فشرده سازی صفحات وب در ASP.NET ۲,۰

در این مقاله فرض بر آن است که خواننده، آشنایی در حد متوسط با ASP.NET و برخی مفاهیم مقدماتی آن دارد.
کدهای این مقاله با زبان VB.NET ارائه شده اند.

در دنیای امروز، انتقال سریع اطلاعات یکی از ارکان اصلی توسعه ی رو به جلوی فناوری محسوب می شود. وب نیز از این قائده مستثنی نیست و تلاش در جهت دریافت پاسخ توسط کلاینت در حداقل زمان ممکن، مخصوصا در زمان استفاده از اینترنت ایرانی(!) کارایی قابل توجهی در افزایش کیفیت ارائه ی خدمات خواهد داشت.

همان طور که می دانید، در زمان ارسال درخواست از سمت کلاینت به سرور، تعدادی اطلاعات که به عنوان Headerشناخته شده هستند نیز همراه با درخواست به سرور ارسال می شوند.
برای آگاهی از این Header ها می توانید خاصیت Trace دایرکتیو Page صفحه را به مقدار True تنظیم کنید. در این حالت، اطلاعاتی در مورد پردازش جاری در انتهای صفحه ی وب شما نمایش داده خواهد شد. اگر به قسمت Headers Collection دقت کنید، در جلوی عبارت"Accept-Encoding" ، انواع فشرده سازی ساپورت شده توسط مرورگر شما نمایش داده خواهد شد.
معمولا دو مقدار "gzip" و "deflate" در این قسمت نمایش داده می شوند و این مشخص می کند که مرورگر شما کدام یک از انواع فشرده سازی را ساپورت می کند .از این اطلاعات می توان برای تشخیص نوع فشرده سازی مناسب در سمت سرور برای درخواستی خاص بهره جست.

تکمیلی:
فشرده سازی داده ها، یکی از امکاناتی است که در نسخه ی ۱,۱ پروتوکول Http قرار داده شده و هر مرورگری که از این پروتوکول استفاده می کند (که تمامی مرورگرها این کار را می کنند)، می تواند از این امکان نیز استفاده کند.
در IE این قابلیت به طور پیش فرض فعال است. منویTools ، گزینه یInternet Options ، سر برگ Advanced، قسمتSettings HTTP ۱,۱ ، گزینه ی Use HTTP ۱,۱

یکی از امکانات جدیدی که در ASP.NET ۲,۰ گنجانده شده، امکان فشرده سازی داده ها است.
نیم اسپیس جدید System.IO.Compression شامل دو کلاس GzipStream و DeflateStream است.
این دو کلاس، دو نوع مختلف از عملیات فشرده سازی را ساپورت می کنند.
در حقیقت با استفاده از این دو کلاس، تعداد بایت های ارسالی به کلاینت را کاهش می دهیم.
این کاهش حجم، به معنای واقعی کلمه خواهد بود!

از آنجا که فشرده سازی باید بر روی تمامی درخواست ها انجام پذیرد، باید بتوان تمامی درخواست های ارسالی را هندل کرد.
استفاده از HttpHandler و HttpModule ها بهترین گزینه بدین منظور است.
HttpModuleها کنترل بیشتری بر روی درخواست ها خواهند داشت؛ پس بهترین کار، ایجاد یک HttpModuleاست که بر روی تمامی درخواست ها کنترل داشته باشد.

