مرجع آموزشی
ASP.NET 4

(جلد2)

مرجع آموزشی
ASP.NET 4
(جلد2)

مهندس محمد اسماعیلی هدی

انتشارات پندار پارس

سرشناسه : اسماعیلی هدی، محمد، 1356 -
عنوان و نام پدیدآور : مرجع آموزشی ASP.NET 4.0/ تالیف و ترجمه محمد اسماعیلی‌هدی.
مشخصات نشر : تهران: پندار پارس، 1390.
مشخصات ظاهری : ج2: 632 ص.:مصور،جدول، نمودار.
شابک : 165000 ریال:978-964-2989-62-1 دوره: 978-964-2989-63-8
وضعیت فهرست نویسی : فیپا
موضوع : صفحه‌های سرور فعال
موضوع : ای‌اس‌پی. (پروتکل شبکه کامپیوتری)
موضوع : وب-- سایت‌ها-- طراحی
رده بندی کنگره : 5105/8885TK/ ص7الف5 1390
رده بندی دیویی : 005/276
شماره کتابشناسی ملی : 8426722

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره 14، واحد 16 **www.pendarepars.com**

تلفن: 66572335 - تلفکس: 66926578همراه: 09122452348 **info@pendarepars.com**

نام کتاب : **مرجع آموزشی ASP.NET 4.0 (جلد2)**

ناشر : **انتشارات پندار پارس ناشر همکار: مانلی**

تالیف : **محمد اسماعیلی هدی**

چاپ اول : **پاییز 90**

شمارگان : **1000 نسخه**

طرح جلد : **محمد اسماعیلی هدی**

لیتوگرافی : **ترام‌سنج**

چاپ، صحافی : **صالحان، خیام**

قیمت : **16500 تومان شابک : 978-964-2989-62-1 دوره: 978-964-2989-63-8**

تقدیم به

پدر و مادر عزیزم

که وجودشان در زندگی برای من امید و انگیزه است.

فهرست مطالب:

فصل یازدهم

کاشه کردن[1] اطلاعات

منظور از کاشه کردن، ذخیره یک کپی از اطلاعات در حافظه است. برای مثال، ممکن است نتایج حاصل از یک کوئری را در کاشه نگهداری کنید تا در درخواست‌های بعدی کاربر، نیاز به دسترسی به بانک اطلاعاتی نداشته باشید. همچنین، کاشه باعث دسترسی سریع به داده‌های مورد نظر خواهد شد؛ زیرا در این حالت، داده‌ها از حافظه بازیابی می‌شوند. یک نکته بسیار جالب در مورد کاشه، کارایی[2] بالا و مقیاس‌پذیری[3] زیاد آن به طور همزمان است. منظور از کارایی بالا، دسترسی بسیار سریع به اطلاعات است. همچنین، مقیاس‌پذیری در کاشه ارتقاء پیدا کرده است؛ زیرا شما نیاز به عملیات بانک اطلاعاتی زیادی ندارید.

البته، ذخیره کردن اطلاعات در حافظه، همیشه مفید نیست؛ زیرا حافظه سرور محدود است و پر کردن فضای حافظه سرور، باعث کاهش شدید کارایی آن خواهد شد[4]. به همین دلیل، فقط اطلاعاتی در کاشه قرار می‌گیرد که در آینده، به کرات به آنها نیاز پیدا خواهید کرد. همچنین، هنگامی که حافظه پر می‌شود، اطلاعات کم کاربردتر به طور انتخابی از حافظه کاشه حذف می‌شود. این کار باعث می‌شود تا کارایی برنامه‌های وب حفظ شود.

در ASP.NET می‌توانید یک شیء دلخواه، یک صفحه HTML و یا حتی قسمتی از آن را در حافظه کاشه ذخیره کنید. همچنین می‌توانید زمان انقضای داده‌های خود را مشخص کرده و یا رابطه بین آنها را تعیین نمایید. با تعیین این رابطه می‌توانید هنگام ویرایش داده‌ها، اطلاعات موجود در کاشه را به طور اتوماتیک حذف نمایید.

