Silverlight 4

پژوهش و نگارش: وحید نصیری
Vahid_Nasiri@Yahoo.Com
بهار ۱۳۸۹

فهرست مطالب

فصل ۲۷- استفاده از MEF در Silverlight 4

مقدمه

MEF یا **M**anaged **E**xtensibility **F**ramework جزو تازه‌های NET Framework 4. و همچنین Silverlight 4 است و هدف از آن فراهم آوردن روشی استاندارد برای تولید برنامه‌های بسط پذیر می‌باشد. برای مثال از این چارچوب کاری برای تهیه‌ی اجزای مختلف Visual Studio.NET 2010 استفاده شده است. مزایای استفاده از MEF به شرح ذیل هستند:

- بدون نیاز به تغییری در کدهای جاری سیستم می‌توان قابلیتی را به آن افزود یا حذف کرد. به این صورت دیگر نگرانی در مورد افزوده شدن مشکلات و یا عیوبات جدید به کدهای موجود، با افزودن قابلیتی جدید به سیستم وجود نخواهد داشت.
- MEF امکان تهیه‌ی برنامه‌هایی را که پس از ارائه‌ی آن‌ها، توسط افزونه‌های اشخاص ثالث قابل توسعه است، فراهم می‌آورد.
- MEF نیز یکی دیگر از کتابخانه‌های سورس بازی است (http://mef.codeplex.com) که پس از بلوغ آن، با مجموعه‌ی NET Framework. یکی شده است و بدون نیاز به کتابخانه‌های اضافی دیگر، هم اکنون در دسترس برنامه نویس‌های Silverlight نیز می‌باشد.
- MEF امکان ماژولار ساختن ارائه‌ی نهایی یک برنامه‌ی Silverlight را فراهم می‌آورد. برای مثال بجای اینکه یک فایل حجیم XAP را به کاربر نهایی ارائه دهیم می‌توانیم قسمت‌های مختلف پروژه را صرفا بر اساس نیاز جاری کاربر به صورت پویا بارگذاری کنیم. به این صورت هم سرعت بارگذاری اولیه برنامه افزایش خواهد یافت و هم صرفه جویی قابل توجهی در منابع مورد استفاده خواهیم داشت.
- درک مفاهیم آن از راه‌حل‌های موجود در کتابخانه‌های Inversion of Control مانند Castle Windsor، Structure Map ، Unity و غیره بسیار ساده‌تر است.

مفاهیم و اصول اولیه MEF

MEF از سه مفهوم اساسی ذیل تشکیل می‌گردد (شکل ۱):

۱. Export : به این ترتیب به MEF خواهیم گفت که سرویسی را قرار است ارائه دهیم:

```
[Export(typeof(IPlugin))]
public class MyPlugin : IPlugin
{ }
```

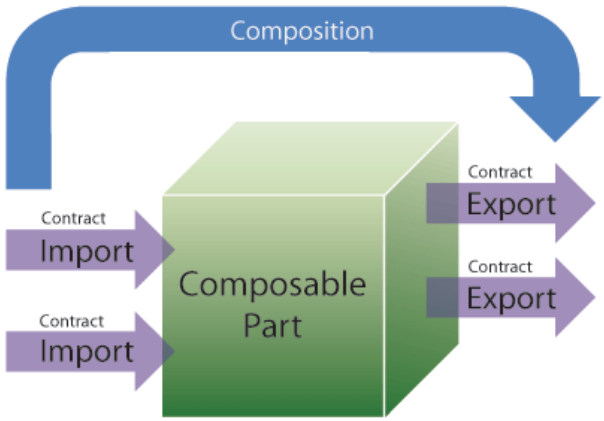
۲. Import : به کمک آن به MEF اعلام خواهیم کرد که به قابلیتی نیاز داریم:

```
[Import]
public IPlugin Plugin { get; set; }

[ImportMany]
public IEnumerable<IPlugin> Plugins { get; set; }
```

۳. Compose : در اینجا به MEF اعلام می‌نمائیم که مجوز انجام عملیات مورد نظر را دارد. سپس کار بررسی Imports/Exports تعریف شده در برنامه آغاز شده و موارد همخوان به یکدیگر پیوند داده خواهند شد. بدون اینکه این اجزا در حین تهیه شدن از وجود یکدیگر مطلع بوده باشند یا ارجاعی از دیگری را در خود داشته باشند:

```
CompositionInitializer.SatisfyImports(this);
```



شکل ۱– نمایی از سه مؤلفه‌ی تشکیل دهنده‌ی MEF .