ابتدا یک کلاس به پروژه ی خود اضافه کنید. نام کلاس را "HttpCompression" می گذاریم.
این کلاس را در یک نیم اسپیس، مثلا با نام "Behrouz.Compression" قرار دهید.
نیم اسپیس هایWeb ، IOو Compression را به کلاس اضافه کنید.
از آنجا که این کلاس نقش یک Http Module را بازی می کند، باید اینترفیس "IHttpModule" را نیز بدین منظور به کار بگیریم:
کد توضیحات فوق:

```
Imports Microsoft.VisualBasic
Imports System.Web
Imports System.IO
Imports System.IO.Compression

Namespace Behrouz.Compression
Public Class HttpCompressionModule
Implements IHttpModule
End Class
End Namespace
```

اینترفیسIHttpModule ، دو متد دارد. متد Dispose و متدInit
متد Init مشخص می کند که HttpModule باید در چه رویدادی از رویدادهای پردازش فراخوانی شود.
برای اینکه این رویداد را به HttpModule معرفی کنیم، متدی تعریف می کنیم و سپس با استفاده از دستور AddHandler، این متد را به روال مربوطه نسبت می دهیم.

```
Public Sub Init(ByVal context As
System.Web.HttpApplication) Implements
System.Web.IHttpModule.Init
AddHandler context.BeginRequest, AddressOf Me.onZip
End Sub
```

آرگومان context که یکی از پارامترهای متد Init است، به برنامه ی جاری اشاره می کند.
روال BeginRequest بهترین مکان برای گرفتن درخواست هاست. به متد Init می گوییم که متدی با نام onZipتعریف خواهیم کرد که باید در روال BeginRequest فراخوانی شود.
قسمت اصلی کار در روال onZip خواهد بود.
در این روال، داده هایی را که باید به کلاینت ارسال شوند، قبل از ارسال، فشرده می کنیم.

```
    Public Sub onZip(ByVal sender As Object, ByVal e As
                                                EventArgs)
          Dim myApp As HttpApplication = CType(sender,
                                       HttpApplication)
                                Dim encodings As String =
          myApp.Request.Headers.Get("Accept-Encoding")
                    If encodings Is Nothing Then Return
        Dim myStream As Stream = myApp.Response.Filter
                         encodings = encodings.ToLower()
                  If (encodings.Contains("gzip")) Then
      myApp.Response.Filter = New GZipStream(myStream,
                                CompressionMode.Compress)
          myApp.Response.AddHeader("Content-Encoding",
                                                 "gzip")
             ElseIf encodings.Contains("deflate") Then
                               myApp.Response.Filter = New
    DeflateStream(myStream, CompressionMode.Compress)
          myApp.Response.AddHeader("Content-Encoding",
                                                 "deflate")
                                                    End If
                                                End Sub
```

sender، مرجعی است که قصد فشرده سازی را دارد و این مرجع، برنامه ی جاری است. به آن یک ارجا پیدا می کنیم.
مقدار هدر "Accept-Encoding" در متغیر encodings قرار می گیرد. در صورتی که این مقدار وجود نداشته باشد، مرورگر به هر دلیلی فشرده سازی داده ها را ساپورت نخواهد کرد. در این صورت، برنامه با دستور Returnبه کار خود پایان خواهد داد.
متد Filter کلاسResponse ، فیلتری را به داده هایی که قرار است به سمت کلاینت ارسال شوند اعمال می کند. این داده ها به شکل Stream خواهند بود.
در ابتدا باید نوع فشرده سازی ای که توسط مرورگر ساپورت می شود را تشخیص دهیم. این کار با متد Containsمتغیر encodings امکان پذیر است. در ابتدا وجود عبارت gzip بررسی می شود و در صورتی که این مقدار جز مقادیر هدر بود، نوع فشرده سازی Gzip به آن اعمال می شود و مقدار هدر-Content"

"Encodingبه "gzip" تنظیم می شود. این مقدار برای آگاهی از نوع فشرده سازی است که بر روی داده های دریافتی اعمال می شود.
در صورتی که فشرده سازی gzip توسط مرورگر ساپورت نشود، وجود مقدار "deflate" بررسی می شود و این نوع فشرده سازی به داده ها اعمال می شود.

