عنوان: **مدیریت درخواست‌های شرطی در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۲ ۰:۲۰
آدرس: www.dotnettips.info
گروه‌ها: MVC

فرض کنید کنید هدرهای کش کردن عناصر پویا و یا ثابت سایت را برای مدتی مشخص تنظیم کرده‌اید.
سؤال: مرورگر چه زمانی از کش محلی خودش استفاده خواهد کرد (بدون ارسال درخواستی به سرور) و چه زمانی مجددا از سرور درخواست دریافت مجدد این عنصر کش شده را می‌کند؟
برای پاسخ دادن به این سؤال نیاز است با مفهومی به نام Conditional Requests (درخواست‌های شرطی) آشنا شد که در ادامه به بررسی آن خواهیم پرداخت.

درخواست‌های شرطی

مرورگرهای وب دو نوع درخواست شرطی و غیر شرطی را توسط پروتکل HTTP و HTTPS ارسال می‌کنند. دراینجا، زمانی یک درخواست غیرشرطی ارسال می‌شود که نسخه‌ی ذخیره شده‌ی محلی منبع مورد نظر، مهیا نباشد. در این حالت، اگر منبع درخواستی در سرور موجود باشد، در پاسخ ارسالی خود وضعیت 200 یا HTTP/200 OK را باز می‌گرداند. اگر هدرهای دیگری نیز مانند کش کردن منبع در اینجا تنظیم شده باشند، مرورگر نتیجه‌ی دریافتی را برای استفاده‌ی بعدی ذخیره خواهد کرد.
در بار دومی که منبع مفروضی درخواست می‌گردد، مرورگر ابتدا به کش محلی خود نگاه خواهد کرد. همچنین در این حالت نیاز دارد که بداند این کش معتبر است یا خیر؟ برای بررسی این مورد ابتدا هدرهای ذخیره شده به همراه منبع، بررسی می‌شوند. پس از این بررسی اگر مرورگر به این نتیجه برسد که کش محلی معتبر است، دیگر درخواستی را به سرور ارسال نخواهد کرد.
اما در آینده اگر مدت زمان کش شدن تنظیم شده توسط هدرهای مرتبط، منقضی شده باشد (برای مثال با توجه به max-age هدر کش شدن منبع)، مرورگر هنوز هم درخواست کاملی را برای دریافت نسخه‌ی جدید منبع مورد نیاز، به سرور ارسال نمی‌کند. در اینجا ابتدا یک conditional request را به وب سرور ارسال می‌کند (یک درخواست شرطی). این درخواست شرطی تنها دارای هدرهای If-Modified-Since و یا If-None-Match است و هدف از آن سؤال پرسیدن از وب سرور است که آیا این منبع خاص، در سمت سرور اخیرا تغییر کرده‌است یا خیر؟ اگر پاسخ سرور خیر باشد، باز هم از همان کش محلی استفاده خواهد شد و مجددا درخواست کاملی برای دریافت نمونه‌ی جدیدتر منبع مورد نیاز، به سرور ارسال نمی‌گردد.
پاسخی که سرور جهت مشخص سازی عدم تغییر منابع خود ارسال می‌کند، با هدر HTTP/304 Not Modified مشخص می‌گردد (این پاسخ هیچ body خاصی نداشته و فقط یک سری هدر است). اما اگر منبع درخواستی اخیرا تغییر کرده باشد، پاسخ HTTP/200 OK را در هدر بازگشت داده شده، به مرورگر بازخواهد گرداند (یعنی محتوا را مجددا دریافت کن).

چه زمانی مرورگر درخواست‌های شرطی If-Modified-Since را به سرور ارسال می‌کند؟

اگر یکی از شرایط ذیل برقرار باشد، مرورگر حتی اگر تاریخ کش شدن منبع ویژه‌ای به 10 سال بعد تنظیم شده باشد، مجددا یک درخواست شرطی را برای بررسی اعتبار کش محلی خود به سرور ارسال می‌کند:
الف) کش شدن بر اساس هدر خاصی به نام vary صورت گرفته‌است (برای مثال بر اساس id یا نام یک فایل).
ب) اگر نحوه‌ی هدایت به صفحه‌ی جاری از طریق META REFRESH باشد.
ج) اگر از طریق کدهای جاوا اسکریپتی، دستور reload صفحه صادر شود.
د) اگر کاربر دکمه‌ی refresh را فشار دهد.
ه) اگر قسمتی از صفحه توسط پروتکل HTTP و قسمتی دیگر از آن توسط پروتکل HTTPS ارائه شود.
و ... اگر بر اساس هدر تاریخ مدت زمان کش شدن منبع، زمان منقضی شدن آن فرا رسیده باشد.