مثال اول – بررسی ابتدایی نحوه‌ی عملکرد MEF

برای توضیحات بیشتر بهتر است یک مثال ساده ارائه شود. قصد داریم برنامه‌ای را ارائه دهیم که در صفحه‌ی اول آن به صورت پویا، تعدادی User control که افزونه‌های ما را تشکیل می‌دهند، به یک StackPanel اضافه خواهند شد.

یک برنامه‌ی جدید Silverlight را آغاز کرده و سپس ارجاعاتی را به دو اسمبلی استاندارد ذیل به آن اضافه نمائید. این دو کتابخانه حاوی کدهای MEF می‌باشند:

- System.ComponentModel.Composition.dll
- System.ComponentModel.Composition.Initialization.dll

کدهای XAML اولین افزونه‌ی برنامه در ادامه ذکر شده است و تنها حاوی یک دکمه‌ی ساده است:

Plugin1.xaml

```
<UserControl x:Class="SilverlightApplication104.Plugins.Plugin1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
      "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button Name="btn1"/>
    </Grid>
</UserControl>
```

از آنجائیکه این User control قرار است سرویسی را ارائه دهد، به ویژگی Export مزین شده است. همچنین خاصیت متنی محتوای دکمه‌ی این کنترل نیز قرار است به صورت پویا دریافت شود. به همین جهت این خاصیت با ویژگی Import مشخص گردیده است. اما از آنجائیکه نمی‌خواهیم در حین مونتاژ نهایی هر نوع رشته‌ی عمومی تعریف شده‌ای در اینجا انتساب داده شود، قرار دادی (contract) را به نام HelloMEF.Message مشخص کرده‌ایم. روش دیگر دریافت اطلاعات استفاده از ویژگی ImportingConstructor می‌باشد که قابل تعریف بر روی سازنده‌ی یک کلاس است (به همین جهت به MEF یک IOC Container نیز گفته می‌شود).

Plugin1.xaml.cs

```
using System.ComponentModel.Composition;
using System.Windows.Controls;

namespace SilverlightApplication104.Plugins
{
    [Export(typeof(UserControl))]
    public partial class Plugin1 : UserControl
    {
        public Plugin1()
        {
            InitializeComponent();
        }

        [Import("HelloMEF.Message")]
        public string Message
```

```
            {
            get { return (string)btn1.Content; }
            set { btn1.Content = value; }
        }
    }
}
```

اکنون یک کلاس دیگر را به برنامه جهت تعریف قرار داد HelloMEF.Message اضافه می‌کنیم. به این صورت دقیقا مشخص خواهد شد که مقدار خاصیت محتوای دکمه باید از کجا دریافت شود.

MessagePart.cs

```
using System.ComponentModel.Composition;

namespace SilverlightApplication104
{
    public class MessagePart
    {
        [Export("HelloMEF.Message")]
        public string Message = "Hello MEF";
    }
}
```

کدهای XAML صفحه‌ی اصلی برنامه در ادامه ذکر شده است و تنها شامل یک StackPanel ساده به عنوان میزبان افزونه‌های برنامه می‌باشد:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication104.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
     "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel Name="PluginPanel"/>
    </Grid>
</UserControl>
```

کدهای قسمت مونتاژ نهایی عملیات در ادامه ذکر شده‌اند:

MainPage.xaml.cs

```
using System.ComponentModel.Composition;
using System.Windows;
using System.Windows.Controls;
using System.Collections.Generic;

namespace SilverlightApplication104
{
    public partial class MainPage : UserControl
    {
```

```
        [ImportMany]
        public IEnumerable<UserControl> Plugins { get; set; }

        public MainPage()
        {
            InitializeComponent();
            this.Loaded += MainPage_Loaded;
        }

        void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            CompositionInitializer.SatisfyImports(this);

