عنوان: **طراحی افزونه پذیر با ASP.NET MVC 4.x/5.x - قسمت دوم**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Plugin

در مطلب « طراحی افزونه پذیر با ASP.NET MVC 4.x/5.x - قسمت اول » با ساختار کلی یک پروژه‌ی افزونه‌ی پذیر ASP.NET MVC آشنا شدیم. پس از راه اندازی آن و مدتی کار کردن با این نوع پروژه‌ها، این سؤال پیش خواهد آمد که ... خوب، اگر هر افزونه تصاویر یا فایل‌های CSS و JS اختصاصی خودش را بخواهد داشته باشد، چطور؟ موارد عمومی مانند بوت استرپ و جی‌کوئری را می‌توان در پروژه‌ی پایه قرار داد تا تمام افزونه‌ها به صورت یکسانی از آن‌ها استفاده کنند، اما هدف، ماژولار شدن برنامه است و جدا کردن فایل‌های ویژه‌ی هر پروژه، از پروژه‌ای دیگر و همچنین بالا بردن سهولت کار تیمی، با شکستن اجزای یک پروژه به صورت افزونه‌هایی مختلف، بین اعضای یک تیم. در این قسمت نحوه‌ی مدفون سازی انواع فایل‌های استاتیک افزونه‌ها را درون فایل‌های DLL آن‌ها بررسی خواهیم کرد. به این ترتیب دیگر نیازی به ارائه‌ی مجزای آن‌ها و یا کپی کردن آن‌ها در پوشه‌های پروژه‌ی اصلی نخواهد بود.

مدفون سازی فایل‌های CSS و JS هر افزونه درون فایل DLL آن

به solution جاری، یک class library جدید را به نام MvcPluginMasterApp.Common اضافه کنید. از آن جهت قرار دادن کلاس‌های عمومی و مشترک بین افزونه‌ها استفاده خواهیم کرد. برای مثال قصد نداریم کلاس‌های سفارشی و عمومی ذیل را هربار به صورت مستقیم در افزونه‌ای جدید کپی کنیم. کتابخانه‌ی Common، امکان استفاده‌ی مجدد از یک سری کدهای تکراری را در بین افزونه‌ها میسر می‌کند.
این پروژه برای کامپایل شدن نیاز به بسته‌ی نیوگت ذیل دارد:

```
PM> install-package Microsoft.AspNet.Web.Optimization
```

همچنین باید به صورت دستی، در قسمت ارجاعات پروژه، ارجاعی را به اسمبلی استاندارد System.Web نیز به آن اضافه نمائید.
پس از این مقدمات، کلاس ذیل را به این پروژه‌ی class library جدید اضافه کنید:

```
using System.Collections.Generic;
using System.IO;
using System.Reflection;
using System.Text;
using System.Web.Optimization;

namespace MvcPluginMasterApp.Common.WebToolkit
{
    public class EmbeddedResourceTransform : IBundleTransform
    {
        private readonly IList<string> _resourceFiles;
        private readonly string _contentType;
        private readonly Assembly _assembly;

        public EmbeddedResourceTransform(IList<string> resourceFiles, string contentType, Assembly
assembly)
        {
            _resourceFiles = resourceFiles;
            _contentType = contentType;
            _assembly = assembly;
        }

        public void Process(BundleContext context, BundleResponse response)
        {
            var result = new StringBuilder();

            foreach (var resource in _resourceFiles)
            {
                using (var stream = _assembly.GetManifestResourceStream(resource))
                {
                    if (stream == null)
                    {
                        throw new KeyNotFoundException(string.Format("Embedded resource key: '{0}' not
found in the '{1}' assembly.", resource, _assembly.FullName));
                    }
```