مدیریت درخواست‌های شرطی در ASP.NET MVC

تا اینجا به این نکته رسیدیم که قرار دادن ویژگی Output cache بر روی یک اکشن متد، الزاما به معنای کش شدن آن تا مدت زمان تعیین شده نخواهد بود و مرورگر ممکن است (در یکی از 6 حالت ذکر شده فوق) توسط ارسال هدر If-Modified-Since ، سعی در تعیین اعتبار کش محلی خود کند و اگر پاسخ 304 را از سرور دریافت نکند، حتما نسبت به دریافت مجدد و کامل آن منبع اقدام

خواهد کرد.

سؤال: چگونه می‌توان هدر If-Modified-Since را در ASP.NET MVC مدیریت کرد؟

پاسخ: اگر از فیلتر OutputCache استفاده می‌کنید، به صورت خودکار هدر Last-Modified را اضافه می‌کند؛ اما این مورد کافی نیست.

در ادامه یک کنترلر و اکشن متد GetImage آن‌را ملاحظه می‌کنید که تصویری را از مسیر app_data/images خوانده و بازگشت می‌دهد. همچنین این تصویر بازگشت داده شده را نیز با توجه به OutputCache آن به مدت یک ماه کش می‌کند.

```
using System.IO;
using System.Web.Mvc;

namespace MVC4Basic.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        const int AMonth = 30 * 86400;

        [OutputCache(Duration = AMonth, VaryByParam = "name")]
        public ActionResult GetImage(string name)
        {
            name = Path.GetFileName(name);
            var path = Server.MapPath(string.Format("~/app_data/images/{0}", name));
            var content = System.IO.File.ReadAllBytes(path);
            return File(content, "image/png", name);
        }
    }
}
```

با این View که تصویر خود را توسط اکشن متد GetImage تهیه می‌کند:

```
<img src="@Url.Action("GetImage","Home", new { name = "test.png"})"/>
```

در سطر اول متد GetImage، یک break point قرار دهید و سپس برنامه را توسط VS.NET اجرا کنید.

بار اول که صفحه‌ی اول برنامه درخواست می‌شود، یک چنین هدرهایی رد و بدل خواهند شد (توسط ابزارهای توکار مرورگر وب کروم تهیه شده‌است؛ همان دکمه‌ی F12 معروف):

```
Remote Address:127.0.0.1:5656
Request URL:http://localhost:10419/Home/GetImage?name=test.png
Request Method:GET
Status Code:200 OK

Response Headers
Cache-Control:public, max-age=2591916
Expires:Sat, 31 May 2014 12:45:55 GMT
Last-Modified:Thu, 01 May 2014 12:45:55 GMT
```

چون status code آن مساوی 200 است، بنابراین دریافت کامل فایل صورت خواهد گرفت. فیلتر OutputCache نیز مواردی مانند Cache-Control، Expires و Last-Modified را اضافه کرده‌است.

در همین حال اگر صفحه را ریفرش کنیم (فشردن دکمه‌ی F5)، اینبار هدرهای حاصل چنین شکلی را پیدا می‌کنند:

```
Remote Address:127.0.0.1:5656
Request URL:http://localhost:10419/Home/GetImage?name=test.png
Request Method:GET
Status Code:304 Not Modified

Request Headers
If-Modified-Since:Thu, 01 May 2014 12:45:55 GMT
```

در اینجا چون یکی از حالات صدور درخواست‌های شرطی (ریفرش صفحه) رخداده است، هدر If-Modified-Since نیز در درخواست حضور دارد. پاسخ آن از طرف وب سرور (و نه برنامه؛ چون اصلا متد کش شده‌ی GetImage دیگر اجرا نخواهد شد و به break point داخل آن نخواهیم رسید)، 304 یا تغییر نکرده‌است. بنابراین مرورگر مجددا درخواست دریافت کامل فایل را نخواهد داد.

