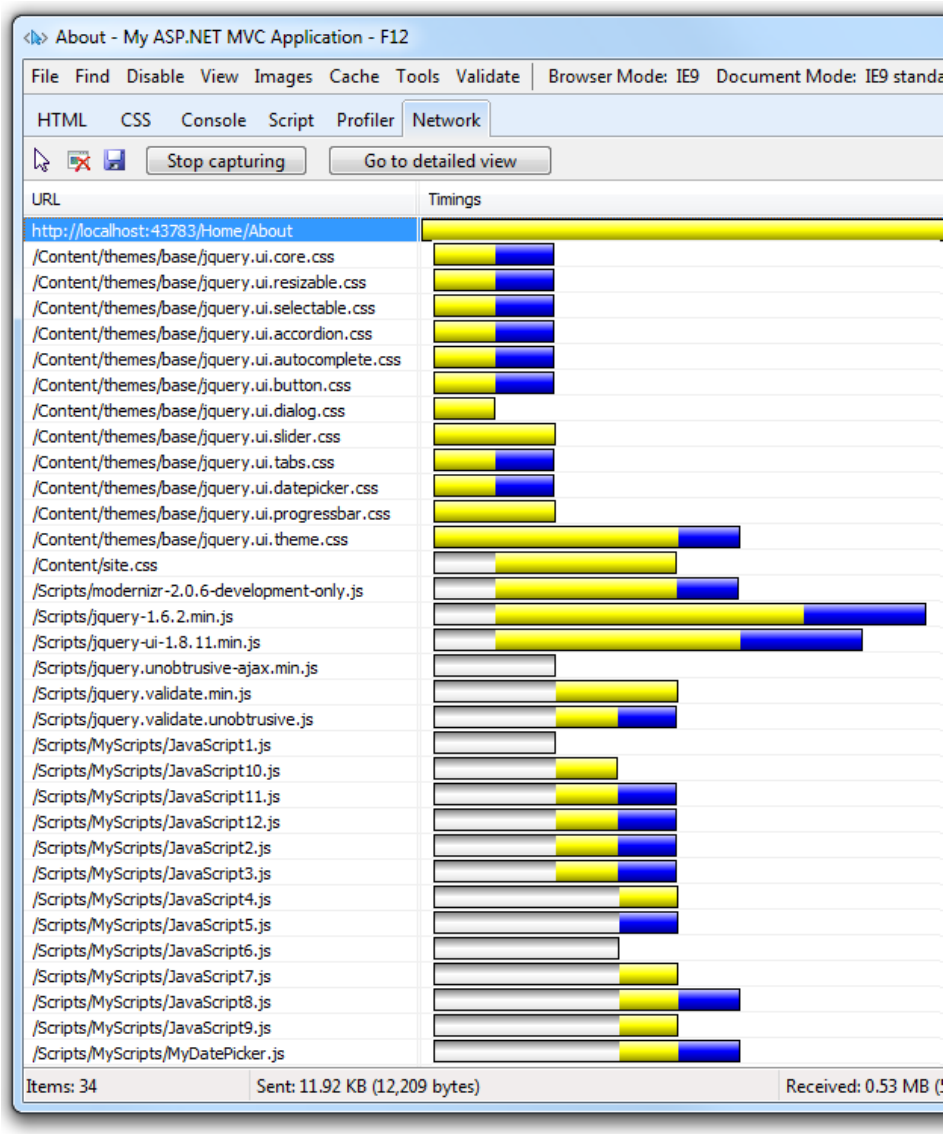
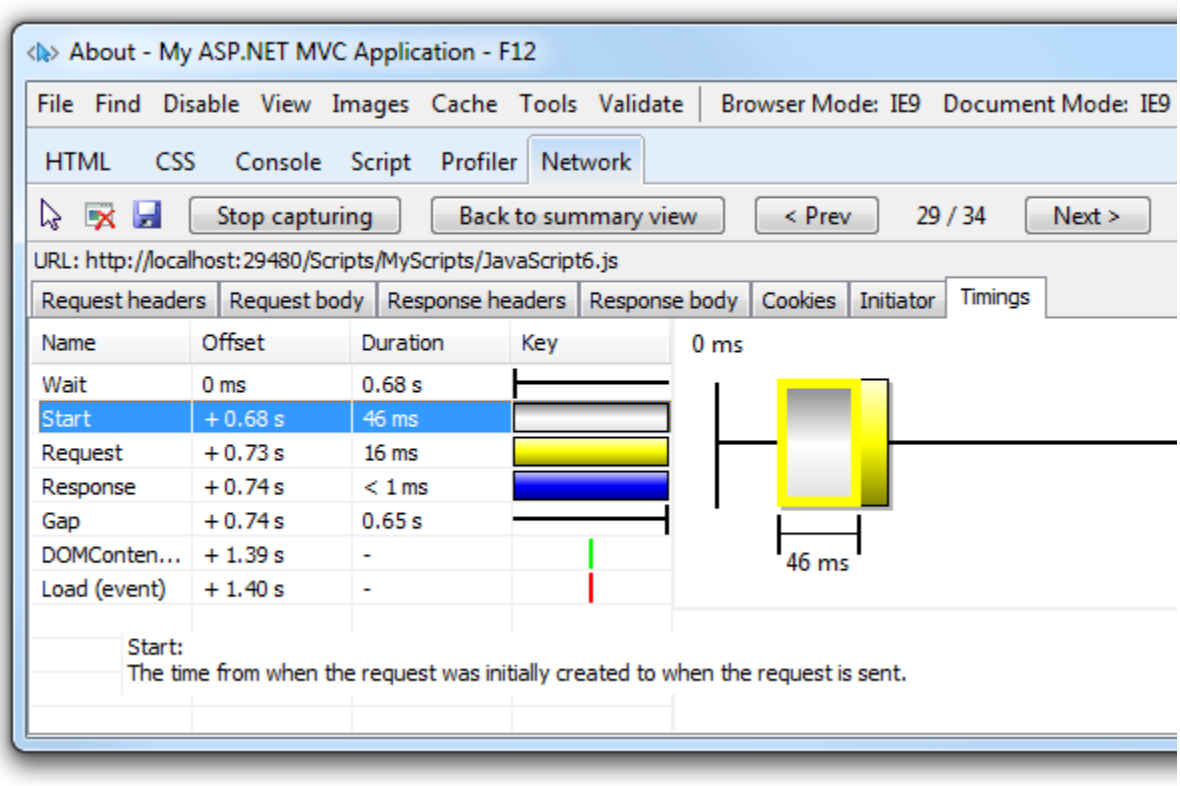
آموزش MVC – Bundling و Minification در MVC

Bundling (گنجاندن چند فایل در قالب یک فایل واحد) و minification (کاهش حجم فایل و حذف کردن space و کاراکترهای اضافی و کامنت ها از فایل های jQuery) دو تکنیک جهت بهینه سازی اپلیکیشن هستند که از ویرایش ASP.NET 4.5 جهت کاهش زمان لازم برای زمان بارگذاری request استفاده می شود. دو روش مزبور در واقع با کاهش تعداد درخواست هایی که به سرور فرستاده شده و نیز فشرده سازی منابع درخواستی از سرویس دهنده (asset هایی نظیر فایل های CSS و JavaScript) این کار را انجام می دهند.

