عنوان: **یکی کردن اسمبلی‌های یک پروژه‌ی WPF**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۳۰ ۱۸:۳۵
آدرس: www.dotnettips.info
گروه‌ها: WPF

فرض کنید پروژه‌ی WPF شما از چندین پروژه‌ی Class library و اسمبلی‌های جانبی دیگر، تشکیل شده‌است. اکنون نیاز است جهت سهولت توزیع آن، تمام این فایل‌ها را با هم یکی کرده و تبدیل به یک فایل EXE نهایی کنیم. مایکروسافت ابزاری را به نام ILMerge ، برای یک چنین کارهایی تدارک دیده‌است؛ اما این برنامه با WPF سازگار نیست. در ادامه قصد داریم اسمبلی‌های جانبی را تبدیل به منابع مدفون شده در فایل EXE برنامه کرده و سپس آن‌ها را در اولین بار اجرای برنامه، به صورت خودکار بارگذاری و در برنامه مورد استفاده قرار دهیم.

یک مثال جهت بازتولید کدهای این مطلب

الف) یک پروژه‌ی WPF جدید را به نام MergeAssembliesIntoWPF ایجاد کنید.
ب) یک پروژه‌ی Class library جدید را به نام MergeAssembliesIntoWPF.ViewModels به این Solution اضافه کنید. از آن برای تعریف ViewModelهای برنامه استفاده خواهیم کرد.
برای نمونه کلاس ذیل را به آن اضافه کنید:

```
namespace MergeAssembliesIntoWPF.ViewModels
{
    public class ViewModel1
    {
        public string Data { set; get; }

        public ViewModel1()
        {
            Data = "Test";
        }
    }
}
```

ج) یک پروژه‌ی WPF User control library را نیز به نام MergeAssembliesIntoWPF.Shell به این Solution اضافه کنید. از آن برای تعریف Viewهای برنامه کمک خواهیم گرفت.
به این پروژه ارجاعی را به اسمبلی قسمت (ب) اضافه نموده و برای نمونه User control ذیل را به نام View1.xaml به آن اضافه نمائید:

```
<UserControl x:Class="MergeAssembliesIntoWPF.Shell.View1"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             xmlns:VM="clr-
namespace:MergeAssembliesIntoWPF.ViewModels;assembly=MergeAssembliesIntoWPF.ViewModels"
             d:DesignHeight="300" d:DesignWidth="300">
    <UserControl.Resources>
        <VM:ViewModel1 x:Key="ViewModel1" />
    </UserControl.Resources>
    <Grid DataContext="{Binding Source={StaticResource ViewModel1}}">
        <TextBlock Text="{Binding Data}" />
    </Grid>
</UserControl>
```

در پروژه اصلی Solution (قسمت الف)، ارجاعاتی را به دو اسمبلی قسمت‌های ب و ج اضافه کنید. سپس MainWindow.xaml آن‌را به نحو ذیل تغییر داده و برنامه را اجرا کنید:

```
<Window x:Class="MergeAssembliesIntoWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:V="clr-namespace:MergeAssembliesIntoWPF.Shell;assembly=MergeAssembliesIntoWPF.Shell"
        Title="MainWindow" Height="350" Width="525">
    <Window.Resources>
```

```
        <V:View1 x:Key="View1" />
    </Window.Resources>
    <Grid>
        <V:View1 />
    </Grid>
</Window>
```

تا اینجا باید متن Test در پنجره اصلی برنامه ظاهر شود.

