عنوان: **فشرده سازی فایل ها در .NET 4.5**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۱۰/۰۶ ۱۳:۵۰
آدرس: www.dotnettips.info
برچسب‌ها: Net Framework 4.5, Compression, Zip.

با اضافه شده فضای نام System.IO.Compression در NET 4.5. دیگر بدون نیاز به کتابخانه‌های همچون DotNetZip به راحتی می‌توانید فایل‌های خود را فشرده یا باز کنید.

کلاس ZipFile

این کلاس امکان فشرده یا باز نمون فایل یا یک پوشه را در اختیارمان قرار میدهد. مثلا برای فشرده سازی یک پوشه از کد زیر استفاده می‌نمایید

```
string startPath = @"c:\example\start";
string zipPath = @"c:\example\result.zip";
ZipFile.CreateFromDirectory(startPath, zipPath);
```

برای بار نمون یک فایل فشرده هم از تابع ExtractToDirectory استفاده نمایید

```
string extractPath = @"c:\example\extract";
ZipFile.ExtractToDirectory(zipPath, extractPath);
```

کلاس ZipArchive

کلاس ZipFile و ZipArchive مکمل هم دیگر هستند و اکثرا با هم دیگر کاربرد دارد اما این کلاس برای دستکاری فایل Zip استفاده می‌شود مثلا برای ایجاد یک فایل zip و کنترل بیشتر بر روی آن در مثال زیر یک ابتدا یک فایل خالی ایجاد کرده ایم و با تابع CreateEntityFromFile فایل‌های مورد را با مسیر و نام آن و حتی کیفیت فشردگی به آن اضافه نموده اید.

```
using (ZipArchive zipFile = ZipFile.Open(zipName, ZipArchiveMode.Create))
{
    zipFile.CreateEntryFromFile(@"C:\Temp\File1.txt", "File1.txt");
    zipFile.CreateEntryFromFile(@"C:\Temp\File2.txt", "File2.txt", CompressionLevel.Fastest);
}
```

اما اگر بخواهیم فایل Zip را در یک MemoryStream ایجاد کنیم کافیست از کد زیر استفاده نمایید

```
using (MemoryStream zipStream = new MemoryStream())
{
        using (ZipArchive zipFile = new ZipArchive(zipStream, ZipArchiveMode.Create))
        {
                zipFile.CreateEntryFromFile(filepath, filename);
        }
}
```

حال می‌خواهیم یک فایل Zip را باز کنیم. کلاس ZipArchiveEntry برای دسترسی به فایل‌های موجود در فایل Zip می‌باشد.

```
using (ZipArchive archive = ZipFile.OpenRead(zipName))
{
    foreach (ZipArchiveEntry file in archive.Entries)
    {
            Console.WriteLine("File Name: {0}", file.Name);
            Console.WriteLine("File Size: {0} bytes", file.Length);
            Console.WriteLine("Compression Ratio: {0}", ((double)file.CompressedLength  /
file.Length).ToString("0.0%"));
            file.ExtractToFile(directorypath);
    }
}
```

عنوان: **استفاده از Async&Await برای پیاده سازی متد های Async**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۱/۱۱/۲۳ ۱۴:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: Net Framework 4.5, Asynchronous Programming.

در این مطلب می‌خوام روش استفاده از Async&Await رو براتون بگم. Async&Await خط و مشی جدید Microsoft برای تولید متدهای Async هستش که نوشتن این متدها رو خیلی جذاب کرده و کاربردهای خیلی زیادی هم داره. مثلا هنگام استفاده از Web Api در برنامه‌های تحت ویندوز نظیر WPF این روش خیلی به ما کمک می‌کنه و در کل نوشتن Parallel Programming را خیلی جالب کرده.
برای اینکه بتونم قدرت و راحتی کار با این ابزار رو به خوبی نشون بدم ابتدا یک مثال رو به روشی قدیمی‌تر پیاده سازی می‌کنم. بعد پیاده سازی همین مثال رو به روش جدید بهتون نشون می‌دم.
می‌خوام یک برنامه بنویسم که لیستی از محصولات رو به صورت Async در خروجی چاپ کنه. ابتدا کلاس مدل:

```
public class Product
    {
        public int Id { get; set; }

        public string Name { get; set; }
    }
```

حالا کلاس ProductService رو می‌نویسم:

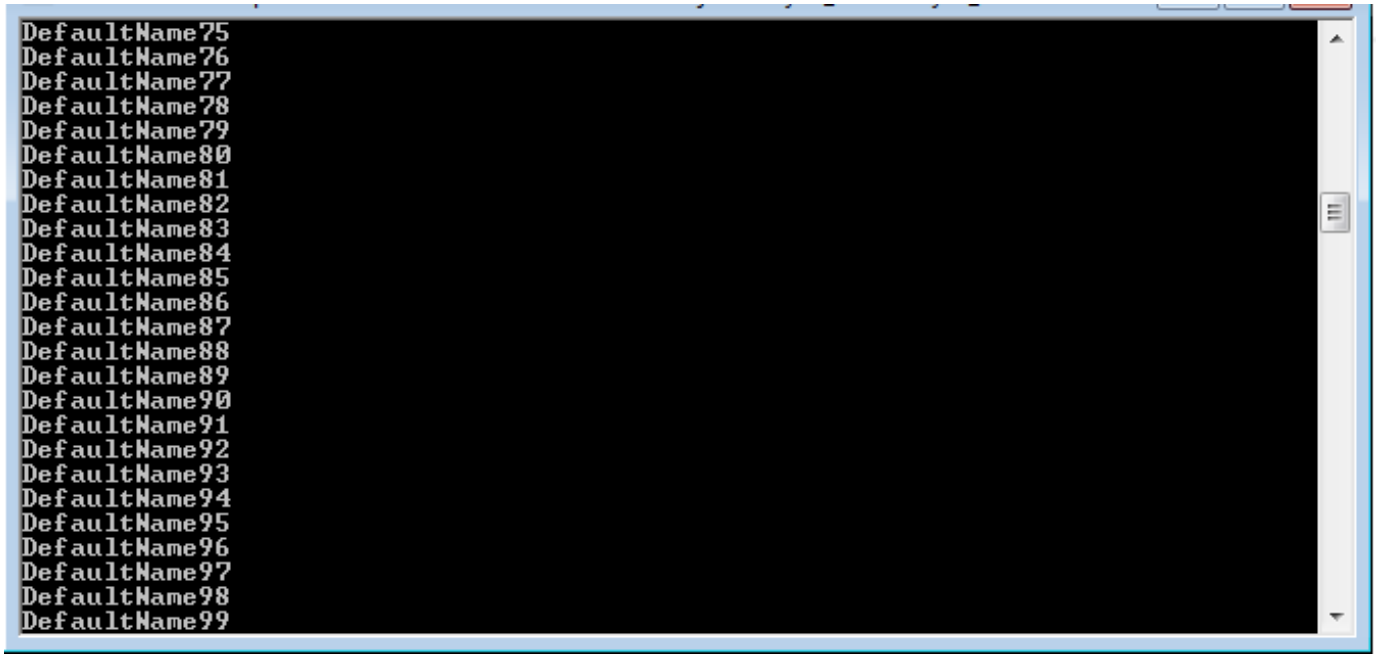
```
public class ProductService
    {
        public ProductService()
        {
            ListOfProducts = new List<Product>();
        }

        public List<Product> ListOfProducts
        {
            get;
            private set;
        }

        private void InitializeList( int toExclusive )
        {
            Parallel.For( 0 , toExclusive , ( int counter ) =>
            {
                ListOfProducts.Add( new Product()
                {
                    Id = counter ,
                    Name = "DefaultName" + counter.ToString()
                } );
            } );
        }

        public IAsyncResult BeginGetAll( AsyncCallback callback , object state )
        {
            var myTask = Task.Run<IEnumerable<Product>>( () =>
            {
                InitializeList( 100 );
                return ListOfProducts;
            } );
            return myTask.ContinueWith( x => callback( x ) );
        }

        public IEnumerable<Product> EndGetAll( IAsyncResult result )
        {
            return ( ( Task<IEnumerable<Product>> )result ).Result;
        }
    }
```

در کلاس بالا دو متد مهم دارم. متد اول آن BeginGetAll است و همونطور که می‌بینید خروجی اون از نوع IAsyncResult است و باید هنگام استفاده، اونو به متد EndGetAll پاس بدم تا خروجی مورد نظر به دست بیاد.
متد InitializeList به تعداد ورودی آیتم به لیست اضافه می‌کند و اونو به CallBack میفرسته. در نهایت برای اینکه بتونم از این

