایشگر داده های موجود در منبع داده هستند، از جمله ی آن ها می توان به فایل XML و یا یک پایگاه داده اشاره کرد. کنترل های ذکر شده از کلاس انتزاعی DataSourceControl مشتق شده و همچنین دارای خاصیت های (property) به ارث برده ی زیر می باشند که در پیاده سازی caching کمک می کنند:

CacheDuration - این خاصیت مدت زمانی (تعداد ثانیه) که منبع داده (data source) اطلاعات را به طور موقت ذخیره (cache) می کند، تعیین می کند.

CacheExpirationPolicy – عملکرد یا رفتار حافظه ی نهان را به هنگام سر آمدن (منقضی شدن) مدت ذخیره ی داده های موجود در cache را تعریف می کند.

CacheKeyDependency – این خاصیت یک کلید برای کنترل ها تعریف می کند که در صورت پاک شدن آن به صورت خودکار محتویات حافظه ی نهان را نیز پاک (منقضی) می کند.

EnableCaching – تعیین می کند آیا داده ها به طور موقت (در حافظه ی نهان) ذخیره شوند یا خیر.

مثال:

جهت تشریح data caching به ایجاد یک وب سایت آزمایشی می پردازیم.

ابتدا یک وب سایت جدید ایجاد کرده، سپس یک web form به آن اضافه کنید. همچنین یک کنترل SqlDataSource که دارای اتصال به پایگاه داده باشد، اضافه کنید.

حال یک label به صفحه اضافه کنید که در آن response time صفحه نمایش داده می شود.

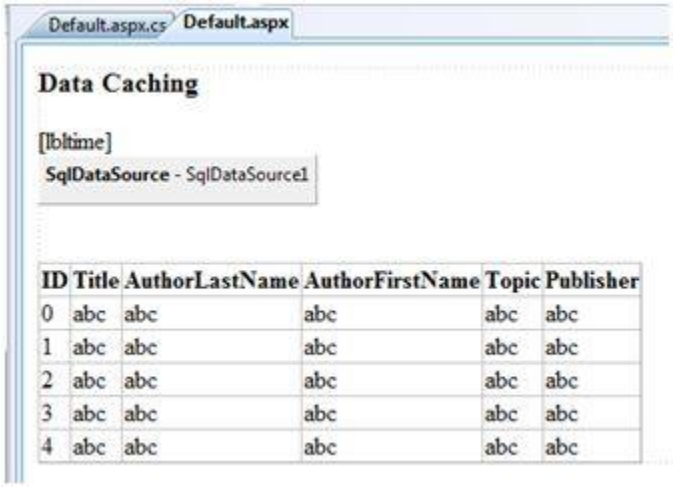
```
<%@ OutputCache Duration="60" VaryByParam="txtname" %>
```

یک event handler به رخداد page load اضافه کنید:

```
protected void Page_Load(object sender, EventArgs e)
{
    lbltime.Text = String.Format("Page posted at: {0}",DateTime.Now.ToLongTimeString());
```

```
        }
```

نتیجه:



هنگامی که برای اولین بار صفحه را اجرا می کنید، هیچ چیز متفاوتی رخ نمی دهد. کنترل label نشان می دهد که با هر بار refresh کردن صفحه، صفحه مجددا بار گذاری شده و نیز زمانی که در کنترل label به نمایش گذاشته شده بود تغییر می کند.

در مرحله ی بعد خصیصه ی EnableCaching را روی مقدار true تنظیم کرده و خصیصه ی Cacheduration را نیز '60' مقداردهی کنید. قابلیت caching پیاده سازی شده و با گذر هر 60 ثانیه cache منقضی (expire) شده و محتویات آن پاک می شود.

timestamp با هر بار refresh تغییر می کند، اما اگر داده های موجود در جدول را طی این 60 ثانیه دستکاری کنید، timestamp پیش از منقضی و پاک شدن محتویات cache اصلا نمایش داده نمی شود.