ب) مدفون کردن خودکار اسمبلی‌های جانبی برنامه در فایل EXE آن

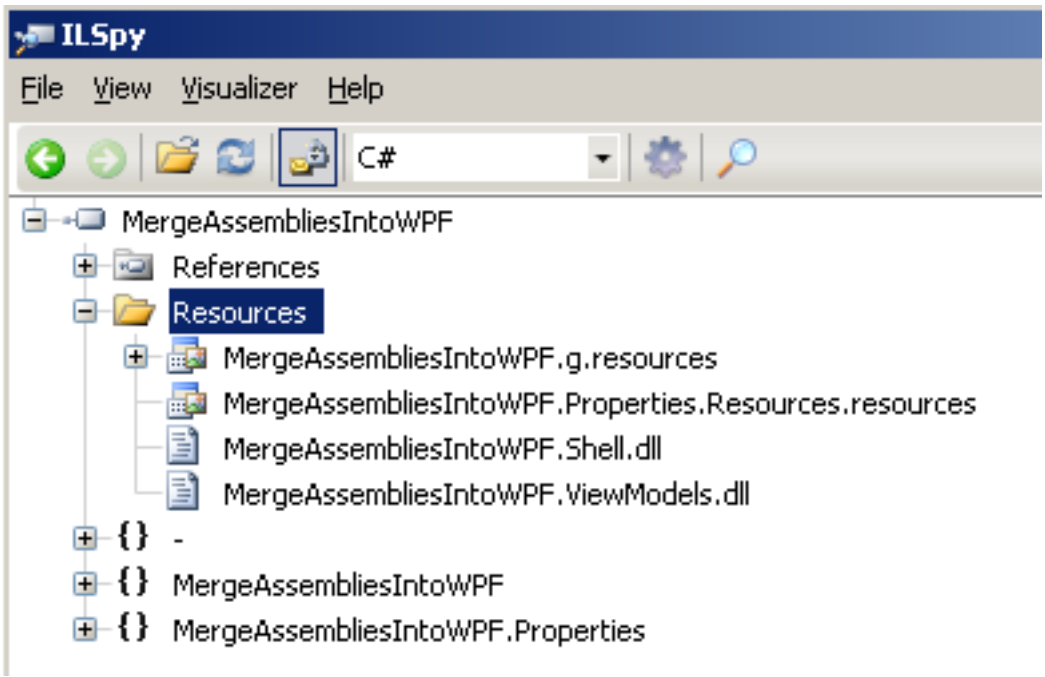
فایل csproj پروژه اصلی را خارج از VS.NET باز کنید. در انتهای آن سطر ذیل قابل مشاهده است:

```
<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
```

پس از این سطر، چند سطر ذیل را اضافه کنید:

```
<Target Name="AfterResolveReferences">
    <ItemGroup>
      <EmbeddedResource Include="@(ReferenceCopyLocalPaths)"
Condition="'%(ReferenceCopyLocalPaths.Extension)' == '.dll'">

<LogicalName>%(ReferenceCopyLocalPaths.DestinationSubDirectory)%(ReferenceCopyLocalPaths.Filename)%(Ref
erenceCopyLocalPaths.Extension)</LogicalName>
      </EmbeddedResource>
    </ItemGroup>
  </Target>
```

این task جدید MSBuild سبب خواهد شد تا با هر بار Build برنامه، اسمبلی‌هایی که در ارجاعات برنامه دارای خاصیت Copy local مساوی true هستند، به صورت خودکار به صورت یک resource جدید در فایل exe برنامه مدفون شوند . عموما ارجاعاتی که دستی اضافه می‌شوند، مانند دو اسمبلی یاد شده در ابتدای بحث، دارای خاصیت Copy local=true نیز هستند.
پس از این تغییر نیاز است یکبار پروژه را بسته و مجددا باز کنید. اکنون پروژه را build کنید و جهت اطمینان بیشتر آن‌را برای مثال توسط ILSpy مورد بررسی قرار دهید:



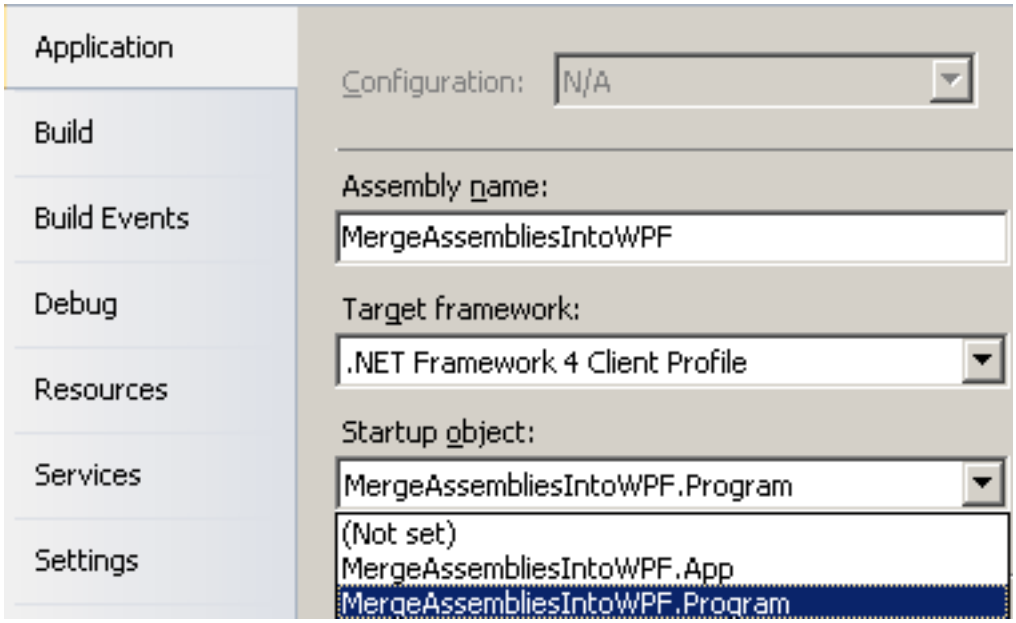
همانطور که مشاهده می‌کنید، دو اسمبلی مورد استفاده در برنامه به صورت خودکار در قسمت منابع فایل EXE مدفون شده‌اند. اگر به مسیر LogicalName تنظیمات فوق دقت کنید، DestinationSubDirectory نیز ذکر شده‌است. علت این است که بسیاری از اسمبلی‌های بومی سازی شده WPF با نام‌هایی یکسان اما در پوشه‌هایی مانند fa، fr و امثال آن ذخیره می‌شوند. به همین جهت نیاز است بین این‌ها تمایز قائل شد.

ج) بارگذاری خودکار اسمبلی‌ها در AppDomain برنامه

تا اینجا اسمبلی‌های جانبی را در فایل EXE مدفون کرده‌ایم. اکنون نوبت به بارگذاری آن‌ها در AppDomain برنامه است. برای اینکار نیاز است تا روال رخدادگردان AppDomain.CurrentDomain.AssemblyResolve را تحت نظر قرار داده و اسمبلی‌هایی را که برنامه درخواست می‌کند، در همینجا از منابع خوانده و به AppDomain اضافه کرد.
انجام اینکار در برنامه‌های WinForms ساده‌است. فقط کافی است به متد Program.Main برنامه مراجعه کرده و تعریف یاد شده را به ابتدای متد Main اضافه کرد. اما در WPF هرچند فایل App.xaml.cs به نظر نقطه‌ی آغازین برنامه است، اما در واقع اینطور نیست. برای نمونه، پوشه‌ی obj\Debug برنامه را گشوده و فایل App.g.i.cs آن‌را بررسی کنید. در اینجا می‌توانید همان رویه شبیه به برنامه‌های WinForm را در متد Program.Main آن، مشاهده کنید. بنابراین نیاز است کنترل این مساله را راسا در دست بگیریم:

```
using System;
using System.Globalization;
using System.Reflection;

namespace MergeAssembliesIntoWPF
{
    public class Program
    {
        [STAThreadAttribute]
        public static void Main()
        {
            AppDomain.CurrentDomain.AssemblyResolve += OnResolveAssembly;
            App.Main();
        }

        private static Assembly OnResolveAssembly(object sender, ResolveEventArgs args)
        {
            var executingAssembly = Assembly.GetExecutingAssembly();
            var assemblyName = new AssemblyName(args.Name);

            var path = assemblyName.Name + ".dll";
            if (assemblyName.CultureInfo.Equals(CultureInfo.InvariantCulture) == false)
            {
                path = String.Format(@"{0}\{1}", assemblyName.CultureInfo, path);
            }

            using (var stream = executingAssembly.GetManifestResourceStream(path))
            {
                if (stream == null)
                    return null;

                var assemblyRawBytes = new byte[stream.Length];
                stream.Read(assemblyRawBytes, 0, assemblyRawBytes.Length);
                return Assembly.Load(assemblyRawBytes);
            }
        }
    }
}
```

کلاس Program را با تعاریف فوق به پروژه خود اضافه نمائید. در اینجا Program.Main مورد نیاز خود را تدارک دیده‌ایم. کار آن مدیریت روال رخدادگردان AppDomain.CurrentDomain.AssemblyResolve برنامه پیش از شروع به هر کاری است. در روال رخداد گردان OnResolveAssembly، برنامه اعلام می‌کند که به چه اسمبلی خاصی نیاز دارد. ما آن‌را از قسمت منابع خوانده و سپس توسط متد Assembly.Load آن‌را در AppDomain برنامه بارگذاری می‌کنیم.
پس از اینکه کلاس فوق را اضافه کردید، نیاز است کلاس Program اضافه شده را به عنوان Startup object برنامه نیز معرفی کنید:



انجام اینکار ضروری است؛ در غیراینصورت با متد Main موجود در فایل App.g.i.cs تداخل می‌کند.
اکنون برای آزمایش برنامه، یکبار آن‌را Build کرده و بجز فایل Exe، مابقی فایل‌های موجود در پوشه‌ی bin را حذف کنید. سپس برنامه را خارج از VS.NET اجرا کنید. کار می‌کند!