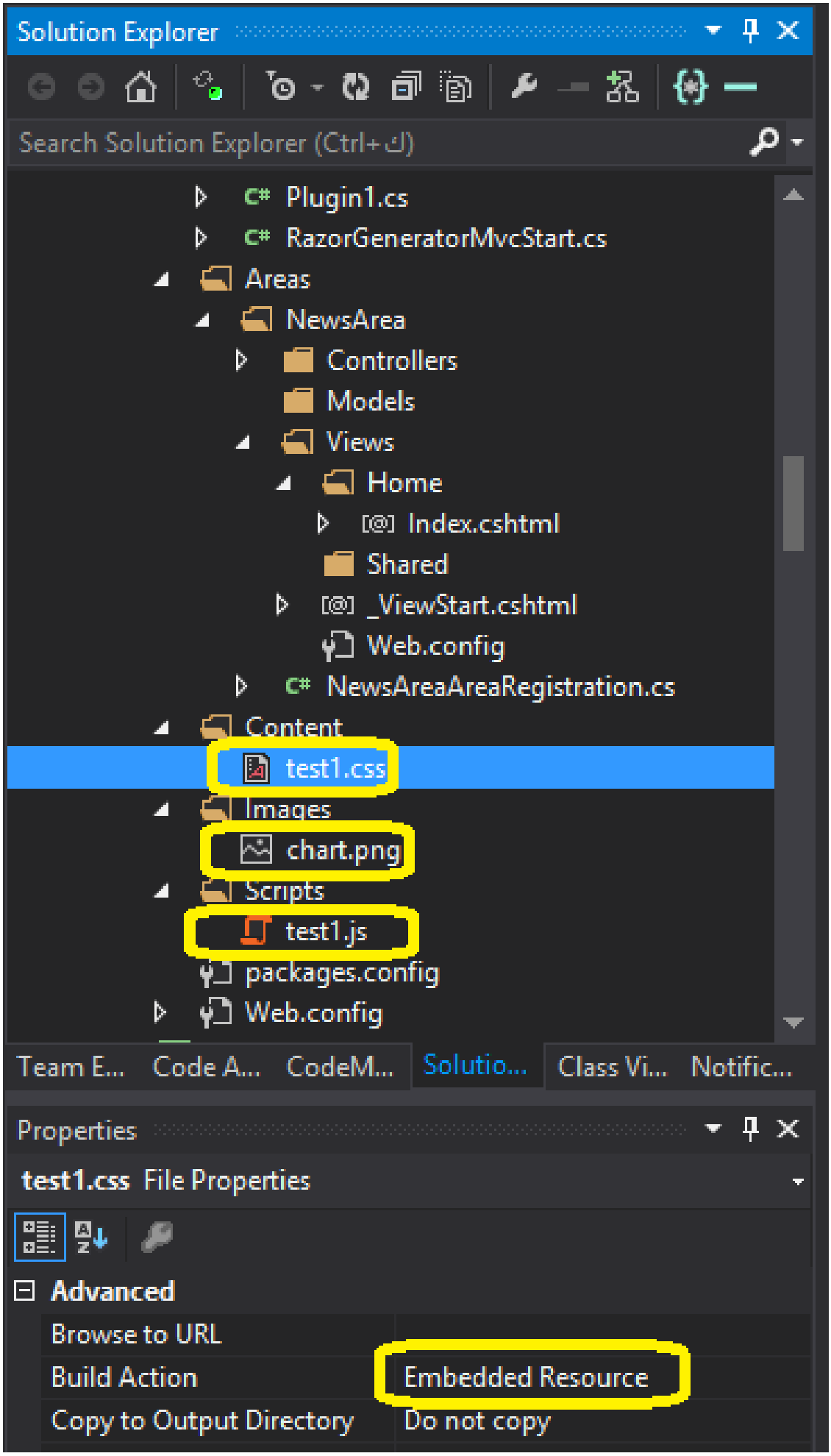
```
                    using (var reader = new StreamReader(stream))
                    {
                        result.Append(reader.ReadToEnd());
                    }
                }
            }

            response.ContentType = _contentType;
            response.Content = result.ToString();
        }
    }
}
```

اگر با سیستم minification & bundling کار کرده باشید، با تعاریفی مانند new Bundle("~/Plugin1/Scripts") آشنا هستید.
سازنده‌ی کلاس Bundle، پارامتر دومی را نیز می‌پذیرد که از نوع IBundleTransform است. با پیاده سازی اینترفیس IBundleTransform می‌توان محل ارائه‌ی فایل‌های استاتیک CSS و JS را بجای فایل سیستم متداول و پیش فرض، به منابع مدفون شده‌ی در اسمبلی جاری هدایت و تنظیم کرد.
کلاس فوق در اسمبلی معرفی شده به آن، توسط متد GetManifestResourceStream به دنبال فایل‌ها و منابع مدفون شده گشته و سپس محتوای آن‌ها را بازگشت می‌دهد.
اکنون برای استفاده‌ی از آن، به پروژه‌ی MvcPluginMasterApp.Plugin1 مراجعه کرده و ارجاعی را به پروژه‌ی MvcPluginMasterApp.Common فوق اضافه نمائید. سپس در فایل Plugin1.cs، متد RegisterBundles آن‌را به نحو ذیل تکمیل کنید:

```
namespace MvcPluginMasterApp.Plugin1
{
    public class Plugin1 : IPlugin
    {
        public EfBootstrapper GetEfBootstrapper()
        {
            return null;
        }

        public MenuItem GetMenuItem(RequestContext requestContext)
        {
            return new MenuItem
            {
                Name = "Plugin 1",
                Url = new UrlHelper(requestContext).Action("Index", "Home", new { area = "NewsArea" })
            };
        }

        public void RegisterBundles(BundleCollection bundles)
        {
            var executingAssembly = Assembly.GetExecutingAssembly();
            // Mostly the default namespace and assembly name are the same
            var assemblyNameSpace = executingAssembly.GetName().Name;
            var scriptsBundle = new Bundle("~/Plugin1/Scripts",
                new EmbeddedResourceTransform(new List<string>
                {
                    assemblyNameSpace + ".Scripts.test1.js"
                }, "application/javascript", executingAssembly));
            if (!HttpContext.Current.IsDebuggingEnabled)
            {
                scriptsBundle.Transforms.Add(new JsMinify());
            }
            bundles.Add(scriptsBundle);
            var cssBundle = new Bundle("~/Plugin1/Content",
                new EmbeddedResourceTransform(new List<string>
                {
                    assemblyNameSpace + ".Content.test1.css"
                }, "text/css", executingAssembly));
            if (!HttpContext.Current.IsDebuggingEnabled)
            {
                cssBundle.Transforms.Add(new CssMinify());
            }
            bundles.Add(cssBundle);
            BundleTable.EnableOptimizations = true;
        }

        public void RegisterRoutes(RouteCollection routes)
        {
        }
```

```
        public void RegisterServices(IContainer container)
        {
        }
    }
}
```