            foreach (var plugin in Plugins)
            {
                plugin.Margin = new Thickness(5);
                PluginPanel.Children.Add(plugin);
            }
        }
    }
}
```

ابتدا به کمک ویژگی ImportMany ، لیست هر تعداد افزونه‌ای از نوع User control را به صورت خودکار تشخیص داده و تهیه می‌کنیم (ImportMany تمام ویژگی‌های Export برنامه را به صورت خودکار بررسی می‌کند و اگر نوع آن‌ها از نوع User control باشد، به این لیست خواهد افزود).

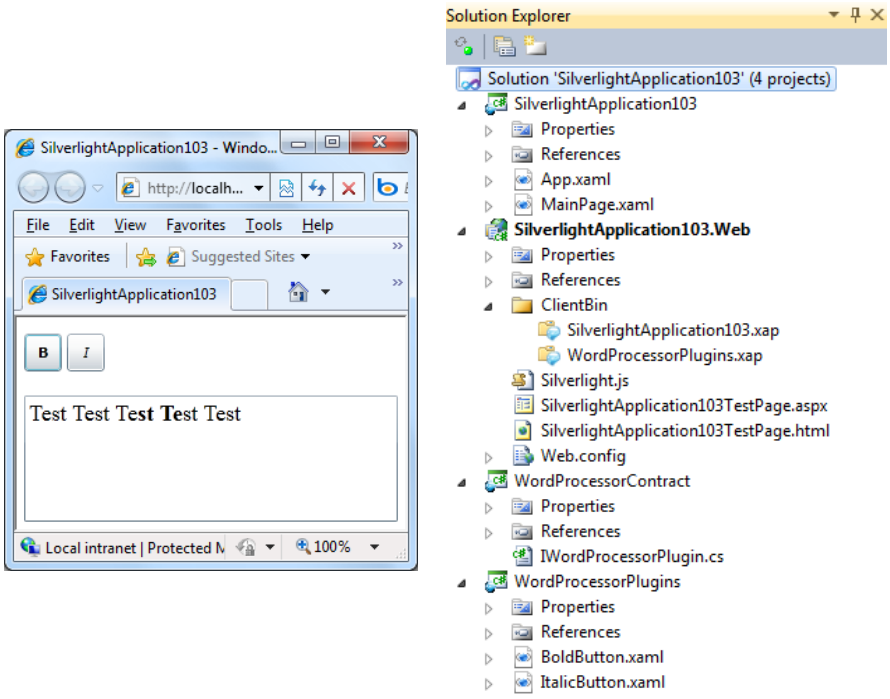
سپس به کمک متد CompositionInitializer.SatisfyImports به MEF اعلام خواهیم کرد که تمام Import و Export های همانند را تشخیص داده و عملیات ترکیب نهایی را به صورت خودکار انجام دهید. اکنون پس از وهله سازی‌های لازم، می‌توان کلیه‌ی افزونه‌های تشخیص داده شده را در طی یک حلقه به صفحه اضافه کرد.

مثال دوم - بررسی مفاهیم پیشرفته‌ی MEF

در این مثال می‌خواهیم یک برنامه‌ی افزونه پذیر را طراحی کنیم. برنامه‌ی اصلی از یک RichTextBox تشکیل شده است. این برنامه می‌تواند جهت بسط توانایی‌های خود افزونه‌هایی را جهت کار با RichTextBox دریافت کند. برای مثال قصد داریم دو دکمه را جهت تغییر وزن و حالت متن انتخابی RichTextBox صفحه اصلی، به صورت افزونه در اختیار برنامه قرار دهیم (شکل ۲). این افزونه‌ها نیز از یک برنامه‌ی دیگر (یک فایل XAP دیگر) به صورت پویا دریافت خواهند شد.

برای این منظور یک پروژه‌ی Silverlight جدید را به همراه پروژه‌ی ASP.NET آن ایجاد کنید. پروژه‌ی Web site آن از این لحاظ حائز اهمیت است که فایل XAP افزونه در پوشه‌ی ClientBin آن در کنار فایل XAP اصلی برنامه قرار خواهد گرفت و مدیریت آن به این شکل در این مثال ساده‌تر خواهد شد.

در حین طراحی یک سیستم مبتنی بر افزونه‌ها ابتدا باید مشخص کنیم که چه انتظاری را از افزونه‌ها داریم؛ چه امکاناتی در اختیار آن‌ها قرار خواهند گرفت و آن‌ها چه مواردی را باید پیاده سازی کرده و در اختیار برنامه‌ی اصلی قرار دهند. به همین جهت به Solution جاری، یک پروژه‌ی Silverlight class library جدید را به نام WordProcessorContract اضافه کنید. این پروژه نیاز به ارجاعی به اسمبلی استاندارد System.ComponentModel.Composition.dll دارد.



شکل ۲– نمایی از ساختار پروژه‌ی برنامه‌ی ویرایشگر افزونه پذیر

در این پروژه یک اینترفیس که بیانگر قرار داد طراحی افزونه‌ها است مشخص می‌گردد. به عبارت دیگر تنها افزونه‌هایی مجاز بوده و بارگذاری خواهند شد که اینترفیس IWordProcessorPlugin ذیل را پیاده سازی کنند.

IWordProcessorPlugin.cs