MergeAssembliesIntoWPF.zip

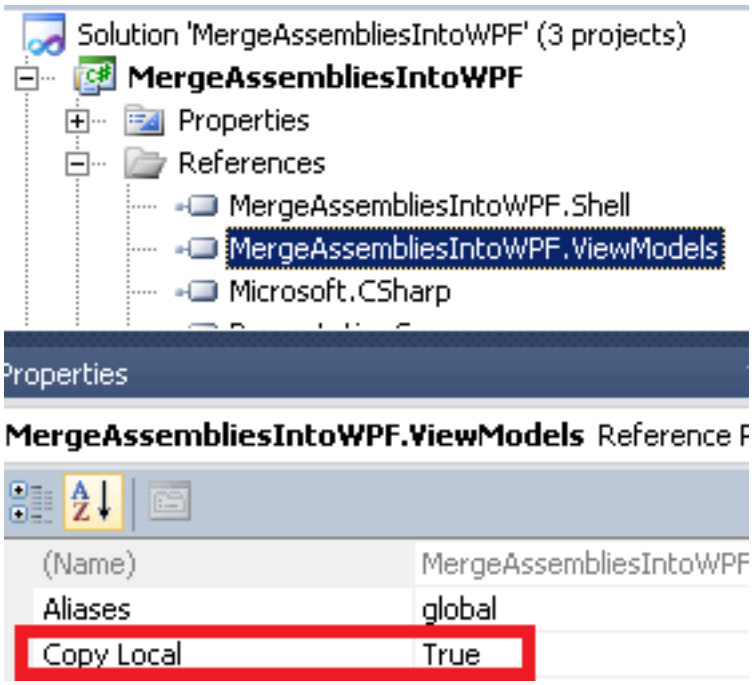
نظرات خوانندگان

نویسنده: ژوپیتر
تاریخ: ۱۳۹۲/۱۱/۰۱ ۷:۵۷

با سلام؛ در یک پروژه ویندوزی روش **ب** را انجام دادم اما پس از Build ارجاع‌ها به بخش Resources اضافه نشد، علت چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۱ ۹:۳۵

Copy local=true در پروژه اصلی (تولید کننده فایل EXE) به این صورت تنظیم می‌شود:



نویسنده: ژوپیتر
تاریخ: ۱۳۹۲/۱۱/۰۱ ۱۰:۵۴

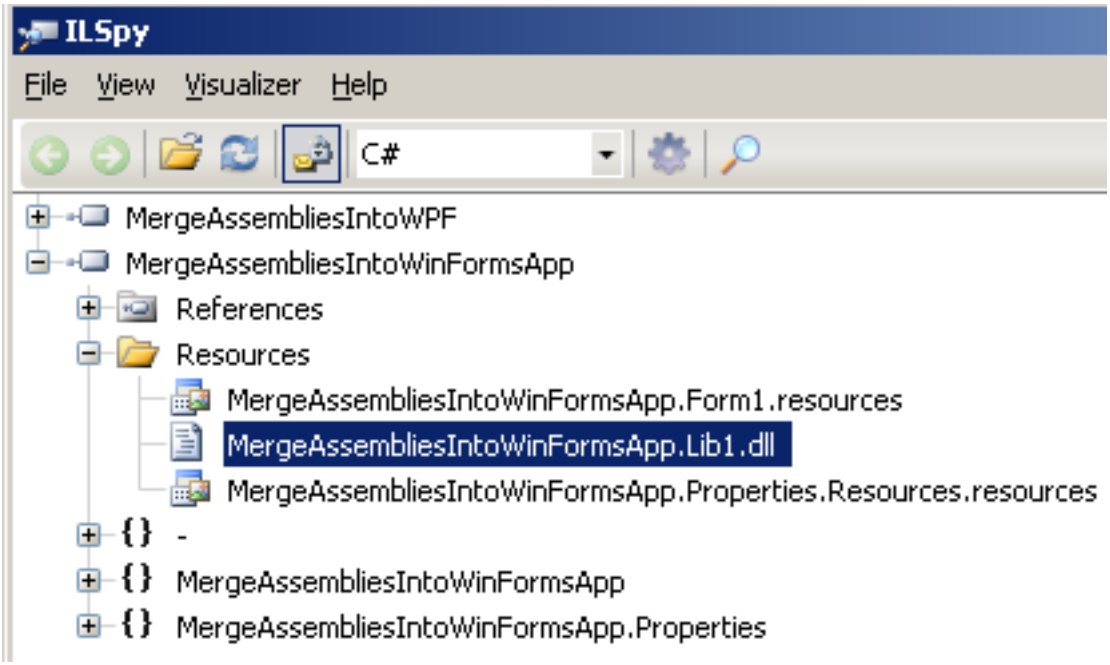
منظورم یک پروژه WinForms بود، نه WPF. تمامی اسمبلی‌های لازم به صورت CopyLocal هستند. بنده فقط بند ب را انجام دادم، یعنی فقط فایل CsProj را ویرایش کردم. (اگر باید الف-ج کامل انجام شوند، برای WinForms قسمت الف چگونه خواهد بود؟)
تا پیش از این از Smart Assembly برای merge استفاده می‌کردم. ولی متاسفانه این نرم‌افزار توانایی ادغام همه‌ی اسمبلی‌ها به خصوص اسمبلی‌های Third party company را ندارد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۱ ۱۱:۳۵

فرقی نمی‌کند. یک مثال:

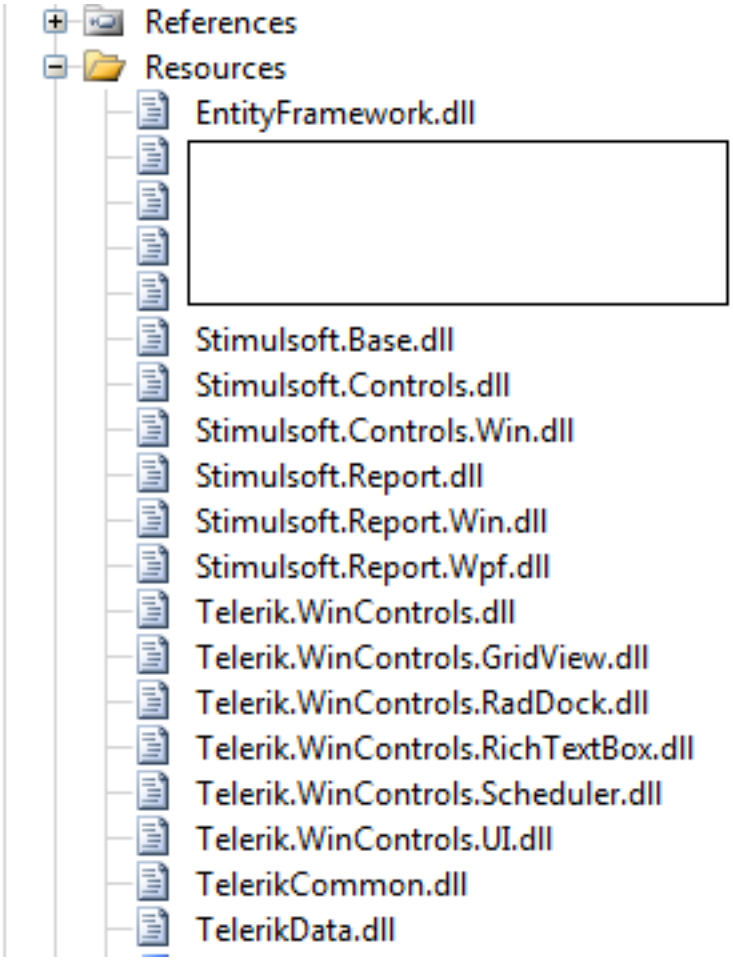
MergeAssembliesIntoWinFormsApp.zip

- فایل MergeAssembliesIntoWinFormsApp.csproj آن (فایل csproj پروژه اصلی) ویرایش شده برای افزودن AfterResolveReferences قسمت ب
- فایل Program.cs استاندارد آن ویرایش شده برای افزودن تعاریف AppDomain.CurrentDomain.AssemblyResolve قسمت ج