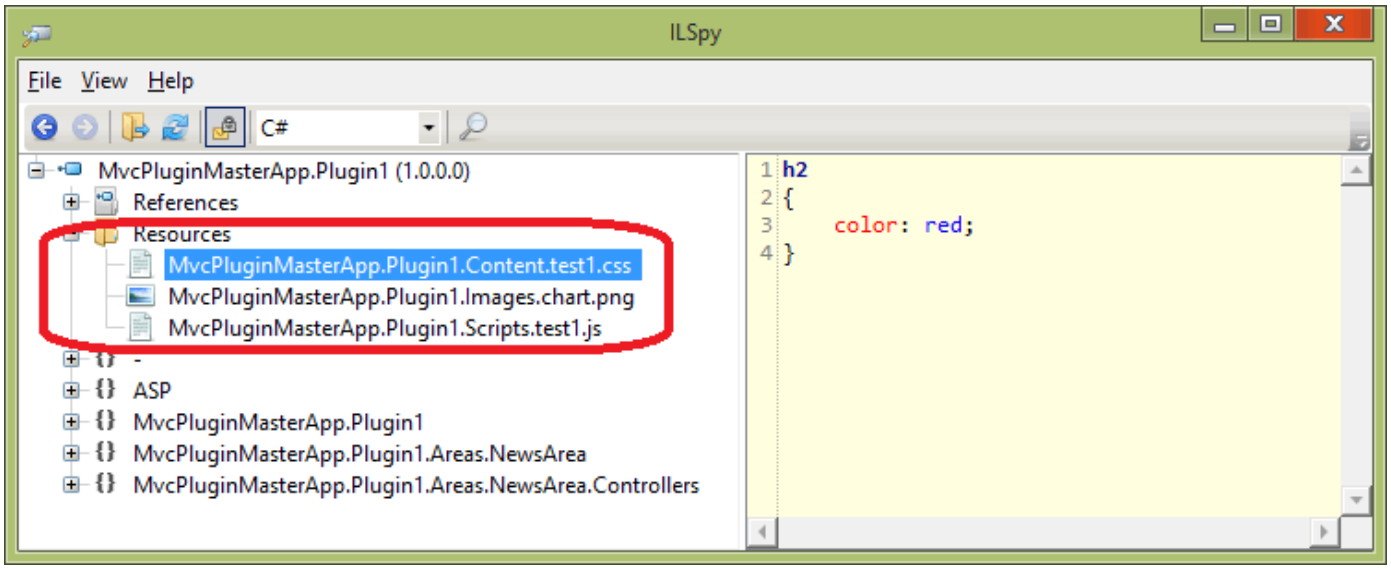
در اینجا نحوه‌ی کار با کلاس سفارشی EmbeddedResourceTransform را مشاهده می‌کنید. ابتدا فایل‌های js و سپس فایل‌های css برنامه به سیستم Bundling برنامه اضافه شده‌اند.
این فایل‌ها به صورت ذیل در پروژه تعریف گردیده‌اند:


Solution Explorer
Search Solution Explorer (Ctrl+ک)
Plugin1.cs
RazorGeneratorMvcStart.cs
Areas
NewsArea
Controllers
Models
Views
Home
Index.cshtml
Shared
_ViewStart.cshtml
Web.config
NewsAreaAreaRegistration.cs
Content
test1.css
Images
chart.png
Scripts
test1.js
packages.config
Web.config
Team E...
Code A...
CodeM...
Solutio...
Class Vi...
Notific...
Properties
test1.css File Properties
Advanced
Browse to URL
Build Action
Embedded Resource
Copy to Output Directory
Do not copy

همانطور که مشاهده می‌کنید، باید به خواص هر کدام مراجعه کرد و سپس Build action آن‌ها را به embedded resource تغییر داد، تا در حین کامپایل، به صورت خودکار در قسمت منابع اسمبلی ذخیره شوند.

یک نکته‌ی مهم

اینبار برای مسیردهی منابع، باید بجای / فایل سیستم، از «نقطه» استفاده کرد. زیرا منابع با نام‌هایی مانند namespace.folder.name در قسمت resources یک اسمبلی ذخیره می‌شوند:



مدفون سازی تصاویر ثابت هر افزونه درون فایل DLL آن

مجددا به اسمبلی مشترک MvcPluginMasterApp.Common مراجعه کرده و اینبار کلاس جدید ذیل را به آن اضافه کنید:

```
using System;
using System.Collections.Generic;
using System.Reflection;
using System.Web;
using System.Web.Routing;

namespace MvcPluginMasterApp.Common.WebToolkit
{
    public class EmbeddedResourceRouteHandler : IRouteHandler
    {
        private readonly Assembly _assembly;
        private readonly string _resourcePath;
        private readonly TimeSpan _cacheDuration;

        public EmbeddedResourceRouteHandler(Assembly assembly, string resourcePath, TimeSpan
cacheDuration)
        {
            _assembly = assembly;
            _resourcePath = resourcePath;
            _cacheDuration = cacheDuration;
        }

        IHttpHandler IRouteHandler.GetHttpHandler(RequestContext requestContext)
        {
            return new EmbeddedResourceHttpHandler(requestContext.RouteData, _assembly, _resourcePath,
_cacheDuration);
        }
    }
```

```
    public class EmbeddedResourceHttpHandler : IHttpHandler
    {
        private readonly RouteData _routeData;
        private readonly Assembly _assembly;
        private readonly string _resourcePath;
        private readonly TimeSpan _cacheDuration;

        public EmbeddedResourceHttpHandler(
            RouteData routeData, Assembly assembly, string resourcePath, TimeSpan cacheDuration)
        {
            _routeData = routeData;
            _assembly = assembly;
            _resourcePath = resourcePath;
            _cacheDuration = cacheDuration;
        }

        public bool IsReusable
        {
            get { return false; }
        }

        public void ProcessRequest(HttpContext context)
        {
            var routeDataValues = _routeData.Values;
            var fileName = routeDataValues["file"].ToString();
            var fileExtension = routeDataValues["extension"].ToString();

            var manifestResourceName = string.Format("{0}.{1}.{2}", _resourcePath, fileName,
fileExtension);
            var stream = _assembly.GetManifestResourceStream(manifestResourceName);
            if (stream == null)
            {
                throw new KeyNotFoundException(string.Format("Embedded resource key: '{0}' not found in
the '{1}' assembly.", manifestResourceName, _assembly.FullName));
            }

            context.Response.Clear();
            context.Response.ContentType = "application/octet-stream";
            cacheIt(context.Response, _cacheDuration);
            stream.CopyTo(context.Response.OutputStream);
        }

        private static void cacheIt(HttpResponse response, TimeSpan duration)
        {
            var cache = response.Cache;

            var maxAgeField = cache.GetType().GetField("_maxAge", BindingFlags.Instance |
BindingFlags.NonPublic);
            if (maxAgeField != null) maxAgeField.SetValue(cache, duration);

            cache.SetCacheability(HttpCacheability.Public);
            cache.SetExpires(DateTime.Now.Add(duration));
            cache.SetMaxAge(duration);
            cache.AppendCacheExtension("must-revalidate, proxy-revalidate");
        }
    }
}
```

تصاویر پروژه‌ی افزونه نیز به صورت embedded resource در اسمبلی آن قرار خواهند گرفت. به همین جهت باید سیستم مسیریابی را پس درخواست رسیده‌ی جهت نمایش تصاویر، به منابع ذخیره شده‌ی در اسمبلی آن هدایت نمود. اینکار را با پیاده سازی یک IRouteHandler سفارشی، می‌توان به نحو فوق مدیریت کرد.

این IRouteHandler، نام و پسوند فایل را دریافت کرده و سپس به قسمت منابع اسمبلی رجوع، فایل مرتبط را استخراج و سپس بازگشت می‌دهد. همچنین برای کاهش سربار سیستم، امکان کش شدن منابع استاتیک نیز در آن درنظر گرفته شده‌است و هدرهای خاص caching را به صورت خودکار اضافه می‌کند.

سیستم bundling نیز هدرهای کش کردن را به صورت خودکار و توکار اضافه می‌کند.

اکنون به تعاریف Plugin1 مراجعه کنید و سپس این IRouteHandler سفارشی را به نحو ذیل به آن معرفی نمائید:

```
namespace MvcPluginMasterApp.Plugin1
{
    public class Plugin1 : IPlugin
    {
```

```
        public void RegisterRoutes(RouteCollection routes)
        {
            //todo: add custom routes.

            var assembly = Assembly.GetExecutingAssembly();
            // Mostly the default namespace and assembly name are the same
            var nameSpace = assembly.GetName().Name;
            var resourcePath = string.Format("{0}.Images", nameSpace);

            routes.Insert(0,
                new Route("NewsArea/Images/{file}.{extension}",
                    new RouteValueDictionary(new { }),
                    new RouteValueDictionary(new { extension = "png|jpg" }),
                    new EmbeddedResourceRouteHandler(assembly, resourcePath, cacheDuration:
TimeSpan.FromDays(30))
                ));
        }
    }
}
```

در مسیریابی تعریف شده، تمام درخواست‌های رسیده‌ی به مسیر /Images **NewsArea** به EmbeddedResourceRouteHandler هدایت می‌شوند.
مطابق تعریف آن، file و extension به صورت خودکار جدا شده و توسط routeData.Values در متد ProcessRequest کلاس EmbeddedResourceHttpHandler قابل دسترسی خواهند شد.
پسوندهایی که توسط آن بررسی می‌شوند از نوع png یا jpg تعریف شده‌اند. همچنین مدت زمان کش کردن هر منبع استاتیک تصویری به یک ماه تنظیم شده‌است.