کلاس‌ها استفاده کنم باید به صورت زیر عمل بشه:

```
class Program
    {
        static void Main( string[] args )
        {
            GetAllProducts().ToList().ForEach( ( Product item ) =>
            {
                Console.WriteLine( item.Name );
            } );

            Console.ReadLine();
        }

        public static IEnumerable<Product> GetAllProducts()
        {
            ProductService service = new ProductService();

            var output = Task.Factory.FromAsync<IEnumerable<Product>>( service.BeginGetAll ,
service.EndGetAll , TaskCreationOptions.None );
            return output.Result;
        }

    }
```

خیلی راحت بود؛ درسته. خروجی مورد نظر رو می‌بینید:



حالا همین کلاس بالا رو به روش Async&Await می‌نویسم:

```
public async Task<IEnumerable<Product>> GetAllAsync()
        {
            var result = Task.Run( () =>
            {
                InitializeList( 100 );
                return ListOfProducts;
            } );
            return await result;
        }
```

در متد بالا به جای استفاده از 2 متد از یک متد GetAllAsync استفاده کردم که خروجی آون از نوع async

Task<IEnumerable<Product>> بود و برای استفاده از اون کافیه در کلاس Program کد زیر رو بنویسم

```
class Program
    {
        static void Main( string[] args )
        {
            GetAllProducts().Result.ToList().ForEach( ( Product item ) =>
            {
                Console.WriteLine( item.Name );
            } );

            Console.ReadLine();
        }

        public static async Task<IEnumerable<Product>> GetAllProducts()
        {
            ProductService service = new ProductService();

            return await service.GetAllAsync();
        }

    }
```

فکر کنم همتون موافقید که روش Async&Await هم از نظر نوع کد نویسی و هم از نظر راحتی کار خیلی سرتره. یکی از مزایای مهم این روش اینه که همین مراحل رو می‌تونید در هنگام استفاده از WCF در پروژه تکرار کنید. به خوبی کار می‌کنه.

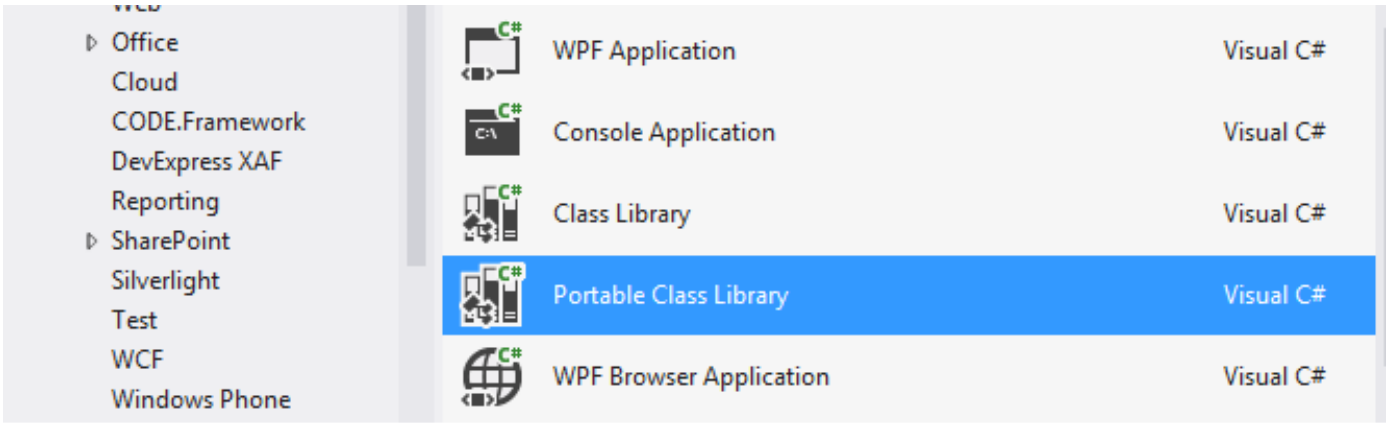
عنوان: **Portable Class Library چیست و چگونه از آن استفاده کنیم؟**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۱/۱۱/۲۵ ۲۲:۲۰
آدرس: www.dotnettips.info
گروه‌ها: Net Framework 4.5, Portability, Portable Class Library.