نویسنده: ژوپیتر
تاریخ: ۱۳۹۲/۱۱/۰۱ ۱۵:۱۲

نمونه ارسالی شما به خوبی کار می‌کند.
اما برای یک پروژه عملیاتی که از کامپوننت‌های ثالث استفاده می‌کند این روش پاسخ گو نبود و با پیام Windows unhandled error متوقف می‌شود. (تمامی اسمبلی هایی که تا پیش از این کنار فایل EXE مستقر بودند و برنامه کنار آنها صحیح کار می‌کرد، با روش گفته شده در فایل EXE برنامه مدفون شدند.)
همچنین ILSpy صحت وجود را تایید کرد:



بررسی لاگ Application ویندوز خبر از پیدا نشدن فایلی (System.IO.Exception) در متد Main می‌دهد و نکته دیگری در آن ذکر نشده.

آیا راه حلی وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۱ ۱۶:۳۶

- بعضی از اسمبلی‌های دات نتی Mixed mode هستند؛ مانند System.Data.SQLite.DLL. کد هسته اصلی آن، SQLite نوشته شده با زبان سی است. برای استفاده از آن در دات نت با استفاده از C++ CLI ، یک روکش دات نتی تهیه کرده‌اند تا در دات نت به راحتی قابل استفاده شود (روش مرسوم و سریعی است برای استفاده از کتابخانه‌های C و C++ در دات نت). این نوع DLLها با استفاده از روش Assembly.Load ذکر شده در متن قابل بارگذاری نیستند. باید در یک پوشه temp نوشته شده و سپس توسط Assembly.LoadFile بارگذاری شوند. یک مثال کامل در این مورد (قسمت Loading Unmanaged DLL آن مد نظر است): Load DLL From Embedded Resource

- یک try/catch در قسمت بارگذاری اسمبلی قرار دهید تا بهتر منبع مشکل را شناسایی کنید. یک مثال

- شخص دیگری در اینجا گزارش داده اگر Generate serialization assembly در قسمت تنظیمات پروژه، ذیل Build > Output فعال است، باید خاموش شود تا پروژه کرش نکند.

- اگر نوع اسمبلی، PCL است (Portable Class Library)، باز هم روش Assembly.Load به نحوی که در مطلب ذکر شده کار نمی‌کند و باید به صورت ذیل اصلاح شود:

```
private static Assembly loadEmbeddedAssembly(string name)
{
    if (name.EndsWith("Retargetable=Yes")) {
```

```
        return Assembly.Load(new AssemblyName(name));
    }
    // Rest of your code
    //...
}
```

- همچنین در کامنت‌های این مطلب شخصی عنوان کرده کرش را با افزودن ویژگی ذیل به متد Main، حل کرده:

```
[MethodImpl(MethodImplOptions.NoOptimization)]
```

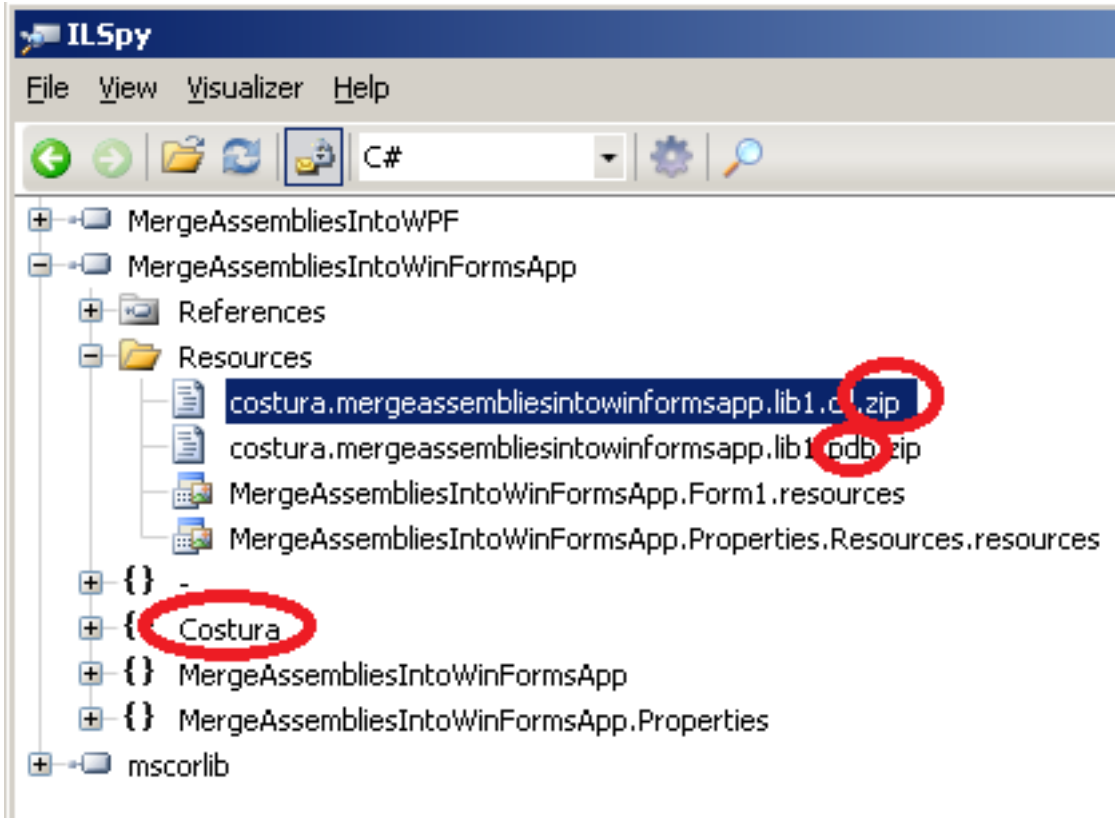
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۲ ۲۲:۱۱