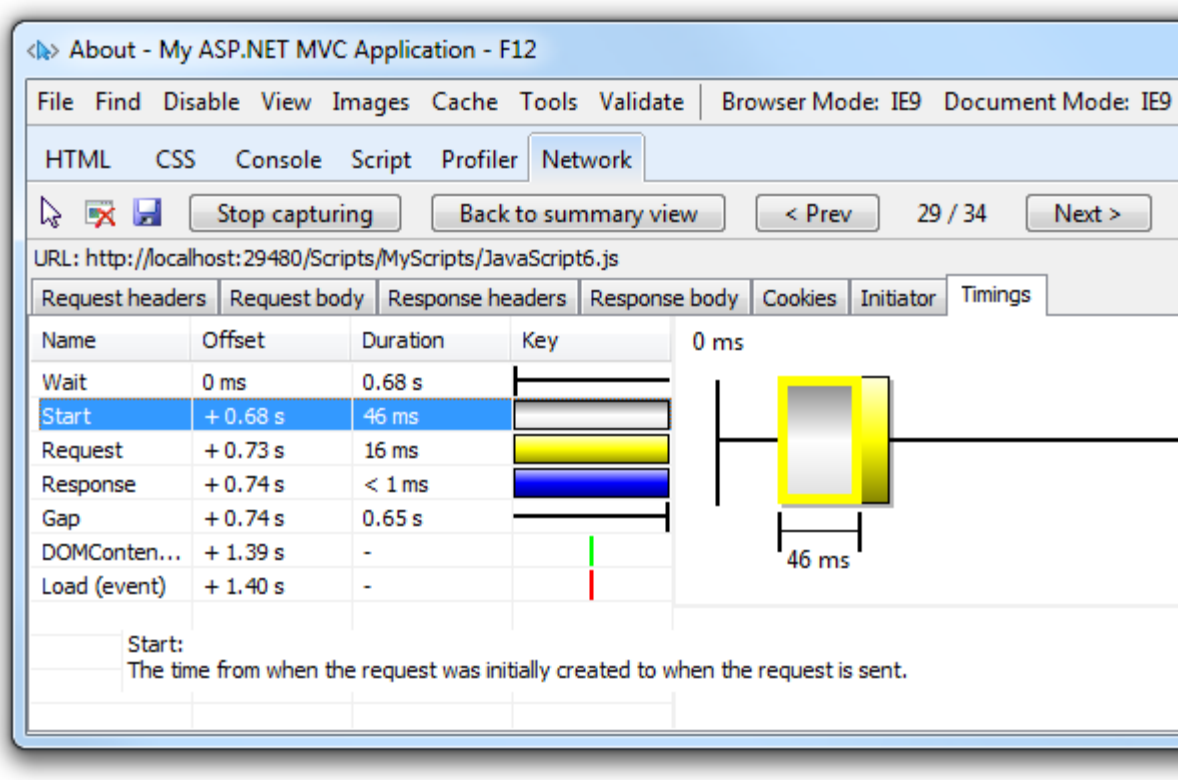
امروزه اکثر مرورگرهای مطرح نهایتا شش درخواست و connection موازی به هر سرور (hostname) ارسال می کنند. بنابراین زمانی که مرورگر شش request را به سرور جهت پردازش ارسال می کند، سایر request هایی که درخواست واکشی منابع (asset) خاصی از سرور را دارند، (توسط مرورگر) در یک صف قرار داده می شوند. در تصویر زیر، تب Network از ابزار توسعه دهنده ی IE (که با F12 باز می شود)، میزان زمانی که طول می کشد تا منابع (asset) مورد نیاز ویو (view) About از اپلیکیشن در سرور واکشی شود را نمایش می دهد.



نوار خاکستری رنگ میزان زمانی که request، در صف منتظر مانده تا سرور به آن پاسخ دهد را نمایش می دهد. به عبارت دیگر مقدار زمانی که از ساخت request تا ارسال آن به سرور به طول می انجامد را نشان می دهد. نوار زرد رنگ نشانگر زمانی است که طول می کشد تا request ارسالی از مرورگر اولین response را از سرور دریافت کند. نوارهای آبی رنگ زمانی که طول می کشد تا داده های درخواستی از سرور واکشی شوند، را نشان می دهد. جهت مشاهده ی اطلاعات دقیق زمان سنجی می توانید بر روی asset مورد نظر دابل کلیک کنید. برای مثال، تصویر زیر اطلاعات دقیق درباره ی میزان زمانی که طول می کشد تا فایل /Scripts/MyScripts/JavaScript6.js بارگذاری شود را نشان می دهد.



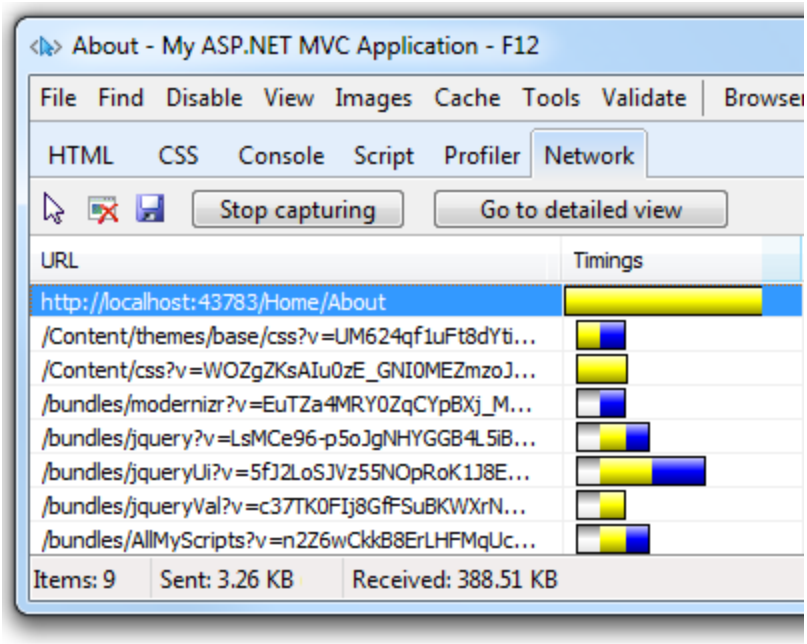
در تصویر بالا میزان زمانی که request در صف منتظر مانده تا request قبلی کامل شده و این request به سرور فرستاده شود را مشخص می کند. در نمونه ی حاضر request مورد نظر 46 ثانیه در صف منتظر مانده تا request قبلی پاسخ خود را از سرور دریافت کرده و کامل شود.