در ادامه بجای اینکه صفحه را ریفرش کنیم، یکبار دیگر در نوار آدرس آن، دکمه‌ی Enter را فشار خواهیم داد تا آدرس موجود در آن (ریشه سایت) مجددا در حالت معمولی دریافت شود.

```
Remote Address:127.0.0.1:5656
Request URL:http://localhost:10419/Home/GetImage?name=test.png
Request Method:GET
Status Code:200 OK (from cache)
```

همانطور که ملاحظه می‌کنید اینبار پاسخ نمایش داده شده 200 است اما در ادامه‌ی آن ذکر شده‌است from cache. یعنی درخواستی را به سرور برای دریافت فایل ارسال نکرده است. عدم رسیدن به break point داخل متد GetImage نیز مؤید آن است.

مشکل! مرورگر را ببندید، تا کار دیباگ برنامه خاتمه یابد. مجددا برنامه را اجرا کنید. مشاهده خواهید کرد که ... اجرای برنامه در Break point قرار گرفته در سطر اول متد GetImage متوقف می‌شود. چرا؟! مگر قرار نبود تا یک ماه دیگر کش شود؟! هدر رد و بدل شده نیز Status Code:200 OK کامل است (که سبب دریافت کامل فایل می‌شود).

```
Remote Address:127.0.0.1:5656
Request URL:http://localhost:10419/Home/GetImage?name=test.png
Request Method:GET
Status Code:200 OK

Request Headers
If-Modified-Since:Thu, 01 May 2014 12:45:55 GMT
```

راه حل: هدر If-Modified-Since را باید برای اولین بار فراخوانی اکشن متدی که حاصل آن نیاز است کش شود، خودمان و به صورت دستی مدیریت کنیم (فیلتر OutputCache این‌کار را انجام نمی‌دهد). به نحو ذیل:

```
using System;
using System.IO;
using System.Net;
using System.Web.Mvc;

namespace MVC4Basic.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        const int AMonth = 30 * 86400;

        [OutputCache(Duration = AMonth, VaryByParam = "name")]
        public ActionResult GetImage(string name)
        {
            name = Path.GetFileName(name);
            var path = Server.MapPath(string.Format("~/app_data/images/{0}", name));

            var lastWriteTime = System.IO.File.GetLastWriteTime(path);
            this.Response.Cache.SetLastModified(lastWriteTime.ToUniversalTime());

            var header = this.Request.Headers["If-Modified-Since"];
            if (!string.IsNullOrWhiteSpace(header))
            {
                DateTime isModifiedSince;
                if (DateTime.TryParse(header, out isModifiedSince) && isModifiedSince > lastWriteTime)
                {
                    return new HttpStatusCodeResult(HttpStatusCode.NotModified);
                }
            }
```

```
            var content = System.IO.File.ReadAllBytes(path);
            return File(content, "image/png", name);
        }
    }
}
```

در این حالت اگر مرورگر هدر If-Modified-Since را ارسال کرد، یعنی آدرس درخواستی هم اکنون در کش آن موجود است؛ فقط نیاز دارد تا شما پاسخ دهید که آیا آخرین تاریخ تغییر فایل درخواستی، از زمان آخرین درخواست صورت گرفته از سایت شما، تغییری کرده‌است یا خیر؟ اگر خیر، فقط کافی است 304 یا HttpStatusCode.NotModified را بازگشت دهید (بدون نیاز به بازگشت اصل فایل).

برای امتحان آن همانطور که عنوان شد فقط کافی است یکبار مرورگر خود را کاملا بسته و مجددا برنامه را اجرا کنید.

```
Remote Address:127.0.0.1:5656
Request URL:http://localhost:10419/Home/GetImage?name=test.png
Request Method:GET
Status Code:304 Not Modified

Request Headers
If-Modified-Since:Thu, 01 May 2014 13:43:32 GMT
```

موارد کاربرد

اکثر فید خوان‌های معروف نیز ابتدا هدر If-Modified-Since را ارسال می‌کنند و سپس (اگر چیزی تغییر کرده بود) محتوای فید شما را دریافت خواهند کرد. بنابراین برای کاهش بار برنامه و هچنین کاهش میزان انتقال دیتای سایت، مدیریت آن در حین ارائه محتوای پویای فیدها نیز بهتر است صورت گیرد. همچنین هر جایی که قرار است فایلی به صورت پویا به کاربران ارائه شود؛ مانند مثال فوق.