```
using System.ComponentModel.Composition;
using System.Windows.Controls;

namespace WordProcessorContract
{
    [InheritedExport]
    public interface IWordProcessorPlugin
    {
```

```
        RichTextBox RichTextBox { set; }
    }
}
```

نکته‌ی جدیدی که در طراحی این اینترفیس بکار گرفته شده است، استفاده از ویژگی InheritedExport می‌باشد. به این صورت کلیه کلاس‌هایی که این اینترفیس را پیاده سازی کنند نیازی به ذکر صریح ویژگی Export نداشته و این ویژگی به صورت خودکار به آن‌ها اعمال خواهد شد.

تا اینجا مشخص ساختیم که سیستم ما یک وهله از RichTextBox خود را در اختیار افزونه‌ها قرار داده و این افزونه‌ها می‌توانند اعمالی را با آن انجام دهند. برای مثال متن انتخابی آن‌را ضخیم کنند.

اکنون نوبت به طراحی یک سری افزونه‌ی جدید می‌رسد. به همین منظور یک پروژه‌ی Silverlight جدید را آغاز کرده و Web site آن‌را همان Web site پروژه اصلی درنظر بگیرید تا فایل XAP آن به صورت خودکار به پوشه‌ی ClientBin برنامه اصلی کپی شود. از آنجائیکه این برنامه قرار نیست یک برنامه‌ی معمولی Silverlight باشد، فایل‌های App.xaml و MainPage.xaml آن‌را حذف کنید. به علاوه ارجاعی را نیز به اسمبلی System.ComponentModel.Composition.dll اضافه نمائید. سپس دو User control جدید را به نام‌های BoldButton.xaml و ItalicButton.xaml به پروژه اضافه کنید. این فایل‌ها دو افزونه را در اختیار پروژه‌ی اصلی قرار می‌دهند.

کدهای XAML فایل BoldButton.xaml در ادامه ذکر شده است. این افزونه یک دکمه را که پس از کلیک بر روی آن متن انتخابی RichTextBox به حالت ضخیم درخواهد آمد، ارائه می‌دهد:

BoldButton.xaml

```
<UserControl x:Class="WordProcessorPlugins.BoldButton"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
     "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"  d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button Content="B" Width="30"
           Height="30" FontWeight="Bold" Click="Button_Click" />
    </Grid>
</UserControl>
```

کدهای کلاس BoldButton.xaml.cs در ادامه ذکر شده‌اند. این کلاس اینترفیس IWordProcessorPlugin را پیاده سازی می‌کند. بنابراین ارجاعی را به اسمبلی پروژه WordProcessorContract نیاز خواهد داشت:

BoldButton.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using WordProcessorContract;

namespace WordProcessorPlugins
{
    public partial class BoldButton : IWordProcessorPlugin
    {
        public BoldButton()
        {
            InitializeComponent();
        }

        #region IWordProcessorPlugin Members
        private RichTextBox _richTextBox;

        public RichTextBox RichTextBox
        {
            set
            {
                _richTextBox = value;
            }
        }
        #endregion

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            if(_richTextBox==null) return;
            _richTextBox.Selection.ApplyPropertyValue(
                   TextElement.FontWeightProperty,
                    FontWeights.Bold);
        }
    }
}
```

در ادامه کدهای XAML و CS مربوط به ItalicButton ذکر خواهند شد. توضیحات مرتبط با این فایل‌ها همانند BoldButton است و نکته‌ی جدیدی ندارند.

ItalicButton.xaml

```
<UserControl x:Class="WordProcessorPlugins.ItalicButton"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
```

```
       "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"  d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button Content="I" Width="30"
            Height="30" FontStyle="Italic" Click="Button_Click" />
    </Grid>
</UserControl>
```

پیاده سازی اینترفیس IWordProcessorPlugin در ادامه انجام شده است. همانطور که پیشتر نیز ذکر شد، با استفاده از ویژگی InheritedExport دیگر نیازی نخواهد بود به هر کدام از کلاس‌های افزونه‌های خود ویژگی Export را صریحا اضافه نمائیم.