Bundling

Bundling یک امکان جدید در ASP.NET 4.5 است که چندین فایل را در قالب یک فایل واحد می گنجاند. با بهره گیری از این قابلیت توسعه دهنده می تواند چندین فایل css یا javascript را در یک فایل به صورت bundle جای داده و بدین وسیله تعداد request های ارسالی از مرورگر به سرور را کاهش داده و زمان بالا آمدن صفحه را پایین بیاورد. به عبارت دیگر، تعداد کمتر فایل برای بارگذاری معادل تعداد HTTP Request کمتر به سرور است که زمان بارگذاری اولین صفحه ی سایت را کاهش می دهد.

تصویر زیر همان تصویر قبلی از صفحه ی About است که این بار تاثیر استفاده از قابلیت های bundle و minify کردن در میزان زمان لازم برای بارگذاری صفحه ی سایت را نمایش می دهد.



Minification

minification فرایندی است که طی آن اسکریپت های js و کدهای css فشرده شده و حجم فایل کاهش می یابد. این کار به وسیله ی حذف فاصله ها و کامنت ها از کد و تغییر اسم متغیرها به یک کاراکتر صورت می پذیرد.

تابع JavaScript زیر را درنظر بگیرید:

```
AddAltToImg = function (imageTagAndImageID, imageContext) {
  ///<signature>
  ///<summary> Adds an alt tab to the image
  // </summary>
  //<param name="imgElement" type="String">The image selector.</param>
  //<param name="ContextForImage" type="String">The image context.</param>
  ///</signature>
  var imageElement = $(imageTagAndImageID, imageContext);
  imageElement.attr('alt', imageElement.attr('id').replace(/ID/, ''));
}
```

پس از minify کردن، کد بالا به صورت زیر درمی آید:

```
AddAltToImg = function (n, t) { var i = $(n, t); i.attr("alt", i.attr("id").replace(/ID/, "")) }
```

علاوه بر حذف کامنت ها و فاصله های غیر ضروری، نام متغیرها و پارامترها جهت کاهش بیشتر حجم فایل به صورت زیر کوتاه و بهینه سازی شده اند:

اسم متغیر پس از minification	اسم اصلی متغیر قبل از minification
n	imageTagAndImageID
t	imageContext
i	imageElement

تاثیر استفاده از قابلیت های bundling و minification در کارایی اپلیکیشن

در جدول زیر تفاوت زمانی که بارگذاری تمامی asset ها به صورت جداگانه انجام می شود با زمانی که از bundle/minify استفاده شده را مشاهده می کنید. برای مثال، در سطر file request، زمانی که از دو امکان نام برده استفاده شده، تعداد request های ارسال شده به سرور به 9 بار کاهش یافته است. همین مقدار در صورت عدم استفاده از این دو امکان معادل 34 می باشد.

	Using B/M	Without B/M	Change
File Requests	9	34	256%
KB Sent	3.26	11.92	266%
KB Received	388.51	530	36%
Load Time	510 MS	780 MS	53%

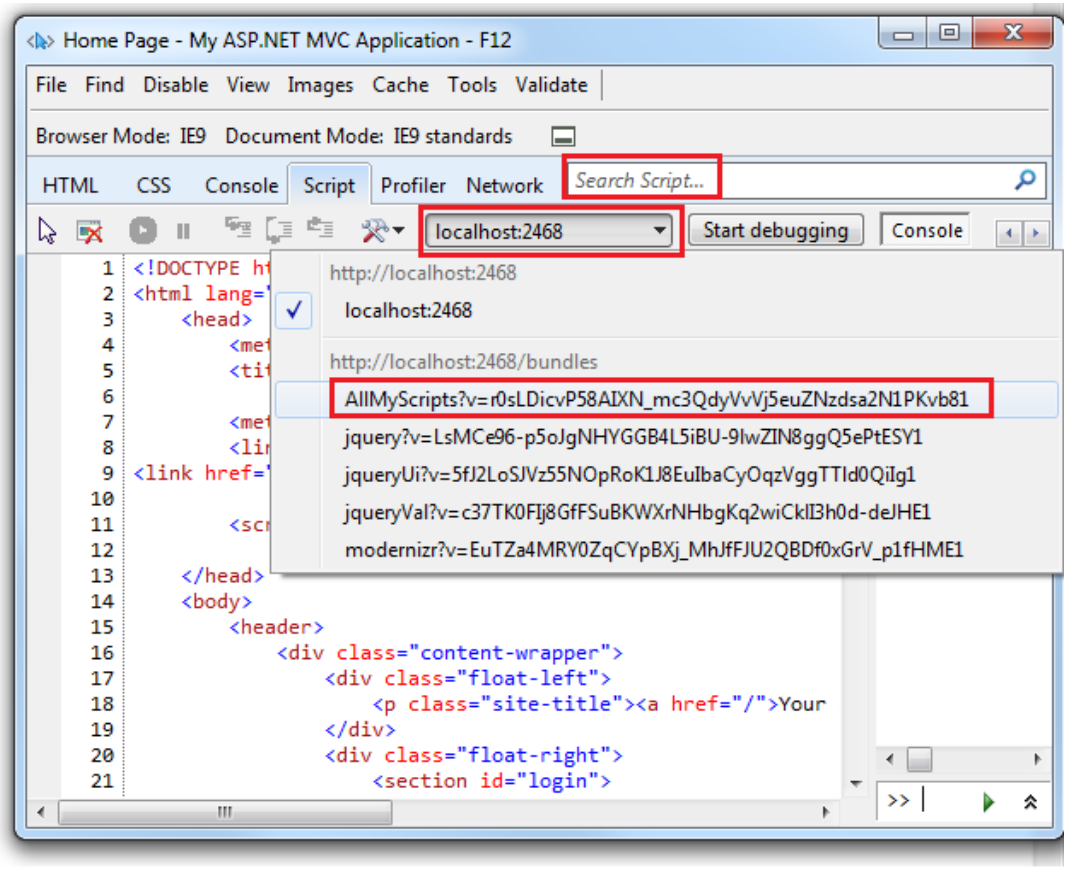
با توجه به اطلاعات جدول فوق، تعداد بایت های رد و بدل شده، پس از اعمال bundle/minify، کاهش یافته است (مرورگرها header های طولانی بر روی request ها اعمال می کنند و با کاهش تعداد درخواست از تعداد header هایی که مرورگر با request به سرور ارسال می نماید، کاسته می شود). همان طور که می بینید تعداد بایت های دریافتی پس از اعمال B/M چندان کاهش نیافته است چرا که فایل های حجیم (Scripts\jquery-ui-1.8.11.min.js و Scripts\jquery-1.7.1.min.js) قبلا minify شده اند.

توجه: مقادیر مربوط به زمان سنجی که در جدول بالا مشاهده می کنید، از ابزار Fiddler برای شبیه سازی یک شبکه ی کند بهره می گیرد. (از منوی Rules در ابزار Fiddler، ابتدا گزینه ی Performance و سپس Simulate Modem Speeds را انتخاب نمایید).