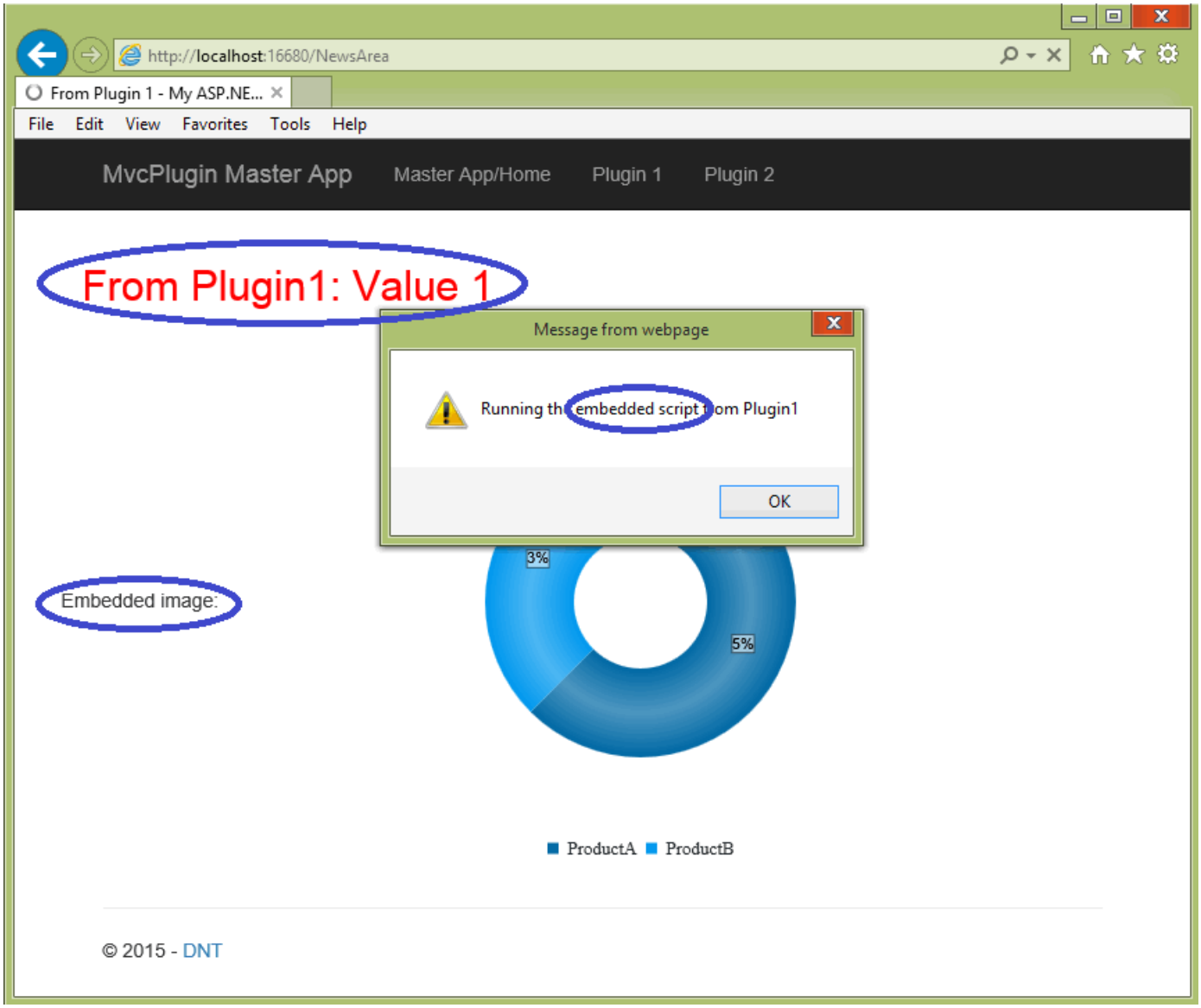
استفاده‌ی نهایی از تنظیمات فوق در یک View افزونه

پس از اینکه تصاویر و فایل‌های css و js را به صورت embedded resource تعریف کردیم و همچنین تنظیمات مسیریابی و bundling خاص آن‌ها را نیز مشخص نمودیم، اکنون نوبت به استفاده‌ی از آن‌ها در یک View است:

```
@{
    ViewBag.Title = "From Plugin 1";
}
@Styles.Render("~/Plugin1/Content")

<h2>@ViewBag.Message</h2>

<div class="row">
    Embedded image:
    <img src="@Url.Content("~/NewsArea/Images/chart.png")" alt="clock" />
</div>

@section scripts
{
    @Scripts.Render("~/Plugin1/Scripts")
}
```

در اینجا نحوه‌ی تعریف فایل‌های CSS و JS ارائه شده‌ی توسط سیستم Bundling را مشاهده می‌کنید.
همچنین مسیر تصویر مشخص شده‌ی در آن، اینبار یک **NewsArea** اضافه‌تر دارد. فایل اصلی تصویر، در مسیر Images/chart.png قرار گرفته‌است اما می‌خواهیم این درخواست‌ها را به مسیریابی جدید { extension}.{file}/Images/ **NewsArea** هدایت کنیم.
بنابراین نیاز است به این نکته نیز دقت داشت.

اینبار اگر برنامه را اجرا کنیم، می‌توان به سه نکته در آن دقت داشت:



الف) alert اجرا شده از فایل js مدفون شده خوانده شده‌است.
ب) رنگ قرمز متن (تگ h2) از فایل css مدفون شده، گرفته شده‌است.
ج) تصویر نمایش داده شده، همان تصویر مدفون شده‌ی در فایل DLL برنامه است.
و هیچکدام از این فایل‌ها، به پوشه‌های پروژه‌ی اصلی برنامه، کپی نشده‌اند.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

MvcPluginMasterApp-Part2.zip

نظرات خوانندگان

نویسنده: رضا ح
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۵:۲۸

سلام.
یه مشکلی برای من پیش اومده که هنگام استفاده از روتر در افزونه وقتی Home رو می‌زنم به Home پروژه می‌ره نه افزونه و وقتی یه اسم دیگه برای کنترلر افزونه می‌زنم خطا می‌ده که نتونسته ویوها رو پیدا کنه. namespaceهای روترها رو هم زدم ولی کار نمی‌کنه.
ممنون می‌شم اگه کمک کنین

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۶:۱۰

در قسمت اول ، لینک به منوی Area جاری به این صورت تعریف شد:

```
public MenuItem GetMenuItem(RequestContext requestContext)
        {
            return new MenuItem
            {
                Name = "Plugin 1",
                Url = new UrlHelper(requestContext).Action("Index", "Home", new { area = "NewsArea" })
            };
        }
```

در این لینک، ذکر نام صحیح area جاری، در آدرس ساخته شده الزامی است. اساسا کلیه لینک‌های ختم به هر area در ASP.NET MVC باید دارای قسمت الزامی {"......name....." = area new} باشند.
همین نکته در پلاگین دوم هم بکار رفته‌است ({"**ArticlesArea**" = area new }).