Portable Class Library چیست؟

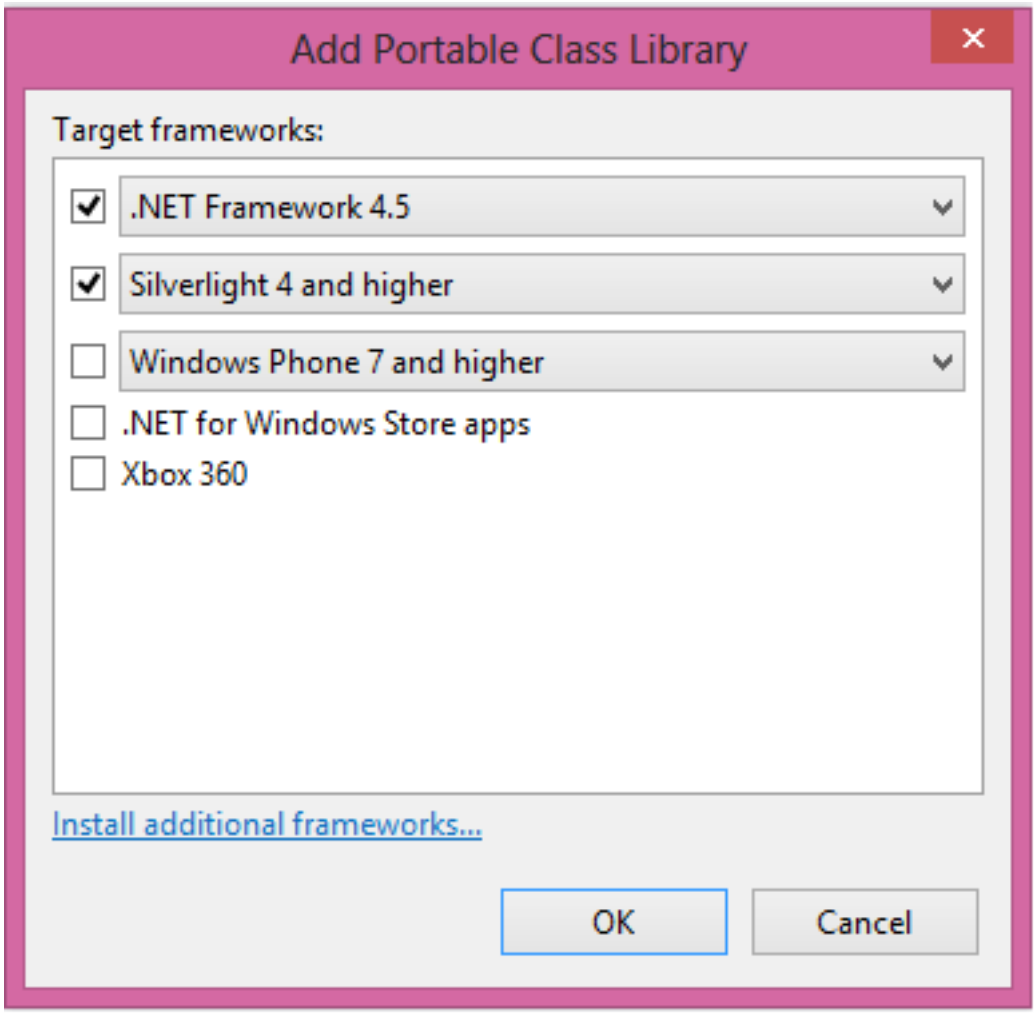
Portable Class Library برای تولید Assembly مدیریت شده که قابلیت استفاده در اکثر تکنولوژی‌ها نظیر WP7(Windows Phone 7 و WPF و Silverlight و Windows Store App و حتی XBox را داراست استفاده می‌شود. این نوع پروژه‌ها میزان استفاده مجدد از کدها رو به حداکثر ممکن می‌رسونه و در عوض تعداد پروژه‌های مورد نیاز رو به حداقل. مورد اصلی استفاده از اون‌ها برای تولید Multi-Targeted Application هاست. برای مثال در تولید پروژه‌های تحت Windows که با تکنولوژی WPF پیاده سازی می‌شوند پروژه ViewModel و Model رو با این تکنولوژی پیاده سازی می‌کنند تا در آینده اگر نیاز بود قسمتی از پروژه با استفاده از Silverlight یا Windows Store App انجام بشه بتونیم ViewModel , Model نوشته شده رو به این پروژه‌ها Reference بدیم. تا قبل از برای انجام این کار باید این دو بخش رو دوباره برای هر تکنولوژی بازنویسی می‌کردیم.