اشکال زدایی (debug) فایل های Bundle و minify شده ی JavaScript

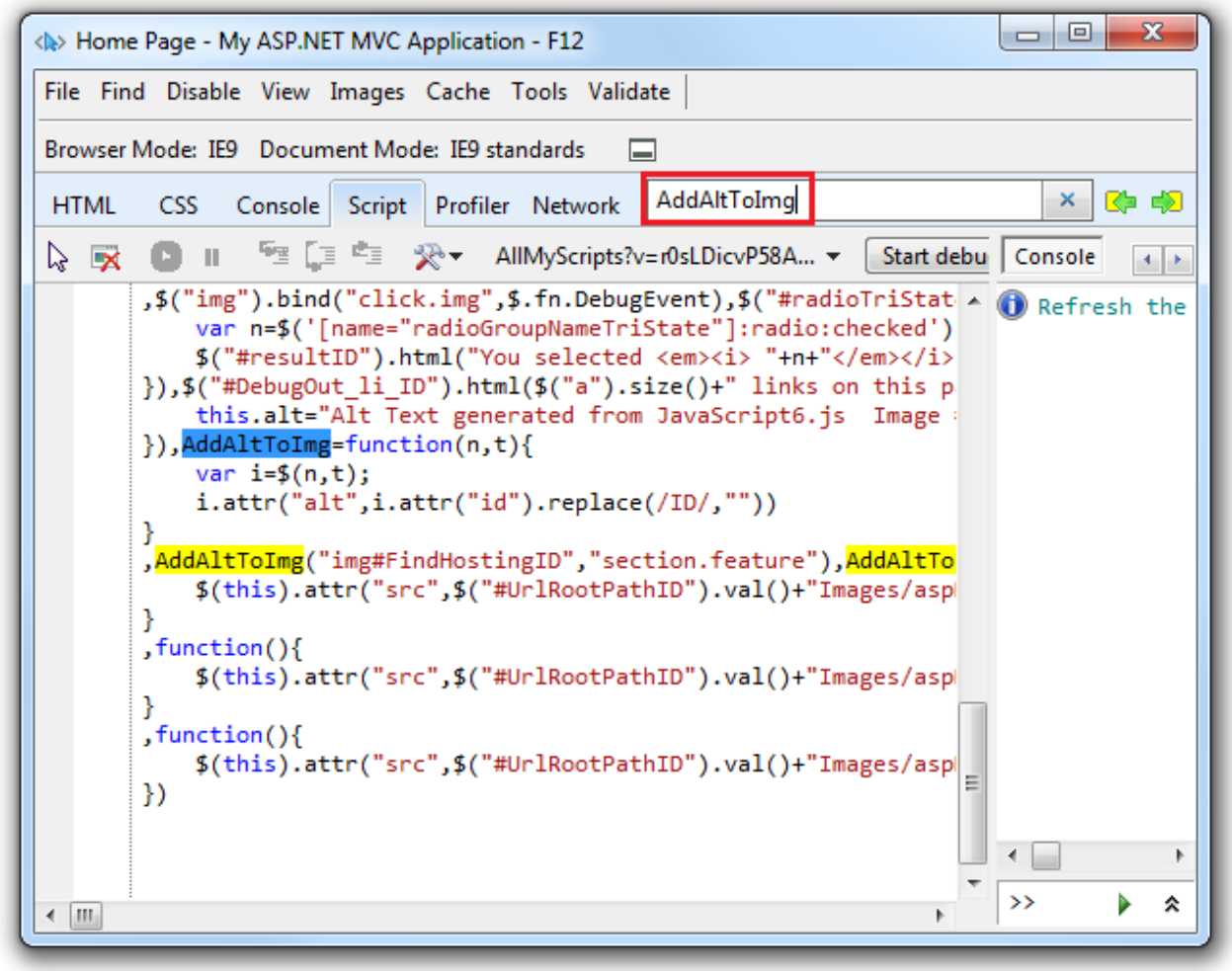
debug کردن کدهای JavaScript در محیط توسعه (محیطی که در آن المان مربوط به compilation داخل فایل web.config به صورت "debug="true تنظیم شده است) آسان است چرا که فایل های JavaScript هنوز bundle و minify نشده اند. با این حال می توان فایل های JavaScript را پس از minify و bundle کردن نیز debug کرد. به طور مثال، جهت debug یک تابع JavaScript (پس از minify و bundle شدن) با استفاده از ابزار مربوط به توسعه دهنده مرورگر IE که با دکمه ی F12 در در اختیار شما قرار می گیرد، می توانید به صورت زیر اقدام کنید:

1. تب Script را انتخاب نمایید و سپس دکمه ی Start debugging را فشار دهید.
2. حال bundle حاوی کد JavaScript که می خواهید با استفاده از دکمه ی assets اشکال زدایی نمایید را انتخاب کنید.



3. جهت تنظیم کد JavaScript، پس از کلیک بر روی دکمه ی  ، گزینه ی Format JavaScript را انتخاب نمایید.

4. در کادر Search Script، اسم تابعی را که می خواهید debug نمایید، وارد کنید. همان طور که در تصویر زیر مشاهده می کنید، اسم تابع AddAltToImg جهت debug در کادر Search Script درج شده است.



تنظیم و مدیریت bundling و minification

جهت فعال/غیرفعال کردن قابلیت های Bundling و minification، کافی است به المان compilation در فایل Web.config مراجعه کرده و خاصیت (attribute) debug را مقدار دهی نمایید. در فایل XML زیر همان طور که مشاهده می کنید، مقدار debug برابر true قرار داده شده است، بنابراین minification و bundling غیرفعال هستند.

```
<system.web>
  <compilation debug="true" />
```

```
  <!-- Lines removed for clarity. -->
</system.web>
```

جهت فعال سازی bundling و minification، مقدار خاصیت debug (attribute) را برابر "false" قرار دهید. می توانید تنظیمات فایل Web.config را با تنظیم ویژگی EnableOptimizations (property) در کلاس BundleTable، مطابق نیاز بازنویسی (override) نمایید. کد زیر دو قابلیت minification و bundling را فعال نموده و تمامی تنظیمات داخل فایل Web.config را بازنویسی می کند.

```
public static void RegisterBundles(BundleCollection bundles)
{
  bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
        "~/Scripts/jquery-{version}.js"));
  // Code removed for clarity.
  BundleTable.EnableOptimizations = true;
}
```

نکته: برای اینکه فایل ها bundle و minify شوند، یا EnableOptimization بایستی بر روی true تنظیم شده باشد یا مقدار خاصیت debug (attribute) در المان compilation، داخل فایل Web.config برابر false قرار داده شود. همچنین در صورتی که یکی از دو تنظیم فوق انجام نشده باشد، ورژن min. فایل های مربوطه انتخاب نمی شود (ورژن full debug فایل ها به مرورگر ارسال می گردد). لازم به ذکر است که ویژگی EnableOptimizations در کلاس BundleTable، بر مقدار خاصیت debug تگ compilation داخل فایل Web.config اولویت داشته و می تواند آن را بازنویسی کند.