---

[1] Caching
[2] Performance
[3] Scalability

[4] در صورت پر شدن حافظه RAM، اطلاعات به دیسک سخت منتقل می‌شوند.

11,1. مفهوم کاشه کردن در ASP.NET

برخی از توسعه‌دهندگان صفحات وب، ممکن است کاشه کردن را امری زاید تلقی کنند؛ اما در واقع چنین نیست. با کاشه کردن اطلاعات ضروری در یک زمان کوتاه، حتی ممکن است کارایی سیستم تا ده برابر افزایش یابد!

در واقع، ASP.NET از دو نوع کاشه کردن پشتیبانی می‌کند. برنامه‌های وب، باید از هر دو آنها استفاده نمایند؛ زیرا، هر کدام از آنها مکمل دیگری هستند:

- **کاشه کردن خروجی**[1]: این نوع کاشه کردن، ساده‌ترین نوع آنها است. در این حالت، کپی صفحه HTML تولید شده، در کاشه ذخیره شده و هنگام درخواست‌های بعدی، به کاربران دیگر ارسال خواهد شد. بنابراین، نیازی به تولید مجدد صفحه نخواهد بود. این کار، باعث صرفه‌جویی در زمان و افزایش کارایی سرور خواهد شد.
- **کاشه کردن داده‌ها**: این نوع کاشه، به صورت کدنویسی دستی انجام می‌شود. در این حالت، فقط داده‌هایی کاشه می‌شوند که تولید مجدد آنها نیاز به زمان داشته باشد؛ مانند DataSet بازیابی شده از یک بانک اطلاعاتی. سایر صفحات وب می‌توانند با بررسی محتویات کاشه، از اطلاعات آن استفاده کنند. آیتم‌های موجود در کاشه می‌توانند به طور اتوماتیک منقضی شوند.

دو نوع خاص از کاشه، از مدل‌های فوق ناشی می‌شوند که عبارتند از:

- **کاشه کردن قطعه‌ای**: این نوع کاشه، نوع خاصی از کاشه کردن خروجی است. در این حالت، فقط قسمتی از صفحه وب در کاشه ذخیره می‌شود. در این صورت، کنترل‌های کاربر در کاشه ذخیره شده و در درخواست‌های بعدی مورد استفاده قرار می‌گیرند. به عبارت دیگر، تمام کدها و رویدادهای صفحه، به جز کنترل‌های کاربری خاص، اجرا خواهند شد.
- **کاشه کردن منبع داده**: این نوع کاشه کردن، در کنترل‌های منبع داده‌ای مانند SqlDataSource، ObjectDataSource و XmlDataSource ساخته می‌شوند. این نوع از کاشه کردن، از مدل کاشه کردن داده‌ها استفاده می‌کند. تنها تفاوت آن است که در این حالت، نیازی به راه‌اندازی صریح فرآیند ندارید. فقط باید خصوصیات مورد نظر را تنظیم نمایید.

در این فصل، با گزینه‌های مختلف کاشه و انواع آن آشنا خواهید شد.

---

[1] Output Caching

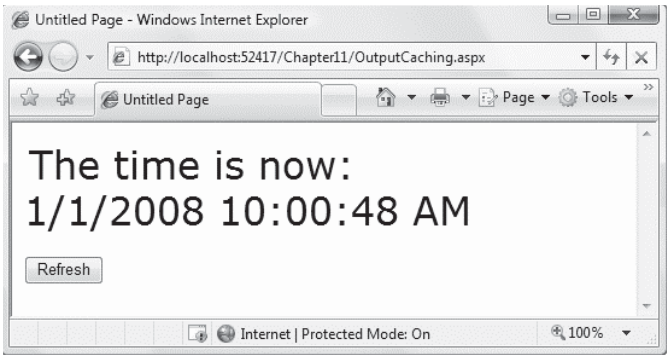
11,2. کاشه کردن خروجی

با کاشه کردن خروجی، صفحه HTML تولید شده در حافظه کاشه ذخیره می‌شود. اگر همان صفحه دوباره درخواست شود، کنترل‌ها ایجاد نشده و کدها اجرا نخواهند شد و در عوض، صفحه موجود در کاشه، مجددا ارسال خواهد شد. به این ترتیب، کارایی برنامه وب افزایش می‌یابد.