روش استفاده

زمانی که یک پروژه از نوع Portable Class Library رو ایجاد می‌کنیم باید حداقل 2 یا چند تا از Platformهای مورد نظر رو انتخاب کنیم. به صورت زیر :



بعد از انتخاب گزینه Portable Class Library و زدن کلید Enter فرم زیر ظاهر خواهد شد که در این قسمت باید Platformهای مورد نظر رو انتخاب کنیم.



در اینجا من Platform 2 رو انتخاب کردم. یکی Silverlight و Net 4.5. یعنی این Portable Class Library هم می‌تونه به پروژه Silverlight و هم به Net 4.5. Class library رفرنس داده بشه.

حتما می‌پرسید چه طوری؟

در واقع Microsoft برای انجام این کار فقط یک سری از Referenceهای مشترک بین این Platform 2 رو به پروژه اضافه میکنه و همین موضوع باعث شده کار با این پروژه‌ها نیاز به یکم مهارت داشته باشه.

برای مثال اگر قصد استفاده از تکنولوژی Async&Await رو دارید باید یکم به خودتون زحمت بدید و CTP مورد نظر رو نصب کنید.(حالا اگر از Net4 استفاده می‌کنید که باید از یک روش دیگه استفاده کنید که در یک مقاله جداگانه براتون توضیح خواهم داد).

به جدول زیر دقت کنید.

Feature	.NET Framework	Windows Store	Silverlight	Windows Phone	Xbox 360
Core	√	√	√	√	√
LINQ	√	√	√	√	
IQueryable	√	√	√	Only 7.5	
Dynamic keyword	Only 4.5	√	√		
Managed Extensibility Framework (MEF)	√	√	√		
Network Class	√	√	√	√	

Feature	.NET Framework	Windows Store	Silverlight	Windows Phone	Xbox 360
Library (NCL)					
Serialization	√	√	√	√	
Windows Communication Foundation (WCF)	√	√	√	√	
Model-View-View Model (MVVM)	Only 4.5	√	√	√	
Data annotations	Only 4.0.3 and 4.5	√	√		
XLINQ	Only 4.0.3 and 4.5	√	√	√	√

در جدول بالا به صورت کامل مشخص شده که در هر پلاتفرم چه چیزی Support میشه. برای مثال Data Annotation فقط در Net4.3 , 4.5 قابل استفاده است.

اسمبلی‌های زیر در یک Portable Class Library قابل استفاده هستند.

mscorlib.dll

System.dll

System.Core.dll

System.Xml.dll

System.ComponentModel.Composition.dll

System.Net.dll

System.Runtime.Serialization.dll

System.ServiceModel.dll

System.Xml.Serialization.dll

System.Windows.dll - From Silverlight

در صورتی که از VS2010 استفاده می‌کنید می‌تونید از این جا Portable Library Tools رو دانلود کنید. بعد از نصب دقیقا همین مراحل رو برای ایجاد پروژه انجام بدید.

پیاده سازی یک مثال عملی

در این مثال قصد دارم یک کلاس برای مدیریت تاریخ شمسی درست کنم. مراحل زیر رو دنبال کنید

یک Portable Class Library به نام Common به پروژه اضافه کنید.

یک کلاس به نام PersianDate به برنامه اضافه کرده به صورت زیر

```
public class PersianDate<TCalendar> where TCalendar : System.Globalization.Calendar
```

```
    {
        public PersianDate()
        {
            this.CurentCalendar = System.Activator.CreateInstance<TCalendar>();
        }

        public TCalendar CurentCalendar
        {
            get;
            private set;
        }

        public string GetDate()
        {
            return string.Format( "{0}/{1}/{2}", this.CurentCalendar.GetYear( DateTime.Today
).ToString( "####0000" )
                , this.CurentCalendar.GetMonth( DateTime.Today ).ToString( "##00" )
                , this.CurentCalendar.GetDayOfMonth( DateTime.Today ).ToString( "##00" ) );
        }
    }
```