استفاده از Bundling و Minification در اپلیکیشن های ASP.NET MVC

در این بخش، یک پروژه ی ASP.NET MVC ایجاد کرده و از قابلیت های bundling و minification جهت افزایش کارایی اپلیکیشن و بهبود زمان بارگذاری سایت استفاده کرده و تاثیر آن را مشاهده می کنیم. ابتدا، یک پروژه ی اینترنتی ASP.NET MVC به نام MvcBM بدون دستکاری تنظیمات پیش فرض، ایجاد نمایید.

فایل App_Start\BundleConfig.cs را باز کرده و متد RegisterBundles که وظیفه ی آن ایجاد، معرفی (register) و تنظیم Bundle ها می باشد را چک نمایید. کد زیر بخشی از متد RegisterBundles را نشان می دهد.

```
public static void RegisterBundles(BundleCollection bundles)
```

```
{
   bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
         "~/Scripts/jquery-{version}.js"));
     // Code removed for clarity.
}
```

کد قبلی یک فایل bundle شده ی جدید JavaScript به نام ~/bundles/jquery ایجاد می کند که تمامی فایل های لازم (که debug و minify شده ولی فایل های با پسوند vsdoc. را شامل نمی شود) در پوشه ی Scripts که با رشته ی "~/Scripts/jquery-{version}.js" مطابقت دارند را داخل خود کپسوله می نماید. در APS.NET MVC 4، برای debug configuration یا نسخه ی غیرنهایی، فایل jquery-1.7.1.js به bundle اضافه می شود. در release configuration یا نسخه ی آماده برای نصب، فایل jquery-1.7.1.min.js به bundle اضافه خواهد شد. فریم ورک bundling، از قواعد قراردادی مشترک زیر پیروی می نماید:

- زمانی که دو فایل "FileX.min.js" و "FileX.js" در پوشه ی Scripts موجود باشد، در آن صورت برای نسخه ی آماده برای نصب (release) فایل ".min" را انتخاب می کند.
- برای نسخه ی غیرنهایی اپلیکیشن یا debug، فایل ".min" را انتخاب می کند.
- فایل های "-vsdoc" (نظیر jquery-1.7.1-vsdoc.js) که تنها برای استفاده ی ابزار IntelliSense تولید می شوند را کاملا نادیده می گیرد.

استفاده از مکان نگهدار {version} در رشته ورودی تابع include در مثال بالا جهت جستجو و تطبیق با فایل های موجود در Scripts، سبب می شود که یک bundle از فایل های jQuery با ورژن متناسب jQuery در پوشه ی Scripts پروژه ی شما، به صورت خودکار تولید شود. در مثال جاری استفاده از wild card (منظور همان مکان نگهدار {version} در رشته ی ورودی تابع include می باشد) مزیت های زیر را دارد:

- به شما این امکان را می دهد تا بدون نیاز به اعمال تغییرات در کد bundling قبلی یا reference های کتابخانه ی jQuery در صفحات view خود، ورژن جدید کتابخانه ی jQuery را دانلود و نصب نمایید.
- به صورت خودکار برای نسخه ی غیرنهایی اپلیکیشن/debug config ورژن کامل فایل را انتخاب می کند و ویژه ی نسخه ی نهایی و آماده ی برای نصب (release build) اپلیکیشن از ورژن ".min" استفاده می نماید.

استفاده از CDN (شبکه ی تحویل محتوا)

کد زیر باندل jQuery مستقر در سیستم (local jQuery bundle) را با باندل jQuery مستقر در CDN جایگزین می کند.

```
public static void RegisterBundles(BundleCollection bundles)
{
  //bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
  //        "~/Scripts/jquery-{version}.js"));
  bundles.UseCdn = true;  //enable CDN support
  //add link to jquery on the CDN
  var jqueryCdnPath = "http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.1.min.js";
  bundles.Add(new ScriptBundle("~/bundles/jquery",
        jqueryCdnPath).Include(
        "~/Scripts/jquery-{version}.js"));
  // Code removed for clarity.
}
```

در کد بالا، زمانی که اپلیکیشن برای نصب نهایی آماده بود و در حالت release قرار دارد، کتابخانه ی jQuery از سرور CDN فراخوانی می شود (Request) و هنگامی که اپلیکیشن در حالت debug و تست تنظیم شده، این کتابخانه به صورت محلی و از سرور میزبان واکشی می شود. لازم به ذکر است که اگر jQuery را از شبکه ی تحویل محتوا (CDN) فراخوانی می کنید، باید یک جایگزین یا مکانیزم پشتیبان (fallback) نیز داشته باشید که چنانچه به هر دلیلی Request موفق نبود، اپلیکیشن با مشکل مواجه نشود. تکه کد (markup) مشخص شده در انتهای فایل layout، یک اسکریپت (کد JavaScript) را نمایش می دهد که اگر درخواست فراخوانی jQuery از CDN با شکست مواجه شد، آن را از جای دیگری واکشی کند.

```
</footer>
    @Scripts.Render("~/bundles/jquery")
  <script type="text/javascript">
    if (typeof jQuery == 'undefined') {
      var e = document.createElement('script');
      e.src = '@Url.Content("~/Scripts/jquery-1.7.1.js")';
      e.type = 'text/javascript';
      document.getElementsByTagName("head")[0].appendChild(e);
    }
    </script>
    @RenderSection("scripts", required: false)
  </body>
</html>
```

ایجاد یک Bundle

متد Include از کلاس Bundle آرایه ای از مقادیر رشته ای را به عنوان پارامتر ورودی دریافت می کند که هر یک از این رشته ها، بیانگر آدرس مجازی (virtual path) منبع و فایل مورد نیاز می باشد. کد زیر از بدنه ی متد RegisterBundles مقیم در پوشه ی APP-Start\BundleConfig.cs ، نحوه ی افزودن چندین فایل به یک bundle را نمایش می دهد:

```
bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
            "~/Content/themes/base/jquery.ui.core.css",
            "~/Content/themes/base/jquery.ui.resizable.css",
            "~/Content/themes/base/jquery.ui.selectable.css",
            "~/Content/themes/base/jquery.ui.accordion.css",
            "~/Content/themes/base/jquery.ui.autocomplete.css",
            "~/Content/themes/base/jquery.ui.button.css",
            "~/Content/themes/base/jquery.ui.dialog.css",
            "~/Content/themes/base/jquery.ui.slider.css",
            "~/Content/themes/base/jquery.ui.tabs.css",
            "~/Content/themes/base/jquery.ui.datepicker.css",
            "~/Content/themes/base/jquery.ui.progressbar.css",
            "~/Content/themes/base/jquery.ui.theme.css"));
```