■ **نکته**: یک صفحه ASP.NET ممکن است از منابع استاتیک دیگری مانند تصاویر استفاده کند که به وسیله ASP.NET رسیدگی نمی‌شود. در این حالت، نگران کاشه کردن آنها نباشید؛ زیرا، IIS این آیتم‌ها را به صورت لازم در کاشه ذخیره خواهد کرد.

11,2,1. کاشه خروجی تعریفی

برای مشاهده عملی کاشه کردن خروجی، می‌توانید یک صفحه ساده مانند زیر ایجاد کنید که زمان جاری را نمایش می‌دهد.



کد لازم برای ایجاد این صفحه، بسیار ساده است. فقط باید زمان جاری را در یک Label قرار دهید. این کد، به صورت زیر، در داخل رویداد Page.Load قرار می‌گیرد:

```
protected void Page_Load(Object sender, EventArgs e)
{
    lblDate.Text = "The time is now:<br />";
    lblDate.Text += DateTime.Now.ToString();
}
```

شما دو روش برای اضافه کردن این صفحه به حافظه کاشه در پیش دارید. روش متداول، اضافه کردن تگ زیر در بالای فایل aspx. است. این کد باید در پایین راهنمای[1] Page قرار گیرد:

```
<%@ OutputCache Duration="20" VaryByParam="None" %>
```

در این مثال، خاصیت Duration به ASP.NET می‌فهماند که باید صفحه را به اندازه 20 ثانیه در کاشه نگهداری کند. خاصیت VaryByParam در بخش بعدی توضیح داده خواهد شد.

هنگامی که صفحه را برای اولین بار اجرا می‌کنید، زمان جاری نشان داده خواهد شد. اگر در فاصله کوتاهی دکمه Refresh را فشار دهید، صفحه به‌هنگام‌سازی نخواهد شد؛ زیرا ASP.NET، مجددا صفحه موجود در کاشه را نمایش خواهد داد. بعد از گذشت 20 ثانیه، کپی موجود در کاشه از بین رفته و در هنگام درخواست مجدد، زمان جاری درست نمایش داده خواهد شد.

این زمان کوتاه، در سایت‌هایی که دارای درخواست‌های زیادی است، بسیار مهم خواهد بود. برای مثال، با این کار می‌توانید دسترسی به بانک اطلاعاتی را به سه بار در دقیقه محدود نمایید. به این ترتیب، به ازای هر درخواست، نیاز به اتصال به بانک اطلاعاتی نخواهد بود.

در صورت کمبود حافظه، ممکن است قبل از زمان 20 ثانیه، صفحه وب از کاشه حذف شود. به این ترتیب، به راحتی می‌توانید از کاشه استفاده نمایید؛ بدون آن که نگران کمبود حافظه باشید.

■ **نکته**: وقتی دوباره یک صفحه موجود در کاشه را کامپایل می‌کنید، ASP.NET آن صفحه را از کاشه حذف خواهد کرد. این کار باعث می‌شود صفحه قدیمی به کاربر نشان داده نشود. در هنگام تست برنامه، بهتر است کاشه کردن را غیرفعال نمایید. در غیر این صورت، هنگام دیباگ کردن صفحه، دچار مشکل خواهید شد؛ زیرا، با وجود کپی آن در کاشه، کد شما اجرا نخواهد شد.