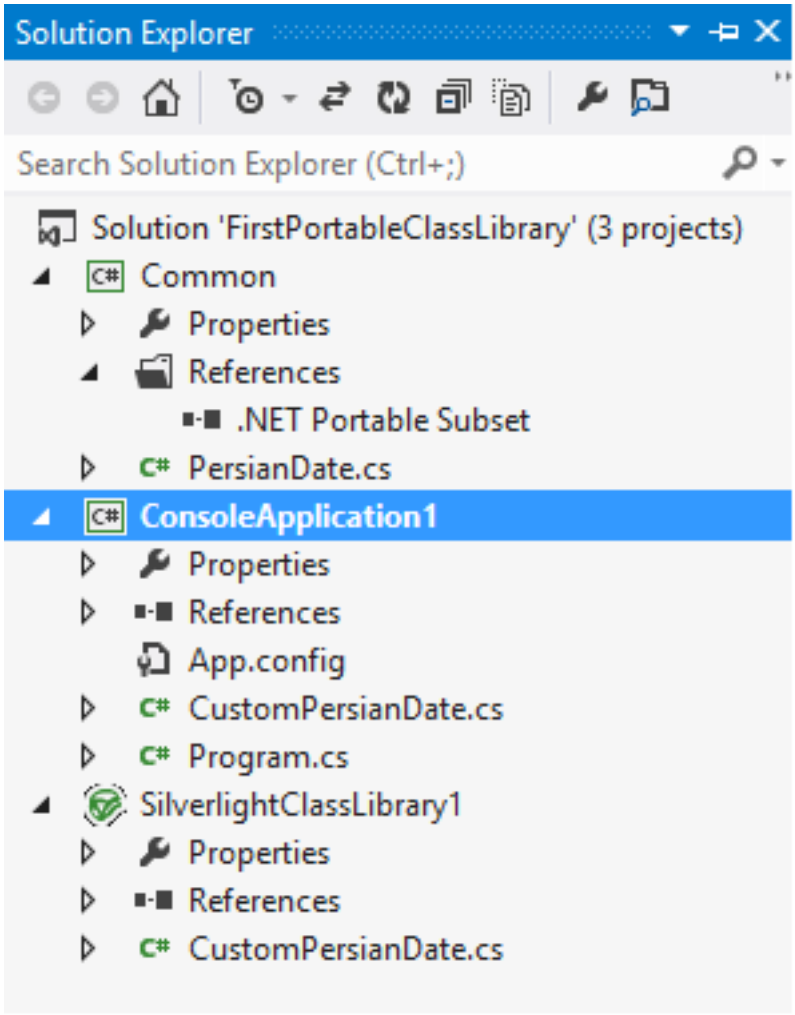
کلاس بالا به صورت Generic تعریف شده که شما باید هنگام استفاده حتما یک نوع Calendar رو بهش بدید. دلیل این کار اینه که کلاس PersianCalendar در Portable Class Library وجود ندارد و فقط یک کلاس پایه Calendar در فضای نام System.Globalization وجود داره.

حالا یک پروژه از نوع Console Application به برنامه اضافه کنید و یک Reference به پروژه Common بدید و بعد کلاس زیر رو بنویسید.

```
public class CustomPersianDate
    {
        public CustomPersianDate()
        {

        }

        public static Common.PersianDate<System.Globalization.PersianCalendar> PersianCalendar
        {
            get
            {
                return _persianCalendar ?? ( _persianCalendar = new
Common.PersianDate<System.Globalization.PersianCalendar>() );
            }
        }
        private static Common.PersianDate<System.Globalization.PersianCalendar> _persianCalendar;
    }
```

در این قسمت ما نوع TCalendar رو از نوع PersianCalendar تعیین کردیم.

حالا یک پروژه از نوع Silverlight به برنامه اضافه کنید و یک Reference به پروژه Common بدید و بعد کلاس بالا بدون تغییر بنویسید.
نمای کلی پروژه باید به صورت زیر باشد.



بعد پروژه ConsoleApplication رو به عنوان پروژه StartUp انتخاب کنید و فایل Program رو به صورت زیر تغییر بدید.

```
class Program
    {
        static void Main( string[] args )
        {
            Console.WriteLine( "Today Is ?{0}", CustomPersianDate.PersianCalendar.GetDate() );

            Console.ReadLine();
        }
    }
```

خوب نتیجه به صورت زیر خواهد بود:

برای پروژه Silverlight هم نتیجه قطعا به همین صورت است.

همان طور که دید انتخاب نوع Calendar به خود Applicationها واگذار شد و شما می‌تونید هر نوع تقویم رو به عنوان TCalendar تعیین کنید.

امیدوارم شما هم مثل من به این تکنولوژی دلچسب علاقه پیدا کرده باشید.