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۱/۲۸ ۱۹:۵۶

سلام،
خیلی ممنون واقعاً از خوندن این سه قسمت لذت بردم. خیلی ممنون
طبق مطالب گفته شده پروژه رو ایجاد کردم اما هنگام اجرا استثناء FileNotFoundException صادر میشه:

```
at MvcPluginMasterApp.Plugin1.Plugin1.RegisterRoutes(RouteCollection routes)
   at MvcPluginMasterApp.PluginsStart.Start() in c:\Users\Sirwan-Afifi\Desktop\MvcPluginMasterApp-
Part2\MvcPluginMasterApp\App_Start\PluginsStart.cs:line 20
```

کدهای این قسمت هم رو دانلود و اجرا کردم، باز هم همین خطا رو دریافت کردم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۸ ۲۰:۱۱

- لطفا stack trace کامل را ارسال کنید.
- اگر در PluginsStart پیام FileNotFoundException را دریافت می‌کنید، احتمالا یکی از وابستگی‌ها و ارجاعات پروژه یا افزونه‌ها، در پوشه‌ی bin برنامه وجود ندارند یا کپی نشده‌اند.

نکته 1

```
Copy "$(ProjectDir)$(OutDir)$(TargetName).*" "$(SolutionDir)MvcPluginMasterApp\bin\"
```

دستور post build event عنوان شده در قسمت اول ، فقط اسمبلی‌های اصلی افزونه را به پوشه‌ی bin پروژه‌ی اصلی کپی می‌کند. اگر این افزونه ارجاعات بیشتری دارد که در پروژه‌ی اصلی وجود ندارند، باید کل پوشه‌ی bin افزونه را کپی کنید و نه فقط فایل‌های اصلی آن‌را.
برای مثال فایل **MvcPluginMasterApp.Common** هم باید به پوشه‌ی bin کپی شود.

نکته 2
اگر پیام FileNotFoundException توسط استراکچرمپ صادر می‌شود، نیاز است inner exception آن‌را بررسی کنید. اصل خطای رخ داده را در inner exception ارائه می‌دهد.

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۱/۲۸ ۲۱:۵۳

این مورد رو فراموش کرده بودم :)
در پروژه MvcPluginMasterApp باید ارجاعی به پروژه MvcPluginMasterApp.Common داشته باشیم.

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۵:۲۳

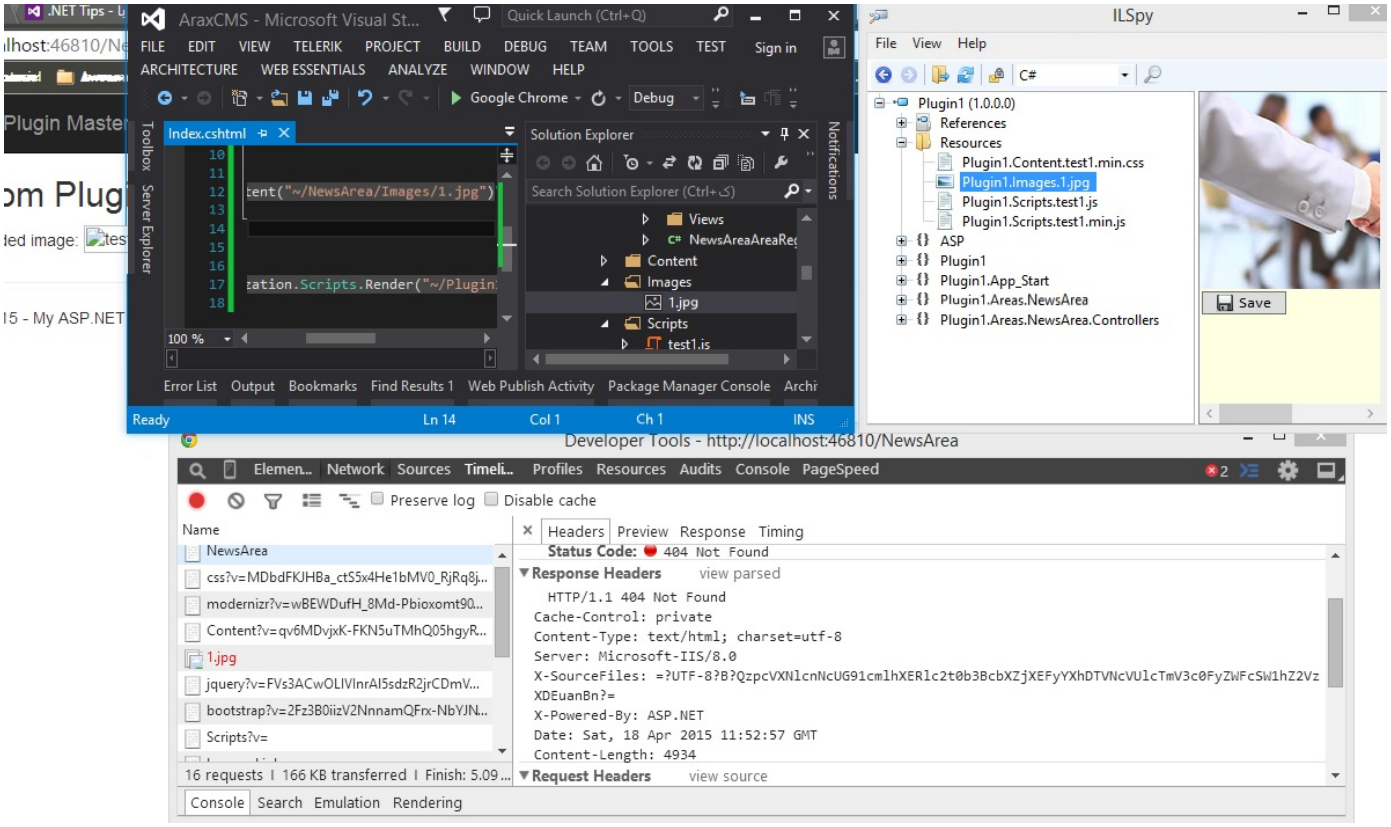
سلام مرسی از آموزش برای من فایل‌های CSS و JS درست کار میکنن ولی عکس رو ارور 404 میده نمی‌دونم مشکل از کجاست