11,2,2. *کاشه کردن و رشته کوئری*[2]

یکی از نکات مهم در کاشه کردن، انتخاب یک صفحه برای انتقال آن به حافظه کاشه است. برای نمونه، هنگامی که در ساخت یک صفحه وب از اطلاعات Session استفاده می‌شود، نمی‌توانید تمام آن صفحه را در کاشه ذخیره

---

[1] Directive
[2] Query String

کنید. با این حال، می‌توانید قسمتی از آن را کاشه نمایید. مثال دیگر، زمانی است که در ساخت یک صفحه، از اطلاعات صفحه دیگری استفاده می‌شود. معمولا، این اطلاعات به وسیله رشته کوئری دریافت می‌شود.

در مثال اخیر، خاصیت VaryByParam را به None تنظیم کنید تا ASP.NET، فقط یک کپی از صفحه را در کاشه ذخیره نماید. اگر درخواست صفحه، دارای رشته کوئری در URL باشد، ASP.NET دوباره همان خروجی موجود در کاشه را ارسال خواهد کرد. توجه کنید که پارامترهای مورد نظر شما می‌تواند در نوار آدرس اضافه شود. همان طور که می‌دانید، پارامترها بعد از علامت ? قرار می‌گیرند. همچنین، هر پارامتر با علامت & از پارامتر دیگر جدا می‌شود.

با این مثال، ممکن است تصور کنید که کاشه کردن، برای صفحات دارای رشته کوئری مناسب نیست؛ اما ASP.NET دارای گزینه دیگری هم است. با تنظیم خاصیت VaryByParam به مقدار * می‌توانید به ASP.NET بفهمانید که صفحه وب از رشته کوئری استفاده می‌کند. به این ترتیب، چند کپی مختلف از صفحه در کاشه قرار می‌گیرد:

```
<%@ OutputCache Duration="20" VaryByParam="*" %>
```

هنگامی که صفحه وب را درخواست می‌کنید، اطلاعات رشته کوئری بررسی می‌شود. اگر رشته شما با درخواست‌های قبلی مطابقت داشته باشد، کپی کاشه شده آن صفحه به شما ارسال می‌شود. در غیر این صورت، یک کپی جدید از صفحه ایجاد شده و به حافظه کاشه اضافه می‌گردد. برای درک بهتر این موضوع، به مراحل زیر توجه کنید:

1. یک صفحه را بدون رشته کوئری درخواست کنید تا یک کپی از صفحه (A) را دریافت کنید.
2. صفحه را با پارامتر ProductID=1 درخواست نمایید. یک کپی از صفحه (B) را دریافت خواهید کرد.
3. یک بار دیگر، صفحه را با پارامتر ProductID=2 درخواست نمایید. به این ترتیب، کپی دیگری از صفحه (C) را دریافت می‌کنید.
4. دوباره صفحه را با پارامتر ProductID=1 درخواست نمایید. اگر کاشه B منقضی نشده باشد، همان نسخه به کاربر ارسال خواهد شد.
5. در نهایت، صفحه را بدون رشته کوئری درخواست کنید. اگر نسخه A در کاشه موجود باشد، به کاربر ارسال خواهد شد؛ در غیر این صورت، نسخه جدیدی تولید شده و به حافظه کاشه کپی می‌شود.

خودتان می‌توانید این موضوع را تست کنید. برای تست این برنامه، بهتر است زمان انقضای آن را افزایش دهید.

11,2,3. *کاشه کردن با پارامترهای رشته کوئری خاص*

تنظیم VaryByParam="*" اجازه می‌دهد تا صفحات دینامیک را به کاشه منتقل کنید. این روش، برای صفحاتی مانند محصول (Product) بسیار مناسب است؛ زیرا، شناسه محصول به صورت رشته کوئری به این صفحه ارسال می‌شود. به این ترتیب، به ازای هر محصول، یک صفحه در حافظه کاشه قرار می‌گیرد. با این حال، باید زمان کاشه را به چند دقیقه یا بیشتر تنظیم نمایید.