یک نکته‌ی تکمیلی

اگر به دنبال یک راه حل پخته‌تر هستید که با انواع و اقسام اسمبلی‌ها بتواند کار کند (از mixed mode گرفته تا pcl و غیره)، افزونه‌ی Fody / Costura توصیه می‌شود. کار با آن نیز بسیار ساده‌است. فقط کافی است دستور زیر را در کنسول پاور شل نیوگت VS.NET اجرا کنید:

```
PM> Install-Package Costura.Fody
```

بعد از نصب، تنها یکبار برنامه را مجددا build کنید.
اکنون اگر اسمبلی آن‌را بررسی کنید موارد ذیل را مشاهده خواهید کرد:



الف) اسمبلی‌های مدفون شده را zip کرده‌است.
ب) فایل pdb هم لحاظ شده.
ج) راه انداز خودکار و کدهای AssemblyResolver را تحت فضای نام Costura به فایل EXE نهایی افزوده‌است.

Fody یکی از ابزارهای AOP سورس باز دات نت است.

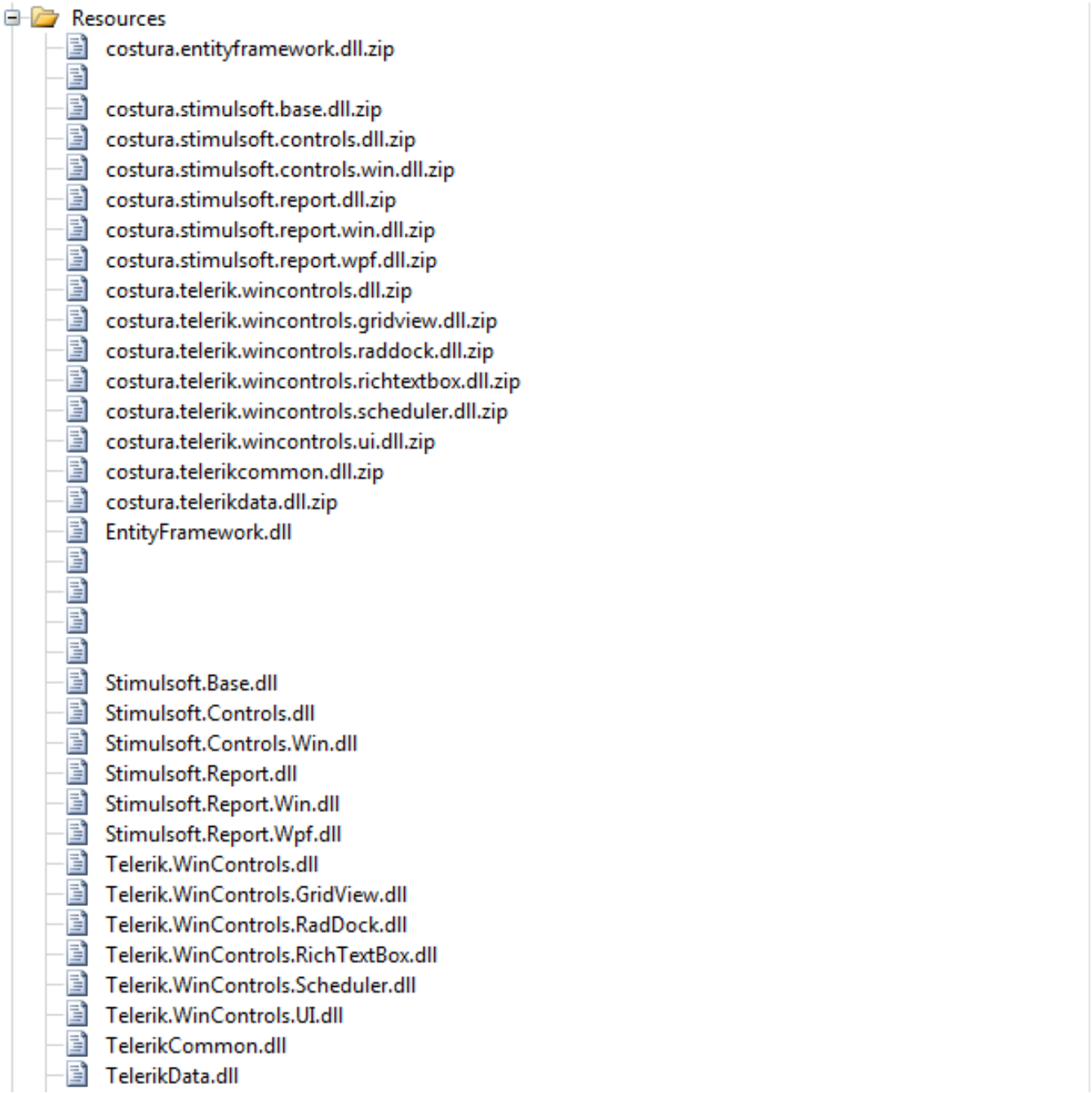
نویسنده: ژوپیتر
تاریخ: ۱۳۹۲/۱۱/۰۳ ۹:۱۳

راه حل بسیار جامع و ساده ای ارائه کردید که مشکلات روش‌های قبل را ندارد، برنامه به خوبی اجرا می‌شود ولی هنگام گرفتن گزارش با استفاده از stimulsoft خطای زیر ظاهر می‌شود:

(ساختار try-catch نادیده گرفته می‌شود و یک Unhandled Exception رخ می‌دهد.)

```
The type or namespace name 'Stimulsoft' could not be found (are you missing a using directive or an assembly reference?)