توجه:
فشرده سازی Gzip و Deflate تفاوتی در میزان کاهش حجم داده ها ندارند .جزئیات فشرده سازی Gzip در RFC ۱۹۵۲و فشرده سازی Deflate در RFC ۱۹۵۱ شرح داده شده است. در این مراجع شرح داده شده است که این نوع فشرده سازی ها از ترکیبی از الگوریتم های "LZ۷۷" و "هافمن" استفاده می کنند و حداکثر میزان فشرده سازی آنها برای داده هایی با حداکثر حجم "چهار گیگابایت" است.
فرمت Gzip یک استاندارد فشرده سازی Open Source و جزء فرمت های متداول در سیستم عامل هایی همچون لینوکس هست.

خسته نباشید! تنها کار باقیمانده، تبدیل کلاس به فایل DLL است.
با استفاده از کامپایلر VB.NET فایل (vbc.exe و یا کامپایلر C#.NET فایل (csc.exe می توانید این کار را انجام دهید.
فایل های فوق در مسیر زیر وجود دارند:

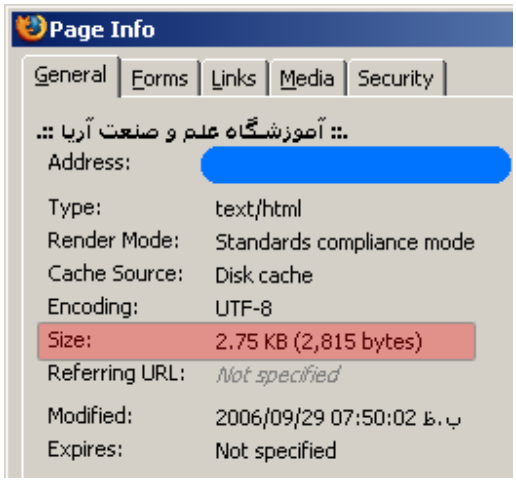
drive:\WINDOWS\Microsoft.NET\Framework\v۲,۰,۵۰۷۲۷
با استفاده از خط فرمان DOS فایل را کامپایل می کنیم.
از منوی Start ویندوز، گزینه ی Run را انتخاب کنید. عبارت "cmd" را وارد کنید.
با دستور cd به مسیر فوق بروید و سپس عبارت زیر را وارد کنید:

vbc /t:library /r:System.dll,System.Web.dll
D:\inetpub\wwwroot\myApp\App_Code\HttpCompression. vb
مسیر فایل HttpCompression.vb را بر مبنای سیستم خود تنظیم کنید.
پس از اجرای دستور فوق، فایل DLL حاصل در پوشه ی App_Code قرار خواهد گرفت.
حال پوشه ای با نام "bin" ایجاد کنید و DLL فوق را به آن انتقال دهید.
برای شناساندن این HttpModule به برنامه، باید آن را در فایل Web.Config تعریف کرد.
پس از عبارت <system.web> دستورات زیر را بنویسید:

```
<httpModules>
<add name="HttpCompressionModule"
type="Behrouz.Compression.HttpCompressionModule,
HttpCompressionModule"/>
```

```
</httpModules>
```

بهترین راه برای دیدن نتیجه ی کار، استفاده از مرورگر Firefox است!
پس از اجرای برنامه درFirefox ، از منوی Tools گزینه ی Page Info را انتخاب کنید.
به عنوان مثال، من صفحه ای رو در هر دو حالت نرمال و فشرده مقایسه کردم. نتیجه واقعا شگفت انگیز بود!
به دو عکس زیر دقت کنید:



همان طور که ملاحظه می کنید، حجم خروجی صفحه پس از فشرده سازی، تقریبا یک هشتم حجم صفحه در حالت نرمال است!

فشرده سازی را به عنوان یک اصل ثابت در تمامی برنامه های خود به کار ببرید..

فایل های ضمیمه

http://barnamenevis.org/forum/attachment.php?attachmentid=۵۵۰۵&d=۱۱۶۲۰۵۶۳۷۰

نویسنده : مهندس بهروز راد