البته، این تکنیک دارای مشکلاتی نیز هست. صفحاتی که دارای محدوده پارامترهای بسیار متنوعی هستند، با این روش به خوبی کار نمی‌کنند. برای مثال، صفحاتی که چند عدد را گرفته و نتایج محاسبه را به کاربر نشان می‌دهند، با این روش، قابل پیاده‌سازی نیستند. دلیل این مساله، مشخص است: احتمال استفاده مجدد از این صفحات بسیار پایین می‌باشد. ضمن آن که با اشغال بی‌رویه حافظه ممکن است باعث حذف سایر اطلاعات موجود در کاشه شوند.

در بسیاری از اوقات، تنظیم VaryByParam به مقدار * موثر نیست؛ بلکه بهتر است متغیر موجود در رشته کوئری را صریحا معرفی نمایید:

```
<%@ OutputCache Duration="20" VaryByParam="ProductID" %>
```

در این حالت، ASP.NET در رشته کوئری به دنبال ProductID خواهد گشت. بنابراین، درخواست‌هایی با پارامترهای مختلف ProductID، به صورت جداگانه کاشه خواهند شد. همچنین، از سایر پارامترهای رشته کوئری چشم‌پوشی خواهد شد. این کار، مخصوصا زمانی مفید است که رشته کوئری دارای پارامترهای بلااستفاده باشد.

همچنین می‌توان از چند پارامتر مختلف استفاده کرد. در این صورت، پارامترها با علامت ";" از هم جدا می‌شوند:

```
<%@ OutputCache Duration="20" VaryByParam="ProductID;CurrencyType" %>
```

در این حالت، ترکیبات متنوع‌تری از پارامترها در حافظه کاشه ذخیره خواهند شد.

■ **نکته**: درحالت کلی، کاشه کردن خروجی زمانی مفید است که بر مبنای داده‌های موجود در سرور (مانند داده‌های بانک اطلاعاتی) باشد. اگر کاشه کردن بر اساس داده‌های کاربر (مانند داده‌های Session یا کوکی) باشد، سرور کارایی خود را از دست خواهد داد. کاشه کردن خروجی، با رویدادهای مربوط به فرم نیز به درستی کار نخواهد کرد؛ زیرا در این حالت، از رویدادها صرف‌نظر شده و صفحات استاتیک به کاربر ارسال می‌شوند.

11,2,4. کنترل کاشه سفارشی

استفاده از پارامترهای متنوع رشته کوئری، تنها راه کاشه کردن نسخه‌های گوناگون از یک صفحه نیست. می‌توان یک روال شخصی ایجاد کرد تا تصمیم بگیرد صفحه کاشه شود و یا آن که یک نسخه جاری به کاربر ارسال گردد. این کد، اطلاعات را بررسی کرده و یک رشته مناسب برمی‌گرداند. سپس، ASP.NET از این رشته برای پیاده‌سازی کاشه استفاده می‌کند. اگر کد شما یک رشته را برای چند درخواست مختلف تولید کند، در این صورت، یک صفحه مناسب در کاشه به کاربر ارسال خواهد شد؛ در غیر این صورت، صفحه جدیدی تولید شده و در کاشه ذخیره می‌گردد.

یکی از روش‌هایی که در کاشه کردن سفارشی استفاده می‌شود، نوع مرورگر است. در این صورت، هر مرورگر مانند Firefox یا IE، صفحات بهینه‌سازی شده خود را دریافت خواهند کرد. برای این کار، ابتدا راهنمای OutputCache را در بالای صفحه‌ای که کاشه خواهد شد، اضافه نمایید. در این حالت، ویژگی VaryByCustom، نوع کاشه سفارشی را مشخص خواهد کرد[1]. در مثال زیر، از نام browser استفاده شده است؛ زیرا، صفحه را بر اساس نوع مرورگر کلاینت، کاشه می‌کند:

```
<%@ OutputCache Duration="10" VaryByParam="None" VaryByCustom="browser" %>
```