```

با قرار دادن اسمبلی‌های StimulReport در کنار فایل EXE مشکل برطرف می‌شود در صورتی که این اسمبلی‌ها درون ExE مدفون هستند:



چرا برای اسمبلی‌های تلریک چنین مشکلی به وجود نمی‌آید و اینکه علاوه بر اسمبلی‌های زیپ شده خود اسمبلی‌ها نیز در فایل قرار داده شد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۳ ۹:۲۲

- به نظر پس از افزودن روش Fody / Costura یاد شده، هنوز تنظیمات قبلی Target Name=AfterResolveReferences که در مطلب جاری توضیح داده شد، در فایل csproj شما موجود است که باید حذف شود. دیگر نیازی به آن نیست (علت درج فایل‌های اضافی؛ چون فایل‌های Fody / Costura فقط با یک پیشوند Costura در فایل نهایی قرار می‌گیرند). همچنین تعریف قبلی AppDomain.CurrentDomain.AssemblyResolve را هم حذف کنید.
+ به احتمال زیاد اسمبلی‌های Stimulsoft فقط همین چند مورد نیستند.
همچنین این مورد را می‌توانید در bug tracker آن‌ها نیز ارسال کنید.

نویسنده: ژوپیتر
تاریخ: ۱۳۹۲/۱۱/۰۶ ۱۱:۱۴

راه حل مشکل یاد شده در اینجا

```
<Weavers>
  <Costura CreateTemporaryAssemblies='true' />
</Weavers>
```

برای مواردی که اسمبلی جاری یک اسمبلی پویا را تولید کرده و سپس ارجاعی را به خود به صورت پویا به آن اضافه می‌کند.