متد IncludeDirectory از کلاس Bundle تمامی فایل های موجود در یک پوشه (و گاهی حتی زیرپوشه ها) که با پارامتر ورودی دوم (searchPattern که نشانگر الگویی است که جستجو بر اساس آن صورت می گیرد) منطبق می باشند را به bundle اضافه می کند. توابع (API) IncludeDirectory از کلاس Bunlde در زیر نمایش داده شده اند:

```
public Bundle IncludeDirectory(
            string directoryVirtualPath,  // The Virtual Path for the directory.
            string searchPattern)       // The search pattern.
public Bundle IncludeDirectory(
            string directoryVirtualPath,  // The Virtual Path for the directory.
            string searchPattern,       // The search pattern.
            bool searchSubdirectories)   // true to search subdirectories.
```

می توان در فایل های view با استفاده از متد Render، به bundle ها دسترسی داشته و آن ها را فراخوانی کرد. برای فراخوانی فایل های CSS از متد Styles.Render و برای فراخوانی فایل های JavaScript از متد Scripts.Render استفاده می شود. کد (markup) زیر از فایل Views\Shared_Layout.cshtml، فایل view های یک پروژه ی MVC که باندل های CSS و JavaScript را فراخوانی می کنند را نمایش می دهد.

```
<!DOCTYPE html>
<html lang="en">
<head>
    @* Markup removed for clarity.*@
 @Styles.Render("~/Content/themes/base/css", "~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    @* Markup removed for clarity.*@
   @Scripts.Render("~/bundles/jquery")
   @RenderSection("scripts", required: false)
</body>
</html>
```

متدهای Render آرایه ای از رشته ها را به عنوان ورودی می گیرند بنابراین شما می توانید همزمان چندین bundle را در تنها یک خط به view اضافه نمایید. بهتر است برای فراخوانی bundle ها در view، از متدهای Render ای که خود کد HTML لازم برای دسترسی (reference) به asset یا منبع مورد نیاز را ایجاد می نماید، استفاده کنید. شما می توانید با استفاده از متد Url آدرس URL منبع مورد نظر (asset) را بدون استفاده از کد HTML تولید کرده و از این طریق به آن منبع دسترسی داشته باشید.

در زیر با استفاده از خاصیت async و متد Url، فایل modernizr را فراخوانی می کنیم.

```
<head>
    @*Markup removed for clarity*@
    <meta charset="utf-8" />
    <title>@ViewBag.Title - MVC 4 B/M</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width" />
    @Styles.Render("~/Content/css")
   @* @Scripts.Render("~/bundles/modernizr")*@
    <script src='@Scripts.Url("~/bundles/modernizr")' async></script>
</head>
```

استفاده از کاراکتر "*" در انتخاب فایل ها

می توان در پارامتر متد Include و پارامتر SearchPattern متد IncludeDirectory از یک کاراکتر "*" به صورت پیشوند یا پسوند استفاده کرد. این امر به شما امکان می دهد که آخرین بخش از آدرس قرارگیری فایل (path segment) را بدون نوشتن اسم کامل آن فراخوانی کنید. رشته ای که جستجو فایل بر اساس آن صورت می گیرد (search string) نسبت به کوچک و بزرگی حروف حساس نیست. متد IncludeDirectory علاوه بر

جستجو پوشه ها، زیرپوشه های (subdirectory) منطبق با پارامتر SearchPattern را نیز تطبیق داده و در صورت پیدا کردن فایل، آن را فراخوانی و به bundle اضافه می کند.

پروژه ای با فایل های JavaScript زیر را در نظر بگیرید:

- Scripts\Common\AddAltToImg.js
- Scripts\Common\ToggleDiv.js
- Scripts\Common\ToggleImg.js
- Scripts\Common\Sub1\ToggleLinks.js

جدول زیر فایل هایی که به وسیله ی کاراکتر "*" به bundle اضافه شده اند را نمایش می دهد:

فراخوانی متد	فایلی که به bundle اضافه شده یا خطایی که در صورت عدم موفقیت رخ می دهد
Include("~/Scripts/Common/*.js")	*AddAltToImg.js, ToggleDiv.js, ToggleImg.js*
Include("~/Scripts/Common/T*.js")	Invalid pattern exception. The wildcard character is only allowed on the prefix or suffix.
Include("~/Scripts/Common/*og.*")	Invalid pattern exception. Only one wildcard character is allowed.
"Include("~/Scripts/Common/T*")	*ToggleDiv.js, ToggleImg.js*
"Include("~/Scripts/Common/*")	Invalid pattern exception. A pure wildcard segment is not valid.
IncludeDirectory("~/Scripts/Common", "T*")	*ToggleDiv.js, ToggleImg.js*
IncludeDirectory("~/Scripts/Common", "T*",true)	*ToggleDiv.js, ToggleImg.js, ToggleLinks.js*

توسعه دهندگان اغلب فایل ها را به صورت صریح و با ذکر نام کامل آن ها به bundle ا ضافه کرده و آن را به بارگذاری با استفاده از کاراکتر "*" ترجیح می دهند. دلایل انجام این کار به شرح زیر می باشند:

- اگر فایل را از پوشه ی scripts با استفاده از wildcard فراخوانی کنید، این روش به صورت پیش فرض فایل ها را بر اساس ترتیب حروف الفبا فراخوانی می کند و این احتمالا مد نظر شما نیست چرا که فایل های CSS و JavaScript معمولا می بایست به ترتیب خاصی (غیر ترتیب حروف الفبا) به bundle

ا ضافه شوند. البته می توانید با پیاده سازی IBundleOrderer تا حدی فایل ها را طبق ترتیب دلخواه خود به پروژه اضافه کنید اما با این وجود فراخوانی صریح فایل ها احتمال رخداد خطاها را بیشتر کاهش می دهد. به عنوان مثال ممکن است در آینده لازم شود asset های جدیدی به یک پوشه ا ضافه کنید که این امر شما مجاب به ویرایش پیاده سازی IBundleOrderer می کند.