سپس، روالی ایجاد کنید که یک رشته سفارشی تولید می‌کند. این روال، باید در فایل global.asax قرار گیرد:

```
public override string GetVaryByCustomString(
  HttpContext context, string arg)
{
    // Check for the requested type of caching.
    if (arg == "browser")
    {
        // Determine the current browser.
        string browserName;
        browserName = Context.Request.Browser.Browser;
        browserName += Context.Request.Browser.MajorVersion.ToString();

        // Indicate that this string should be used to vary caching.
        return browserName;
    }
    else
    {
        return base.GetVaryByCustomString(context, arg);
    }
}
```

[1] این نام، اختیاری است.

تابع ()GetVaryByCustomString، مقدار VaryByCustom را به صورت یک آرگومان پاس می‌کند. به این ترتیب می‌توانید تابعی ایجاد کنید که انواع کاشه سفارشی را پیاده‌سازی می‌کند. هر نوع کاشه کردن، از یک نام VaryByCustom متفاوت (مانند Browser، BrowserVersion یا DayOfWeek) استفاده می‌کند. تابع ()GetVaryByCustomString شما، نام VaryByCustom را بررسی کرده و یک رشته مناسب برمی‌گرداند.

اصولا، تابع ()GetVaryByCustomString، مبتنی بر مرورگر ساخته شده بود. به این ترتیب، در متد قبلی نیازی به نوشتن کد ندارید؛ بلکه، متد ()GetVaryByCustomString رشته مورد نیاز را بر اساس نوع مرورگر و نسخه آن تولید خواهد کرد. اگر بخواهید منطق آن را عوض کنید، باید متد ()GetVaryByCustomString را مانند مثال قبل، بارگذاری[1] نمایید.

راهنمای OutputCache دارای خاصیت سومی به نام VaryByHeader نیز هست که اجازه می‌دهد تا نسخه‌های مختلفی از یک صفحه را بر اساس مقدار هدر HTTP آن ذخیره نمایید. در اینجا می‌توانید از یک یا چند هدر استفاده کنید که با علامت ";" از هم جدا می‌شوند. این تکنیک بیشتر در سایت‌های چند زبانه استفاده می‌شود که در آن، نسخه‌های مختلف صفحه بر اساس زبان مرورگر کاشه می‌شوند:

```
<%@ OutputCache Duration="20" VaryByParam="None"
    VaryByHeader="Accept-Language" %>
```

11,2,5. کاشه کردن با کلاس HttpCachePolicy

استفاده از راهنمای OutputCache دارای اهمیت بالایی است؛ زیرا دستورات کاشه را از سایر کدهای برنامه جدا می‌کند. همچنین اجازه می‌دهد تا خاصیت‌های مختلف را در یک خط تنظیم نمایید.

با این حال، انتخاب دیگری هم دارید. شما می‌توانید کدی بنویسید که از خاصیت درون‌ساخت Response.Cache استفاده می‌کند. این خاصیت، یک نمونه از کلاس System.Web.HttpCachePolicy را ارائه می‌کند که به برنامه‌نویس اجازه می‌دهد تا کاشه کردن را برای صفحه جاری فعال نماید. به این ترتیب، با کدنویسی می‌توان تصمیم گرفت که یک صفحه باید در کاشه ذخیره شود یا خیر.