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings: ASPDotNetStepByStepConnectionString %>"
ProviderName="<%$ ConnectionStrings: ASPDotNetStepByStepConnectionString.ProviderName %>"
SelectCommand="SELECT * FROM [DotNetReferences]"
EnableCaching="true" CacheDuration="60"></asp:SqlDataSource>
```

Object Caching

Object caching در مقایسه با دیگر تکنیک های ذخیره سازی موقت داده، انعطاف پذیری بیشتری را فراهم می کند.

می توان با بهره گیری از تکنیک object caching هر شئی را داخل حافظه ی نهان قرار داد. شئ می تواند از هر نوعی باشد – یک نوع داده (data type)، یک کنترل وب (web control)، یک کلاس، یک dataset object و غیره . . . آیتم مورد نظر را می توان تنها با تخصیص اسم کلید (key name)، به حافظه ی نهان اضافه کنید:

```
Cache["key"] = item;
```

ASP.NET برای درج یک شئ به داخل cache متد Insert() را ارئه داده است. ان تابع دارای چهار نسخه ی overload شده است که زیر مشاهده می کنید:

Overload	شرح
Cache.Insert((key, value);	آیتم را به همراه کلید (که برای ارجاع به شئ بکار می ورد) و مقدار (شئی که داخل کش درج می شود) داخل cache درج کرده و سیاست های پیش فرض انقضا و اولویت را برای آن ها اعمال می کند.
Cache.Insert(key, value, dependencies);	آیتم را داخل کش به همراه کلید، مقدار، اولویت و انقضا پیش فرض و همچنین یک اسم CacheDependency که به دیگر فایل ها یا آیتم ها لینک می شود درج می کند. در صورت تغییر dependency، آن شئ دیگر نامعتبر محسوب شده و از حافظه ی نهان حذف می گردد.
Cache.Insert(key, value, dependencies, absoluteExpiration, slidingExpiration);	نشانگر سیاست انقضا/expiration policy (اینکه چه زمانی شئ از حافظه ی نهان پاک شود) می باشد
Cache.Insert(key, value, dependencies, absoluteExpiration, slidingExpiration, priority, onRemoveCallback);	این نسخه ی overload شده به همراه پارامترهای ورودی آن به شما امکان می دهد اولویت برای آیتم مورد نظر cache و همچنین یک delegate تخصیص دهید که به هنگام پاک شدن آیتم مورد نظر یک متد را فراخوانی می کند.

Sliding expiration را زمانی بکار می بریم که می خواهیم آیتمی را که طی مدت زمان مشخصی مورد استفاده قرار نگرفته از حافظه ی نهان یا cache حذف کنیم. تکه کد زیر آیتمی را که Sliding expiration آن 10 دقیقه تنظیم شده و هیچ dependency ای ندارد ذخیره می کند.

```
Cache.Insert("my_item", obj, null, DateTime.MaxValue, TimeSpan.FromMinutes(10));
```

مثال:

یک صفحه ایجاد کنید که فقط دارای یک کنترل label و button باشد. اکنون کد زیر را در رخداد page load وارد کنید:

```
protected void Page_Load(object sender, EventArgs e)
        {
            if (this.IsPostBack)
            {
                lblinfo.Text += "Page Posted Back.<br/>";
            }
            else
            {
                lblinfo.Text += "page Created.<br/>";
            }
            if (Cache["testitem"] == null)
            {
                lblinfo.Text += "Creating test item.<br/>";
                DateTime testItem = DateTime.Now;
                lblinfo.Text += "Storing test item in cache ";
                lblinfo.Text += "for 30 seconds.<br/>";
                Cache.Insert("testitem", testItem, null,
                DateTime.Now.AddSeconds(30), TimeSpan.Zero);
            }
            else
            {
                lblinfo.Text += "Retrieving test item.<br/>";
                DateTime testItem = (DateTime)Cache["testitem"];
                lblinfo.Text += "Test item is: " + testItem.ToString();
                lblinfo.Text += "<br/>";
            }
            lblinfo.Text += "<br/>";
        }
```

اولین باری که صفحه بارگذاری می شود، پیغام زیر را ارائه می دهد:

```
Page Created.
Creating test item.
Storing test item in cache for 30 seconds.
```

اگر 30 ثانیه بعد مجددا روی دکمه کلیک کنید، صفحه postback (بازگردانده) می شود اما کنترل label اطلاعات خود را از حافظه ی نهان بازیابی می کند:

```
Page Posted Back.
Retrieving test item.
```

```
Test item is: 14-07-2010 01:25:04
```