- فایل های view ای که به وسیله ی wild card به پوشه ا ضافه می شوند، احتملا در تمامی view هایی که آن bundle را فراخوانی می کنند، گنجانده می شود. حال اگر اسکریپت مرتبط با view مورد نظر به یک bundle اضافه شود، ممکن است در سایر view هایی که آن bundle را فراخوانی می کنند، یک خطای JavaScript رخ دهد.
- فایل های CSS ای که فایل های دیگری را import می کنند، ممکن است آن فایل ها را دو بار لود کنند. برای مثال کد زیر یک bundle ایجاد می کند که بیشتر فایل های jQuery UI theme CSS را دوبار فراخوانی و بارگذاری می نمایند.

```
bundles.Add(new StyleBundle("~/jQueryUI/themes/baseAll")
  .IncludeDirectory("~/Content/themes/base", "*.css"));
```

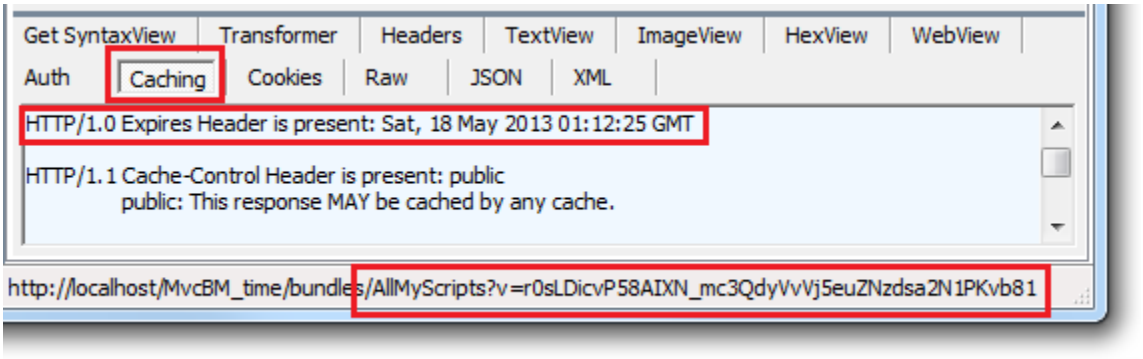
کد "*.css" (کاراکتر * که فایل ها را انتخاب می کند)، تمامی فایل های CSS موجود در پوشه ی مورد نظر، از جمله فایل Content\themes\base\jquery.ui.all.css، را بارگذاری می کند. فایل jquery.ui.all.css خود به نوبه فایل های دیگری را import می نماید.

Bundle caching (ذخیره ی موقت bundle ها در کش)

Bundle ها مقدار HTTP Expires Header و مدت زمانی که بعد از آن response دیگر معتبر نیست را بر روی یک سال پس از ایجاد bundle تنظیم می کنند. اگر به صفحه ی قبلی سایت (صفحه ای که قبلا بارگذاری شده) مراجعه کنید، می بینید که Fiddler هیچ request شرطی جهت واکشی bundle مربوطه از سمت مرورگر IE نمایش نمی دهد. به عبارت دیگر هیچ درخواست HTTP GET از مرورگر IE به سرور فرستاده نشده و پاسخ (response) 304 از سرویس دهنده به مرورگر ارسال نشده است. می توانید با فشردن کلید F5 به مرورگر دستور بدید که برای هر فایل bundle یک request شرطی به سرور ارسال کند (که سبب می شود یک response با کد 304 برای هر bundle از سرور دریافت شود). همچنین می توانید به وسیله ی ^F5 کل صفحه

را بروز رسانی کرده (full refresh) که سبب ارسال response با کد 200 از سرور به ازای هر bundle به مرورگر می شود.

تصویر زیر تب Caching از صفحه ی مربوط به نمایش اطلاعات response ابزار Fiddler را به نمایش می گذارد:



درخواست

http://localhost/MvcBM_time/bundles/AllMyScripts?v=r0sLDicvP58AIXN_mc3QdyVvVj5euZNzdsa2N1PKvb81 ، در واقع فایل باندل ALLMyScripts را از سرور درخواست کرده و دربردارنده ی متغیر (query string) v=r0sLDicvP58AIXN_mc3QdyVvVj5euZNzdsa2N1PKvb81 می باشد. متغیر نام برده حامل مقدار token است که شناسه ی یکتا برای caching می باشد. تا زمانی که محتوای bundle تغییری نکرده، اپلیکیشن ASP.NET فایل AllMyScripts را با همین token از سرور درخواست می کند. حال اگر فایلی در bundle بروز رسانی شود، مجموعه کتابخانه های بهینه سازی (optimization framework) اپلیکیشن مذکور، یک token جدید تولید کرده و بدین وسیله اطمینان حاصل می کند که request به آخرین فایل bundle دسترسی داشته و آن را واکشی می کند.

اکنون زمانی که ابزار F12 مرورگر IE9 را صدا زده و به صفحه ی قبلی سایت (یک صفحه از سایت که قبلا بارگذاری شده) مراجعه می کنید، مرورگر نام برده به خطا درخواست های شرطی GET که برای هر bundle به سرور فرستاده شده و پاسخ 304 دریافت شده از سرور را نمایش می دهد.

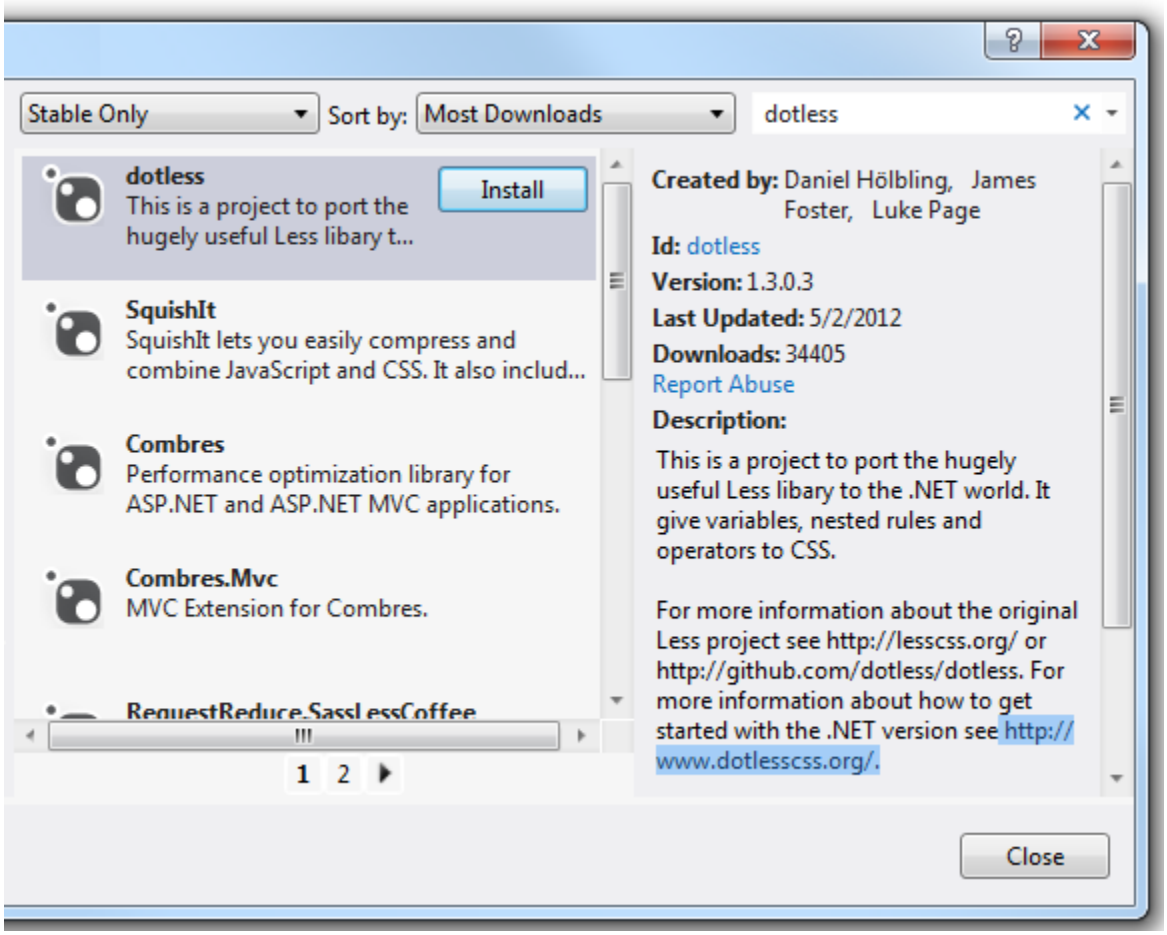
Bundle کردن فایل های Sass، SCSS، CoffeeScript و LESS