در مثال زیر، صفحه مربوط به نمایش تاریخ، دوباره‌نویسی شده است. در اینجا، کاشه کردن در اولین بارگذاری صفحه فعال می‌شود. فعال کردن کاشه، با استفاده از متد ()SetCacheability انجام می‌شود. به این ترتیب، صفحه وب، در سرور و کلاینت‌هایی که می‌توانند از نسخه کاشه شده استفاده کنند، ذخیره خواهد شد. متد

[1] Override

()SetExpires، زمان انقضای صفحه را مشخص می‌کند. در اینجا، تاریخ انقضا به زمان جاری بعلاوه 60 ثانیه تنظیم شده است.

```
protected void Page_Load(Object sender, EventArgs e)
{
    // Cache this page on the server.
    Response.Cache.SetCacheability(HttpCacheability.Public);

    // Use the cached copy of this page for the next 60 seconds.
    Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));

    // This additional line ensures that the browser can't
    // invalidate the page when the user clicks the Refresh button
    // (which some rogue browsers attempt to do).
    Response.Cache.SetValidUntilExpires(true);

    lblDate.Text = "The time is now:<br />" + DateTime.Now.ToString();
}
```

از دیدگاه طراحی، استفاده از کدنویسی برای کاشه کردن ملموس نیست. در عین حال، قرار دادن کدهای کاشه در بین کدهای دیگر، باعث شلوغی و آشفتگی در صفحه می‌شود. به خاطر داشته باشید که کدهای موجود در رویداد Page.Load، فقط زمانی اجرا می‌شوند که صفحه مورد نظر در کاشه نباشد.

■ **نکته**: باید مطمئن شوید که از خاصیت Response.Cache صفحه استفاده می‌کنید نه از خاصیت Page.Cache. از خاصیت Page.Cache برای کاشه کردن خروجی استفاده نمی‌شود؛ بلکه، دسترسی شما به کاشه داده را فراهم می‌کند. این موضوع در قسمت "کاشه کردن داده‌ها" مورد بررسی قرار می‌گیرد.

11,2,6. *جانشینی Post-Cache و کاشه کردن قطعه‌ای*

در برخی شرایط، ممکن است متوجه شوید که نمی‌توانید تمام قسمت‌های یک صفحه را در کاشه ذخیره نمایید؛ اما دوست دارید حداقل قسمتی از آن را در کاشه نگهداری کنید. برای انجام این کار، دو راه در پیش رو دارید:

کاشه کردن قطعه‌ای: در این حالت، فقط محتویاتی را مشخص می‌کنید که قصد دارید آن را کاشه نمایید. برای این کار، آن را در یک کنترل کاربری قرار داده و سپس خروجی آن را در کاشه ذخیره می‌کنید.

جانشینی Post-Cache: در این حالت، فقط محتویات دینامیکی را مشخص می‌کنید که نمی‌خواهید اطلاعات آنها در کاشه ذخیره شود. محتویات این قسمت‌ها با استفاده از کنترل Substitution تعیین می‌گردد.

با این که پیاده‌سازی کاشه کردن قطعه‌ای، ساده‌تر از روش دوم است؛ اما انتخاب یکی از این دو روش، به حجم داده‌ها بستگی دارد. اگر حجم داده‌ها کم باشد، روش کاشه کردن قطعه‌ای بهتر عمل می‌کند. برعکس، اگر حجم داده‌های دینامیک کم باشد، روش جانشینی Post-Cache راحت‌تر است. به هر حال، هر دو روش کارایی یکسانی دارند.

11,2,7. *کاشه کردن قطعه‌ای*

برای پیاده‌سازی این روش، نیاز به ایجاد یک کنترل کاربری[1] دارید تا قسمت مورد نظر برای کاشه شدن را پوشش دهد. سپس باید راهنمای OutputCache را به کنترل کاربری اضافه نمایید. نتیجه کار، در کاشه قرار گرفتن محدوده مورد نظر است. جزئیات مربوط به کنترل‌های کاربری، در فصل 15 مورد بحث قرار خواهند گرفت.

کاشه کردن قطعه‌ای، بسیار شبیه کاشه کردن صفحه وب است. در اینجا، یک مشکل وجود دارد. اگر صفحه شما، نسخه کاشه شده کنترل کاربری را دریافت کند، نمی‌تواند با آن تعامل داشته باشد. برای مثال، اگر کنترل کاربری دارای یک سری خصوصیت باشد، نمی‌توان آنها را با کدنویسی تغییر داد.