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۵:۴۵

برگه‌ی web developer مرورگر را باز کنید (و یا از فایرباگ استفاده کنید). بعد محتوای response مرتبط با درخواست تصویر را بررسی کنید. اصل خطا در آنجا قابل مشاهده‌است.

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۶:۲۸

ممنون از راهنماییتون . یک اسکرین از صفحات گذاشتم .



نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۶:۴۴

در متد RegisterRoutes ایی که در مثال فوق هست:

```
public void RegisterRoutes(RouteCollection routes)
        {
            //....
            routes.Insert(0,
                new Route("NewsArea/Images/{file}.{extension}",
                    new RouteValueDictionary(new { }),
                    new RouteValueDictionary(new { extension = "png|jpg" }),
                    new EmbeddedResourceRouteHandler(assembly, resourcePath, cacheDuration:
TimeSpan.FromDays(30))
                ));
        }
```

آدرسی‌هایی با فرمت /file **NewsArea/Images** به EmbeddedResourceRouteHandler هدایت می‌شوند.

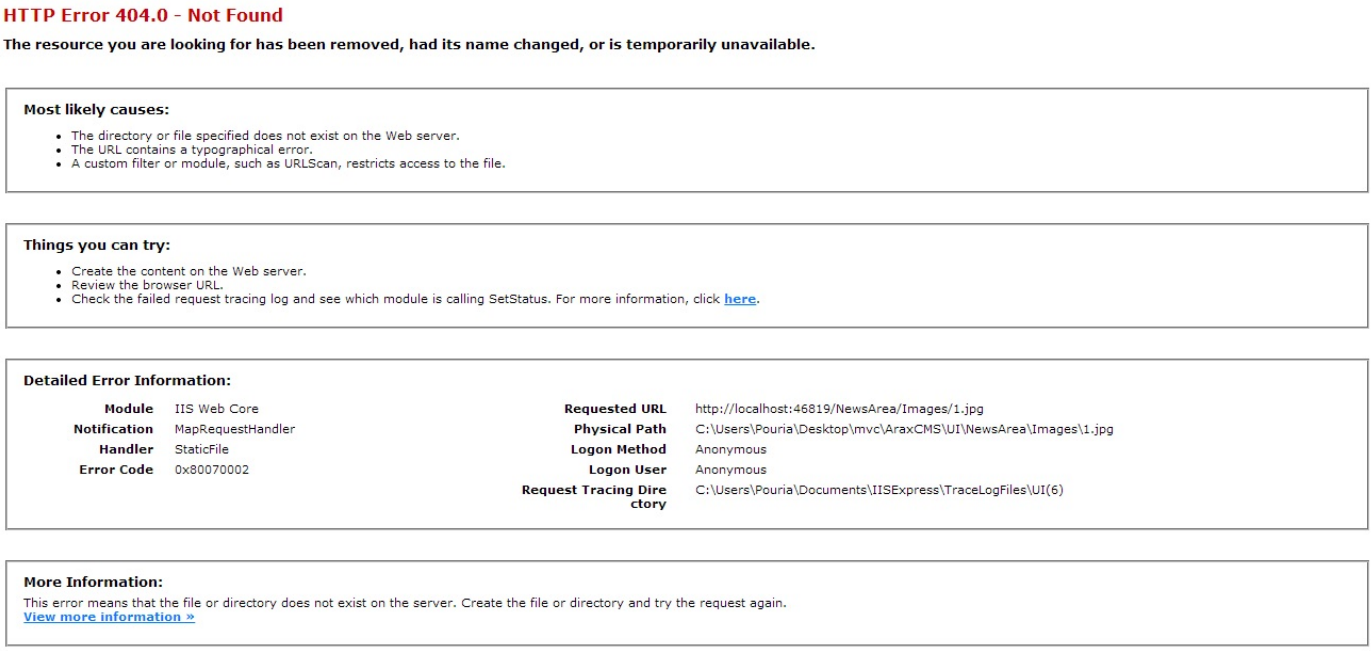
- بررسی کنید آدرس کاملی که به 404 ختم شده چیست؟ آیا آدرس درخواستی با **NewsArea/Images** شروع می‌شود؟
- در برگه‌ی response آن چه خروجی را مشاهده می‌کنید؟