تبدیل این کدها به روش سازگار با ASP.NET MVC

ما در اینجا رسیدیم به یک سری کد تکراری if و else که باید در هر اکشن متدی که OutputCache دارد، تکرار شود. روش AOP وار آن در ASP.NET MVC، تبدیل این کدها به یک فیلتر با قابلیت استفاده‌ی مجدد است:

```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = false)]
    public sealed class SetIfModifiedSinceAttribute : ActionFilterAttribute
    {
        public string Parameter { set; get; }
        public string BasePath { set; get; }

        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            var response = filterContext.RequestContext.HttpContext.Response;
            var request = filterContext.RequestContext.HttpContext.Request;

            var path = getPath(filterContext);
            if (string.IsNullOrWhiteSpace(path))
            {
                response.StatusCode = (int)HttpStatusCode.NotFound;
                filterContext.Result = new EmptyResult();
                return;
            }

            var lastWriteTime = File.GetLastWriteTime(path);
            response.Cache.SetLastModified(lastWriteTime.ToUniversalTime());

            var header = request.Headers["If-Modified-Since"];
            if (string.IsNullOrWhiteSpace(header)) return;
            DateTime isModifiedSince;
            if (DateTime.TryParse(header, out isModifiedSince) && isModifiedSince > lastWriteTime)
            {
                response.StatusCode = (int)HttpStatusCode.NotModified;
                response.SuppressContent = true;
                filterContext.Result = new EmptyResult();
            }
        }

        string getPath(ActionExecutingContext filterContext)
```

```
        {
            if (!filterContext.ActionParameters.ContainsKey(Parameter)) return string.Empty;
            var name = filterContext.ActionParameters[Parameter] as string;
            if (string.IsNullOrWhiteSpace(name)) return string.Empty;

            var path = Path.GetFileName(name);
            path = filterContext.HttpContext.Server.MapPath(string.Format("{0}/{1}", BasePath, path));
            return !File.Exists(path) ? string.Empty : path;
        }
    }
```

در اینجا توسط filterContext، می‌توان به مقادیر پارامترهای ارسالی به یک اکشن متد، توسط filterContext.ActionParameters دسترسی پیدا کرد. بر این اساس می‌توان مقدار پارامتر نام فایل درخواستی را یافت. سپس مسیر کامل آن‌را بازگشت داد. اگر فایل موجود باشد، هدر If-Modified-Since درخواست، استخراج می‌شود. اگر این هدر تنظیم شده باشد، آنگاه بررسی خواهد شد که تاریخ تغییر فایل درخواستی جدیدتر است یا قدیمی‌تر از آخرین بار مرور سایت توسط مرورگر.

و برای استفاده از آن خواهیم داشت:

```
[SetIfModifiedSince(Parameter = "name", BasePath = "~/app_data/images/")]
        [OutputCache(Duration = AMonth, VaryByParam = "name")]
        public ActionResult GetImage(string name)
        {
            name = Path.GetFileName(name);
            var path = Server.MapPath(string.Format("~/app_data/images/{0}", name));
            var content = System.IO.File.ReadAllBytes(path);
            return File(content, "image/png", name);
        }
```

البته بدیهی است اگر منطق ارسال 304 بر اساس تاریخ تغییر فایل باشد، روش فوق جواب خواهد داد. برای مثال اگر این منطق بر اساس تاریخ ثبت شده در دیتابیس است، قسمت محاسبه‌ی lastWriteTime را باید مطابق روش مطلوب خود تغییر دهید.

خلاصه‌ی بحث

چون فیلتر OutputCache در ASP.NET MVC، هدر If-Modified-Since را پردازش نمی‌کند (از این جهت که پردازش آن برای نمونه در مثال فوق وابسته به منطق خاصی است و عمومی نیست)، اگر با هر بار گشودن سایت خود مشاهده کردید، تصاویر پویایی که قرار بوده یک ماه کش شوند، دوباره از سرور درخواست می‌شوند (البته به ازای هرباری که مرورگر از نو اجرا می‌شود و نه در دفعات بعدی که صفحات سایت با همان وهله‌ی ابتدایی مرور خواهند شد)، نیاز است خودتان دسترسی کار پردازش هدر If-Modified-Since را انجام داده و سپس status code 304 را در صورت نیاز، ارسال کنید.

و در حالت عمومی، طراحی سیستم caching محتوای پویای شما بدون پردازش هدر If-Modified-Since ناقص است (تفاوتی نمی‌کند که از کدام فناوری سمت سرور استفاده می‌کنید).

برای مطالعه بیشتر

Understanding Conditional Requests and Refresh

Use If-Modified-Since header in ASP.NET

Make your browser cache the output of an HttpHandler

Your images from a database 304

Conditional GET

Website Performance with ASP.NET - Part4 - Use Cache Headers

ASP.NET MVC 304 Not Modified Filter for Syndication Content