11,2,8. *جانشینی Post-Cache*

عموما جانشینی Post-Cache، با یک متد به نام ()WriteSubstitution کار می‌کند که به کلاس HttpResponse اضافه شده است. این متد، یک پارامتر ورودی دریافت می‌کند که به یک متد Callback اشاره می‌نماید. این متد Callback در کلاس Page پیاده‌سازی شده است و محتویات قسمت مورد نظر از صفحه را برمی‌گرداند.

در اینجا یک نکته مهم وجود دارد. وقتی صفحه ASP.NET، قسمت کاشه شده را دریافت می‌کند، به طور اتوماتیک، متد Callback را اجرا کرده و محتوای دینامیک را به دست می‌آورد. سپس آن را به صفحه کاشه شده اضافه می‌کند. حتی اگر صفحه شما هنوز کاشه نشده باشد (مانند اولین باری که صفحه تولید می‌شود)، باز هم متد Callback اجرا خواهد شد.

متدی که محتویات دینامیک را تولید می‌کند، باید از نوع استاتیک باشد. این متد، یک شیء HttpContext را می‌گیرد که حاوی درخواست جاری است. سپس یک رشته با HTML جدید برمی‌گرداند. مثال زیر، یک تاریخ به حالت پررنگ (Bold) را برمی‌گرداند:

---

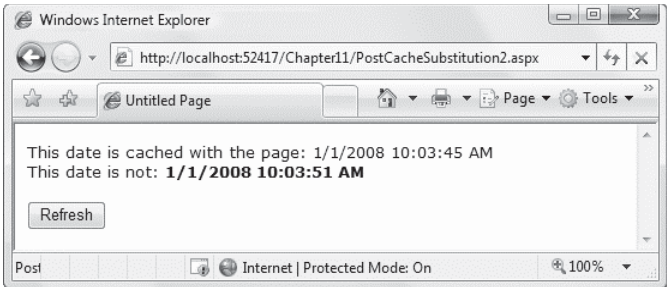
[1] User Control

```
private static string GetDate(HttpContext context)
{
    return "<b>" + DateTime.Now.ToString() + "</b>";
}
```

برای گرفتن این رشته و نمایش آن در صفحه، باید از متد ()Response.WriteSubstitution استفاده نمایید:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("This date is cached with the page: ");
    Response.Write(DateTime.Now.ToString() + "<br />");
    Response.Write("This date is not: ");
    Response.WriteSubstitution(new HttpResponseSubstitutionCallback(GetDate));
}
```

حال، حتی اگر کاشه کردن را با راهنمای OutputCache به این صفحه اعمال کنید، تاریخ دوم به صورت به‌هنگام شده[1]، نشان داده خواهد شد. شکل زیر این موضوع را نشان می‌دهد.



معمولا وقتی یک صفحه ASP.NET را طراحی می‌کنید، اصولا از شیء Response استفاده نمی‌کنید؛ بلکه کنترل‌های وب را به کار می‌برید. این کنترل‌ها، از شیء Response برای تولید محتوای خود استفاده می‌کنند. یک مشکل آن است که اگر مانند مثال قبل، از شیء Response استفاده کنید، نمی‌توانید محتویات مورد نظر را در جای مناسب از صفحه قرار دهید. تنها راه حل این است که محتویات دینامیک را در یک سری کنترل قرار دهید. به این ترتیب، کنترل می‌تواند هنگام تولید کردن خود، از ()Response.WriteSubstitution استفاده نماید.

اگر قصد ندارید از کنترل‌های شخصی استفاده کنید، برای به کار بردن جانشینی Post-Cache، یک راه میانبر دارید: یک کنترل عمومی Substitution که از این تکنیک برای دینامیک کردن محتویات خود استفاده می‌کند. باید کنترل Substitution را به متد استاتیکی متصل نمایید که محتویات دینامیک شما را برمی‌گرداند (مانند مثال قبل).

[1] Update