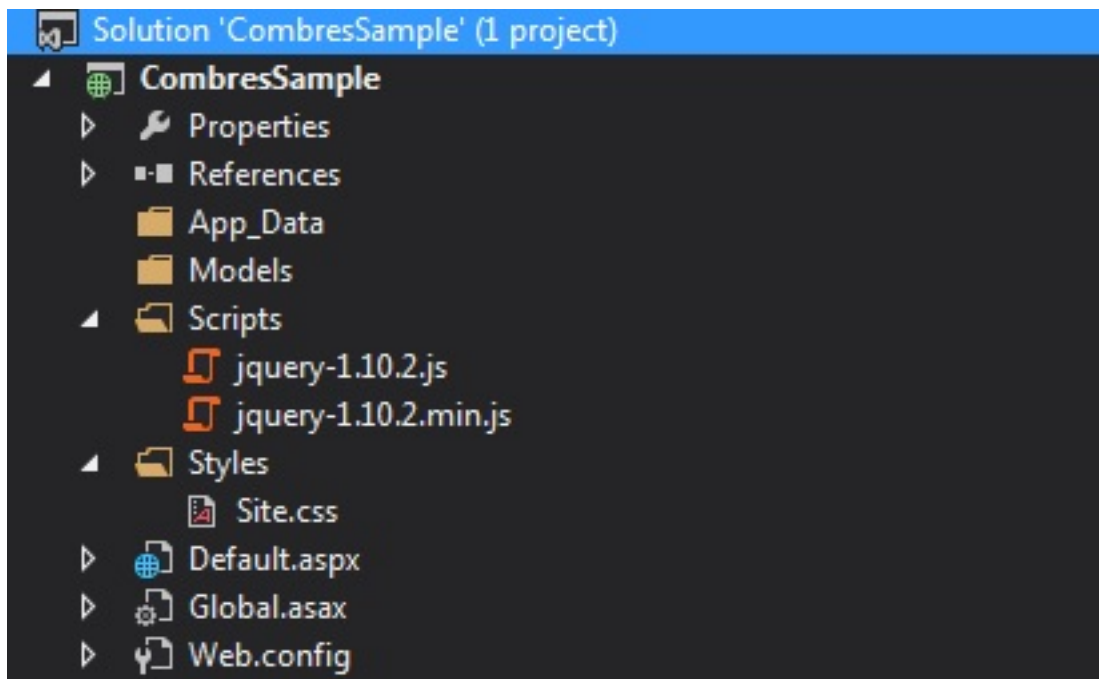


یکی از مواردی که در توسعه وب نقش مهمی دارد، بهینه سازی فایل های js و css است که با فشرده سازی و کش کردن آنها می توان سرعت بارگذاری را تا حد چشمگیری افزایش داد. برای درک بهتر، به مثال زیر توجه کنید.
یک پروژه ساده را ایجاد می کنیم و فایل های CSS و js را مانند شکل زیر، به آن اضافه می کنیم:



طبق تصویر فایل ها را به صفحه ای که ساختیم اضافه می کنیم:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>combress</title>
  <link href="Styles/Site.css" rel="stylesheet" />
  <script src="Scripts/jquery-1.10.2.js"></script>
  <script src="Scripts/jquery-1.10.2.min.js"></script>
</head>
<body>
  <form id="form1" runat="server">
    <div>
    </div>
  </form>
</body>
</html>
```

پروژه را اجرا کرده و توسط افزونه ی [firebug](#) درخواست هایی را که از سرور شده است، بررسی می کنیم. مشاهده خواهید کرد که به ازای هر فایل، یک درخواست به سرور ارسال شده و هیچکدام از فایل ها توسط وب سرور فشرده سازی نشده اند و اطلاعاتی در مورد کش، به هدر آنها اضافه نشده است.

+ GET Site.css	200 OK
+ GET jquery-1.10.2.js	200 OK
+ GET jquery-1.10.2.min.js	200 OK

برای رفع این موارد، روش های گوناگونی وجود دارد که امروز قصد داریم این کار را توسط کتابخانه Combress انجام دهیم ! **نصب کتابخانه Combres**

شما می توانید با استفاده از [nuget](#) این کتابخانه را به پروژه خود اضافه کنید. **ایجاد فایل تنظیمات**
پس از نصب کتابخانه، فایلی با نام combres.xml در فولدر app_data ساخته می شود که تمامی فعالیت های کتابخانه براساس آن انجام می شود و ساختار آن بصورت زیر است:

```
<?xml version="1.0" encoding="utf-8" ?>
<combres xmlns='urn:combres'>
  <filters>
    <filter type="Combres.Filters.FixUrlsInCssFilter, Combres" />
  </filters>
  <resourceSets url="~/combres.axd"
    defaultDuration="30"
    defaultVersion="auto"
    defaultDebugEnabled="false"
    defaultIgnorePipelineWhenDebug="true"
    localChangeMonitorInterval="30"
    remoteChangeMonitorInterval="60">
    <resourceSet name="siteCss" type="css">
      <resource path="~/content/Site.css" />
      <resource path="~/content/anotherCss.css" />
      <resource path="~/scripts/yetAnotherCss.css" />
    </resourceSet>
    <resourceSet name="siteJs" type="js">
      <resource path="~/scripts/jquery-1.4.4.js" />
    </resourceSet>
  </resourceSets>
```

```
<resource path="~/scripts/anotherJs.js" />
<resource path="~/scripts/yetAnotherJs.js" />
</resourceSet>
</resourceSets>
</combres>
```

ResourceSet : با استفاده از هر ResourceSet می‌توانید آن مجموعه فایل را در یک درخواست از سرور دریافت کنید. پارامتر **url** : برای تولید لینک فایل‌ها از آن استفاده میکند.

پارامتر **defaultDuration** : این عدد به تعداد روزهای پیشفرض برای کش کردن فایل‌ها اشاره میکند.

پارامتر **defaultVersion** : در صورتی که مقدار آن auto باشد به ازای هر تغییر، آدرس جدیدی برای فایل موردنظر ایجاد میشود.

پارامتر **defaultDebugEnabled** : با استفاده از آن می‌توانید debug mode را مشخص کنید. در صورتیکه مقدار آن auto باشد، این مقدار مستقیماً از وب‌کانفیگ خوانده میشود.

مقادیر پیش فرض برای تمامی ResourceSet‌ها استفاده می‌شود و در صورت نیاز میتوان این مقادیر را برای هر ResourceSet بازنویسی کرد. فیلترها برای اعمال تغییراتی در فایل js و CSS استفاده می‌شوند که باید به این شکل معرفی شوند. در قسمت بعد با فیلترها بیشتر آشنا میشویم.

فایل cobmres.xml را به منظور استفاده در پروژه به صورت زیر تغییر می‌دهیم.

```
<?xml version="1.0" encoding="utf-8" ?>
<combres xmlns='urn:combres'>
  <filters>
    <filter type="Combres.Filters.FixUrlsInCssFilter, Combres" />
  </filters>
  <resourceSets url="~/combres.axd"
    defaultDuration="30"
    defaultVersion="auto"
    defaultDebugEnabled="false"
    defaultIgnorePipelineWhenDebug="true"
    localChangeMonitorInterval="30"
    remoteChangeMonitorInterval="60">
    <resourceSet name="siteCss" type="css">
      <resource path="~/Styles/Site.css" />
    </resourceSet>
    <resourceSet name="siteJs" type="js">
      <resource path="~/Scripts/jquery-1.10.2.js" />
      <resource path="~/Scripts/jquery-1.10.2.min.js" />
    </resourceSet>
  </resourceSets>
</combres>
```

اگر از نیوگت برای نصب کتابخانه استفاده کرده باشید تغییرات مورد نیاز در فایل وب کانفیگ به صورت خودکار اعمال می‌شود؛ در غیر اینصورت باید قسمتهای زیر را به آن اضافه کنید.

```
<configuration>
  <configSections>
    <section name="combres" type="Combres.ConfigSectionSetting, Combres, Version=2.2, Culture=neutral,
    PublicKeyToken=1ca6b37997dd7536" />
  </configSections>
  <system.web>
    <pages>
      <namespaces>
        <add namespace="Combres" />
      </namespaces>
    </pages>
  </system.web>
  <combres definitionUrl="~/App_Data/combres.xml" />
  <appSettings>
    <add key="CombresSectionName" value="combres" />
  </appSettings>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="AjaxMin" publicKeyToken="21ef50ce11b5d80f" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.84.4790.14405" newVersion="4.84.4790.14405" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

حال باید Route مربوط به Combres را به RouteTable اضافه کنیم. در صورتیکه از نیوگت استفاده کرده باشید، کلاسی به فولدر app_start اضافه شده است که با استفاده از [WebActivator](#) اینکار را در Application_Start انجام میدهد؛ در غیر اینصورت باید به صورت زیر این کار را انجام دهیم.

```
protected void Application_Start(object sender, EventArgs e)
{
    RouteTable.Routes.AddCombresRoute("Combres Route");
}
```