کتابخانه و توابع بهینه سازی bundling و minifiaction مکانیزمی جهت پردازش زبان های میانی (intermediate) نظیر SCSS، Sass، LESS یا CoffeeScript ارائه کرده و عملیاتی نظیر minification

(فشرده سازی) را بر روی فایل خروجی (bundle) اعمال می کنند. به طور مثال، جهت افزودن فایل های less.

به پروژه ی MVC 4 خود مراحل زیر را طی نمایید:

1. یک پوشه جهت میزبانی محتوای LESS ایجاد کنید. مثال زیر فایل های less. را داخل پوشه ی Content\MyLess ذخیره می کند.
2. پکیج less. را از مدیریت کننده ی پکیج های NuGet به پروژه ی خود اضافه نمایید.



3. یک کلاس که interface ای به نام IBundleTransform را پیاده سازی می کند، به پروژه اضافه نمایید. برای تبدیل باندل به less.، کد زیر را به پروژه اضافه کنید:

```
using System.Web.Optimization;

public class LessTransform : IBundleTransform
{
```

```
  public void Process(BundleContext context, BundleResponse response)
  {
    response.Content = dotless.Core.Less.Parse(response.Content);
    response.ContentType = "text/css";
  }
}
```

4. یک bundle از فایل های LESS (به وسیله ی new Bundle) و با ساخت آبجکت هایی از کلاس های LessTransform و CssMinify به صورت زیر ایجاد نمایید. کد زیر را به متد RegisterBundles در فایل App_Start\BundleConfig.cs اضافه نمایید.

```
var lessBundle = new Bundle("~/My/Less").IncludeDirectory("~/My", "*.less");
lessBundle.Transforms.Add(new LessTransform());
lessBundle.Transforms.Add(new CssMinify());
bundles.Add(lessBundle);
```

کد زیر را به تمامی view هایی که باندل LESS را فراخوانی می کنند، اضافه نمایید.

```
@Styles.Render("~/My/Less");
```

ملاحظات و نکات مفید در خصوص Bundle

یک روش صحیح که در ایجاد bundle بهتر است رعایت شود، الصاق "bundle" به ابتدای اسم فایل bundle می باشد. با پیروی از این رهنمود می توان به راحتی از رخداد تداخل آدرس دهی یا routing conflict جلوگیری کرد.

اگر یک فایل مقیم در bundle تغییر کند، توابع مربوط ASP.NET به صورت خودکار یک token جدید ویژه ی آن bundle تولید کرده و آن را در قالب متغیر query string با request ارسال می کند. در این صورت زمانی که کاربر صفحه ی دربردارنده ی bundle را از سرور درخواست کند، کل آن bundle به همراه صفحه ی میزبان آن بارگذاری می شود. در ورژن های قدیمی کد markup/html که در آن هر asset (منبع) به صورت جداگانه لیست می شد، تنها فایل بروز رسانی شده از سرور دانلود می شد. آن دسته از منابع و asset هایی که به طور مکرر محتوای آن ها بروز رسانی می شوند را نباید همراه سایر فایل های مورد نیاز bundle کرد.

Bundling و minification بیشترین تاثیر خود در بهینه سازی اپلیکیشن را با کاهش زمان مورد نیاز برای بارگذاری صفحه ی اول سایت نشان می دهند. زمانی که صفحه ی وب از سرور درخواست می شود، مرورگر فایل های asset (نظیر CSS، JavaScript و عکس) را در حافظه ی نهان خود کش می کند. به این دلیل زمانی که کاربر همان صفحه را دوباره درخواست می کند یا از صفحاتی از همان سایت که asset های یکسان را فراخوانی

می کنند، بازدید می نماید، bundling و minification هیچ تاثیری در افزایش کارایی و سرعت بالا آوردن سایت ندارند. اگر expires header را در asset های خود به درستی مقداردهی نکنید و همزمان از minification و bundling نیز استفاده نکنید، مرورگر فایل های نام برده را با گذشت چند روز منقضی علامت گذاری کرده و برای بارگذاری هر asset یک درخواست اعتبارسنجی منبع (validation request) به سرور می فرستد. در این شرایط استفاده از دو قابلیت نام برده می تواند در سرعت بارگذاری اولین صفحه از سایت تاثیر بسزایی داشته باشند.

محدودیت مرورگر در خصوص ارسال تنها 6 درخواست در آن واحد به یک سرور (hostname) را می توان با استفاده از CDN تا حدی برطرف کرد. چرا که اسم سرور CDN از اسم سرور میزبان سایت (hosting site) متفاوت بوده و درخواست هایی که برای asset ها از CDN به سرور فرستاده می شود، با محدودیت ارسال 6 درخواست در آن واحد از مرورگر به سرور میزبان سایت (hosting environment) مغایرتی ندارد. علاوه بر آن CDN با قرار دادن محتوای مورد نیاز به صورت موقت در سرورهای نزدیک به کاربر، مزیت های edge caching و package caching را به ارمغان می آورد (edge caching به ذخیره ی داده های مورد نیاز در سرورهای مخصوص caching نزدیک تر به کاربر گفته می شود).

صفحاتی از اپلیکیشن که فایل های bundle را درخواست می کنند، بایستی آن ها را بخش بندی کرده و به partition های مجزا تبدیل کنند. برای مثال، template پیش فرض ASP.NET MVC برای اپلیکیشن تحت وب، یک فایل باندل jQuery validation مجزا از پوشه ی jQuery ایجاد می کند چرا که view های پیش فرض مقادیر ورودی نداشته، هیچ مقداری را نیز post نمی کنند و به همین دلیل bundle حاوی کد اعتبارسنجی (validation) را احتیاج ندارند (شامل نمی شوند).

namepace یا پوشه System.Web.Optimization داخل System.Web.Optimization.DLL پیاده سازی شده است. این dll برای کاهش سایز فایل ها و minification از کتابخانه ی (WebGrease.dll) WebGrease استفاده می کند که آن نیز خود فایل Antlr3.Runtime.dll را فراخوانی می نماید.