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۶:۵۵

منظورتون رو از برگه ریسپانس متوجه نمیشم .

```
http://localhost:46819/NewsArea/Images/1.jpg
```

```
C:\Users\Pouria\Desktop\mvc\AraxCMS\UI\NewsArea\Images\1.jpg
```



نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱۷:۱۲

- در همان تصویر اولی که ارسال کردید، برگه‌ی اول headers هست و برگه‌ی سوم آن response، که صفحه‌ی زرد رنگ معروف خطاهای ASP.NET را نمایش می‌دهد.
- ابتدا بررسی کنید response ایی که در حالت دیباگ مشاهده می‌کنید چی هست.
- سپس بررسی کنید اصلا متد RegisterRoutes ایی که عنوان شد، فراخوانی می‌شود و مسیریابی آن در سیستم ثبت می‌شود یا خیر؟ (یک break point داخل آن قرار دهید)

اگر فراخوانی نمی‌شود، بررسی کنید آیا فایل‌های پوشه‌ی bin این افزونه، به پوشه‌ی bin پروژه‌ی اصلی کپی شده‌اند یا خیر؟

نویسنده: پوریا انوشیروانی
تاریخ: ۱۳۹۴/۰۲/۰۱ ۱۳:۱

مرسی مشکل یکجای دیگه بود. کلا برنامم با نقطه مشکل داشت و قبل از اینکه کاری انجام بده ارور میداد. دوستانی که مشکل منو داشتن با کد زیر داخل وب کانفیگ حل میشه.

```
<system.webServer>
    <modules  runAllManagedModulesForAllRequests="true">
```

اگه میشه توضیحی بدید کد بالا چی هست و چرا با این درست شد؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۱ ۱۳:۳۰

این کد (در حالت تنظیمات integrated mode) باعث خواهد شد تا تمام درخواست‌های رسیده (منجمله درخواست دریافت فایل‌های استاتیک)، توسط موتور ASP.NET پردازش شود و IIS پیش از آن راسا اقدام نکند.

نویسنده: سید مجید رضویان

تاریخ: ۱۳۹۴/۰۲/۱۳ ۱۳:۱۰

سلام؛
- آیا میشه هرکدوم از پلاگین‌ها برای خودشون web.config داشته باشند؟ چون ممکنه بخواهیم برای هر پلاگین سرویس‌ها و Connection String ها و تنظیمات خودش را داشته باشیم.
- آیا این روش را میشه برای پروژه‌های webform هم استفاده کرد؟ چگونه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۳ ۱۳:۱۷

- خیر. هر پلاگین نهایتا با سیستم اصلی یکی می‌شود و از فایل config اصلی استفاده می‌کند. این‌ها هر کدام یک سیستم جدا نیستند. یک سیستم از همکاری اجزای آن جهت رسیدن به یک هدف مشخص تشکیل می‌شود.
در طراحی ارائه شده، یک رشته‌ی اتصالی بیشتر در کل برنامه وجود ندارد. اطلاعات بیشتر
- خیر. این طراحی از سیستم مسیریابی و همچنین ViewEngine سفارشی خاصی استفاده می‌کند که مختص ASP.NET MVC است.

نویسنده: سید مجید رضویان
تاریخ: ۱۳۹۴/۰۲/۱۴ ۱۰:۱۹

سلام
من میخوام یک مجموعه از applicationهای تحت وب داشته باشم که همگی یک نوع طراحی UI داشته باشند و یک منو کلی داشته باشم که کاربرها بتون از اون به هریک از applicationها هدایت بشوند منتها چند نکته وجود داره :

همه این برنامه‌ها به لحاظ عملکردی یکسری مشترکات دارند (مثلا احراز هویت ، لیست کاربرها و ....)
هر کدوم از این برنامه‌ها باید بتونن مستقل طراحی بشن در عین حال که از یک شکل ui استفاده می‌کنند

به راحتی بتونم یک برنامه به مجموعه برنامه هام اضافه کنم یا یکی ازش کم کنم یا یکی را غیر فعال کنم (باید معماری scalable باشد
در واقع یه چیزی شبیه مجمو.عه نرم افزارهای گوگل
به نظر شما روش ارائه شده در این صفحه بدرد من می‌خوره؟
اگر نه از چه معماری استفاده کنم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۴ ۱۱:۱۶

- کمی وقت بگذارید و این سه قسمت را به همراه نظرات آن‌ها « *یکبار* » مطالعه کنید. پروژه‌ی نهایی آن‌را « *یکبار* » اجرا کنید. به پاسخ سؤالات خودتان خواهید رسید.
- این افزونه‌ها هر کدام اساسا یک سیستم مستقل هستند؛ اما در قالب پروژه‌ی اصلی مفهوم پیدا می‌کنند و در کنار هم قرار می‌گیرند.
- در قسمت اول، در مورد منوی مشترک و نحوه‌ی ثبت منوی خاص یک افزونه در منوی اصلی برنامه بحث شده‌است.
- تمام افزونه‌ها از Layout پروژه‌ی اصلی استفاده می‌کنند و نهایتا سیستم، یک شکل و یک دست است.
- در قسمت سوم، نحوه‌ی استفاده‌ی از بانک اطلاعاتی اصلی برنامه به صورت یکسان و مشترکی در تمام افزونه‌ها بحث شده‌است. بنابراین زمانیکه یک افزونه دسترسی به کل Context دارد، دسترسی به نیازهای مشترک بین افزونه‌ها قابل مدیریت خواهد بود.