نظرات خوانندگان

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۱/۲۶ ۱:۳۵

با سلام
کار جالبی نیست، چون به شدت محدودیت می‌آره
کار صحیح‌تر این هست که یک پروژه مبتنی بر .NET بسازی، و برای استفاده مجدد کدت برای مثال در سیلورلایت، یه پروژه Silverlight ای بسازی، و فایل‌های پروژه .NET. ای رو در اون Add As Link کنی که یه نوع Shortcut به حساب می‌آد، یعنی فقط یه فایل وجود داره
یک وقت هست که Library مورد نظر شما برای استفاده (مثلا WCF Data Services Client) در هر دو هست، آن هم با یک خط کد مشابه، ولی در حالت Portable امکان استفاده از اون رو ندارید، در حالی که در این روش هیچ محدودیتی نیست، مگر این که کلا کد شما برای مثال در سیلورلایت کار نکنه
موفق باشی

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۲۶ ۹:۸

Add As Link فقط زمانی کار می‌کنه که کدهای شما در هر دو پروژه قابل کامپایل باشند. اگر قابلیتی در WPF باشه که در سیلورلایت نیست، این روش جواب نمی‌ده و خیلی از قابلیت‌ها مشترک نیست.

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۱/۱۱/۲۶ ۱۵:۴۴

نوع نگاه شما به Portable Class Library مناسب نیست. شما وضعیتی رو دز نظر بگیرید که در یک پروژه WPF که به روش MVVM پیاده سازی شده و از Cinch Framwork یا Onyx برای پیاده سازی قسمت ViewModel استفاه کردید. در این حالت اصلا کدهای شما در Silverlight کامپایل نخواهد شد و روش Add As Link در این جا کاربرد نداره.

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۱/۲۶ ۱۸:۴۷

دوستان، اصلا مطلب من رو مطالعه کردید ؟
بله، مسلما وقتی کدی در سیلورلایت کار نکنه، یعنی کار نمی‌کنه، حالا به هر روشی، مگه این که شما بفرمایید Portable Library قابلیت جدیدی رو برای مثال به سیلورلایت اضافه می‌کنه که "در حالت عادی" در دسترس نیست.
وقتی پروژه شما Silverlight و .NET. اش جدا باشد، Silverlight ای اون هم از بقیه Silverlight ای‌های موجود در اینترنت می‌تونه استفاده کنه (برای مثال WCF Data Services Client Library)، و هم می‌تونه از Portableهای که Silverlight شون تیک خورده باشه استفاده کنه، مثل Post Sharp
اما وقتی شما به جای Add As Link از Portable استفاده می‌کنید، با این که در تعامل با WCF Data Services یک دست خط کد کاملا یکسان دارید، نمی‌تونید تو پروژه تون از WCF Data Services استفاده کنید.
یک وقت هست، شما از MVVM Light Toolkit دارید استفاده می‌کنید، کد WPF و Silverlight تون هم کاملا مشابه هستش، در این جا کار شما با Add As Link راه می‌افته، ولی با Portable نه

در حالت Portable شما System.Linq رو دارید، خوبه، در Add As Link هم اون رو دارید، ولی Expression.Interactivity رو فقط در Add As Link دارید، با این که کد می‌تونه 100% یک دست باشه
موفق و پایدار باشید

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۲۶ ۱۸:۵۲

کتابخانه پرتابل یک روش مدیریت پروژه بهتر است. کدهایی رو که مشترک است بجای اینکه مدام Add as link کنید و یا copy/paste می‌تونید در این کتابخانه‌ها قرار بدید و نهایتا از اسمبلی مشترک استفاده کنید. مابقی مواردی که در این کتابخانه پرتابل قابل استفاده نیست، خوب به روش معمول استفاده خواهند شد.

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۱/۱۱/۲۶ ۲۰:۱۹

زمانی که شما از Add as Link استفاده می‌کنید در واقع معایب زیر رو هم قبول کردید.
اولا مبحث Refactoring برای Add as Link کار نمی‌کنه. یعنی اگه یک فایل یا یک متد رو Rename کنید در بقیه فایل‌ها این Rename انجام نمیشه. یا باید دوباره عملیات Add Link رو انجام بدید یا باید از گزینه RunCustomTools استفاده کنید.
زمانی که یک فایل لینک شده باز کنید و اگر اون فایل در یک پروژه دیگه باز باشه؛ Visual Studio به شما چنین پیغامی میده "This document is opened by another project".