بعد از انجام مراحل قبل، نوبت به آن رسیده است که از لینکهای combres در صفحات خود استفاده کنیم. شیوه استفاده از آن در وب فرم به این صورت است:

```
<%@ Import Namespace="Combres" %>
<head runat="server">
    <%= WebExtensions.CombresLink("siteCss") %>
    <%= WebExtensions.CombresLink("siteJs") %>
</head>
```

و در MVC به صورت زیر می باشد:

```
<%= Url.CombresLink("siteCss") %>
<%= Url.CombresLink("siteJs") %>
```

در هر دو مورد نام ResourceSet برای تولید لینک به متد CombresLink ارسال میشود. پس از اجرای مجدد برنامه و با استفاده از firebug خواهیم دید که به ازای هر Resource Set ، یک درخواست به سرور ارسال شده است و حجم فایلها به صورت چشمگیری کاهش یافته است و اطلاعات مربوط به کش هم به آن اضافه شده است.

+	GET /combres.axd/siteCss/13518664!	200 OK
+	GET /combres.axd/siteJs/119039392!	200 OK

در ادامه می توانید فایل site.css قبلی و فعلی را مقایسه کنید!

[-] GET /combres.axd/siteCss/13518664	200 OK	localhost:55439	220 B
Headers	Response	Cache	
	body{padding-top:50px;padding-bottom:20px}.body-content{padding-left:15px;padding-right:15px;text-align:center;textarea{max-width:280px}.field-validation-error{color:#b94a48}.field-validation-valid{color:#55a868}input.input-validation-error{border:1px solid #b94a48}input[type="checkbox"].input-validation-error{border:0 none}.validation-summary-errors{color:#b94a48}.validation-summary-valid{display:none}		
[+] GET /combres.axd/siteJs/119039392	200 OK	localhost:55439	67.5 KB
[-] GET site.css	200 OK	localhost:55439	463 B
Headers	Response	Cache	
	body { padding-top: 50px; padding-bottom: 20px; } /* Set padding to keep content from hitting the edges */ .body-content { padding-left: 15px; padding-right: 15px; } /* Set width on the form input elements since they're 100% wide by default */ input, select, textarea { max-width: 280px; }		

در قسمت بعد با سازوکار combres و روش استفاده از فیلترها، بیشتر آشنا میشویم.

[CombresProject-67bd3b7299f24c5484edc35537fa990b.zip](#)

نظرات خوانندگان

نویسنده: آرایه

تاریخ: ۱۳۹۳/۰۲/۲۹ ۹:۴۰

ضمن تشکر به خاطر توضیحات خوب، مزیت استفاده از Combres نسبت به امکان bundle و minification خود ASP.NET MVC چیست؟

نویسنده: سامان هاشمی

تاریخ: ۱۳۹۳/۰۲/۳۰ ۱۳:۴۴

شاید عمده مزیت های Combres این چهار مورد زیر باشد:

1- از کتابخانه های مختلفی برای بهینه سازی پشتیبانی میکند مانند Microsoft Ajax Minifier و YUI Compressor for .NET و Google Closure Compiler.

2- تعریف جداگانه اطلاعات کش به ازای هر فایل

3- فیلترها

4- لاگ کردن پردازش های داخلی (به صورت پیش فرض توسط Log4Net انجام میگیرد)

نویسنده: rezaei

تاریخ: ۱۳۹۳/۰۲/۳۰ ۱۷:۳۶

با سلام؛ من تمامی مراحل را که فرمودید را انجام دادم. اما موقع اجرا خطای زیر رو می گیره

```
System.ArgumentException: Illegal characters in path
```

```
@Url.CombresLink("jquery")
```

نویسنده: سامان هاشمی

تاریخ: ۱۳۹۳/۰۲/۳۱ ۷:۵۷

احتمالا تو نوع آدرس دهی یکی از ResourceSet های فایل xml یک مشکلی وجود داره! (بهتر بود نمونه کد یا پروژه اینجا قرار میدادید تا بهتر بتونم کمکتون کنم)

نویسنده: afsharsalar

تاریخ: ۱۳۹۳/۰۳/۱۳ ۱۱:۵۶

برای MVC از [Combres.Mvc](#) استفاده کنید و سپس از دستور

```
@Html.Raw(WebExtensions.CombresLink("siteCss"))
```

نویسنده: rezaei

تاریخ: ۱۳۹۳/۰۶/۱۱ ۲۳:۱۳

با سلام

موقع استفاده یکی از فایل های جاوا اسکریپت توی cache قرار میگیره و تغییرات روش اعمال نمیشه اما وقتی به صورت عادی بدون combres فایل جاوا اسکریپت رو استفاده میکنم این مشکل وجود نداره ولی در مورد فایل های Css همچین مشکلی وجود نداره

ممنون میشم راهنمایی کنید