زمانی که در حال استفاده از فایل‌های لینک شده هستید intellisense به شما API هایی رو نمایش می‌ده که ممکنه توی اون Platform قایل استفاده نباشند.

زمانی که قصد داشته باشید که یک Library بسازید که بعدا از اون دوباره استفاده کنید یک Portable Library خیلی بهتر به ما کمک می‌کنه تا فایل هایی که Add as Link شده باشند.
Add as Link برای مدیریت و نگهداری در پروژه‌ها با حجم و تعداد Porgrammer بالا مناسب نیست.
و...
با تشکر از توضیحات دوست عزیز-سعید

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۱/۲۷ ۰:۳۵

علاقه ای به طولانی کردن بحث ندارم، لکن
در مورد این قسمت " مابقی مواردی که در این کتابخانه پرتابل قابل استفاده نیست خوب به روش معمول استفاده خواهند شد "، بیایم و با مثال صحبت کنیم
ما فرضا می‌آییم View Model رو با Portable Class Library می‌زنیم، سپس، می‌آییم و یک پروژه Silverlight ای و WPF ای و اندرویدی قرار می‌دهیم و از View Model مربوطه استفاده می‌کنیم (از این که View Model مربوطه رو می‌خواهیم در اندروید هم استفاده کنیم و در این صورت از کدام حالت Portable Class Library بگذریم)، حال اگر بخواهیم از فریم ورک X که در هر 3 هست و خط کد یکسانی داره استفاده کنیم، باید اون رو به کدوم پروژه اضافه کنیم تا به این جمله برسیم ؟ " مابقی مواردی که در این کتابخانه پرتابل قابل استفاده نیست خوب به روش معمول استفاده خواهند شد " ، ممنون می‌شوم مثالی ارائه دهید، چون من واقعا به این آیتم احتیاج دارم تا بتونم استفاده از Portableها رو به صورت جدی شروع کنم.
زمانی که در پروژه اصلی (که بقیه پروژه‌ها از روی آن Add As Link شده اند)، متدی رو Rename می‌کنیم در خود اون پروژه که عمل Refactor انجام می‌شه، به سادگی، بقیه پروژه که از خودشون چیزی ندارند، و به پروژه اصلی فقط Shortcut دارند، عملا اونها هم Refactor شده هستند.
اشکال This document is opened by رو قبول دارم، ولی اگه شما مبحث هزینه فایده رو قبول داشته باشید، به نظر من این هزینه کوچک به فایده اش واقعا می‌ارزه، مگه این که شما بفرمایید سناریوی آندروید، WPF و Silverlight و کتابخانه‌های مشترک شون رو چه جوری با Portable حل می‌کنید، اون وقت روش Portable فایده اش بیشتره، من که از Portable بدم نمی‌آد، از خدام هستش که همه چی در یه پروژه باشه تا 3 تا پروژه
در مورد Intelliscne بله، این امکان داره، ولی من ترجیح می‌دم به جای این که کور کورانه محدود بشم، و نتونم از کدی که مطمئنم در WPF و Silverlight و اندروید به یک شکل هست نتونم استفاده کنم، چون فقط در Portable Library دیده نشده، ترجیح می‌دم 1% به اندازه 5 خط کد اشتباه بنویسم، که کامپایلر 2 دقیقه بعد خطاش رو ساده و واضح بهم می‌گه
مورد آخر جمله‌ی کلی است و قابلیت نقد و بررسی را ندارد.
موفق باشید

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۲۷ ۱:۷

- چند نفر این دور و اطراف با VS.NET کار اندروید انجام می‌دن؟ به چه جهت مایکروسافت باید برای اندروید کتابخانه پرتابل ارائه بده؟ چه نفعی براش داره؟

- " مابقی مواردی که در این کتابخانه پرتابل قابل استفاده نیست خوب به روش معمول استفاده خواهند شد " مربوط به مثال WCF Data Services بود که زدی؛ این جزو مابقی موارد هست. کتابخانه پایه و کدهای مشترک رو با پرتابل درست می‌کنی و این مورد در کارهای متداول قابل انجام با VS.NET که پیش بینی شده قابل استفاده است، مابقی موارد مثل WCF Data Services خارج از این کتابخانه پرتابل قرار می‌گیره و وابسته به فناوری پایه خاص.