ItalicButton.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using WordProcessorContract;
namespace WordProcessorPlugins
{
    public partial class ItalicButton : IWordProcessorPlugin
    {
        public ItalicButton()
        {
            InitializeComponent();
        }

        #region IWordProcessorPlugin Members
        private RichTextBox _richTextBox;

        public RichTextBox RichTextBox
        {
            set { _richTextBox = value; }
        }
        #endregion

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            if (_richTextBox == null) return;
            _richTextBox.Selection.ApplyPropertyValue(
               TextElement.FontStyleProperty,
                FontStyles.Italic);
        }
    }
}
```

اکنون نوبت به عملیات مونتاژ نهایی می‌رسد. به برنامه‌ی اصلی Silverlight مراجعه کرده و ارجاعاتی را به اسمبلی‌های زیر اضافه کنید:

- System.ComponentModel.Composition.dll
- System.ComponentModel.Composition.Initialization.dll
- WordProcessorContract.dll

نیازی به افزودن ارجاع مستقیمی به کلاس‌های افزونه نیست زیرا آن‌ها را به صورت پویا بارگذاری خواهیم نمود.

کدهای XAML صفحه‌ای اصلی برنامه را در ادامه ملاحظه می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication103.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
    "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <Grid.RowDefinitions>
            <RowDefinition  Height="56" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <StackPanel
            Orientation="Horizontal"
            HorizontalAlignment="Left"
            Height="48" Margin="4"
            Name="PluginPanel">
        </StackPanel>
        <RichTextBox
            Name="textBox" Grid.Row="1" IsReadOnly="False"
            FontSize="18" FontFamily="Times New Roman"
            TextWrapping="Wrap"
            Margin="6"
            >
            <Paragraph>Test Test Test Test Test</Paragraph>
        </RichTextBox>
    </Grid>
</UserControl>
```

در اینجا از یک StackPanel ساده برای افزودن افزونه‌ها به صفحه استفاده خواهیم کرد. کدهای متناظر با صفحه‌ی اصلی برنامه به شرح بعد هستند:

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.Composition;
using System.ComponentModel.Composition.Hosting;
using System.Windows;
using System.Windows.Controls;
using WordProcessorContract;

namespace SilverlightApplication103
{
    public partial class MainPage : IPartImportsSatisfiedNotification
    {
        [ImportMany(AllowRecomposition = true)]
        public IEnumerable<IWordProcessorPlugin>
               Plugins { set; get; }

        public MainPage()
        {
            InitializeComponent();
            initialize();
        }

        private void initialize()
        {
            var pluginXap =
               new Uri("WordProcessorPlugins.xap", UriKind.Relative);
            var deploymentCatalog = new DeploymentCatalog(pluginXap);
            deploymentCatalog.DownloadAsync();

            var container = new CompositionContainer(deploymentCatalog);
            CompositionHost.Initialize(container);
            CompositionInitializer.SatisfyImports(this);
        }

        #region IPartImportsSatisfiedNotification Members

        public void OnImportsSatisfied()
        {
            PluginPanel.Children.Clear();

            foreach (var plugin in Plugins)
            {
                if (plugin is UserControl)
                {
                    plugin.RichTextBox = textBox;
                    var uc = plugin as UserControl;
                    uc.Margin = new Thickness(2);
```

```
                    PluginPanel.Children.Add(uc);
                }
            }
        }

        #endregion
    }
}
```

توضیحات:

ابتدا توسط ImportMany از MEF درخواست خواهیم کرد که لطفا تمام افزونه‌هایی را که از نوع IWordProcessorPlugin هستند به صورت خودکار یافته و سپس لیستی از آن‌ها را ارائه دهید. از آنجائیکه این افزونه‌ها ارجاعی مستقیم به پروژه‌ی جاری ندارند و در طی برنامه به صورت پویا بارگذاری خواهند شد نیاز است تا AllowRecomposition = true را صریحا قید نمائیم تا پس از بارگذاری افزونه‌ها فرصت داشته باشیم تا Import/Export های متناظر را مونتاژ کنیم.

سپس متد initialize کار دریافت غیرهمزمان اطلاعات فایل افزونه را انجام داده، وهله‌هایی از افزونه‌ها را ایجاد کرده و نهایتا به این صورت لیست Plugins قابل استفاده خواهد بود. از آنجائیکه این عملیات غیر همزمان است نیاز می‌باشد تا بتوان در لحظه‌ی پایان کار، افزونه‌های اضافه شده را به صورت خودکار تشخیص داده و به StackPanel خود اضافه نمود. به همین جهت کلاس جاری، اینترفیس استاندارد IPartImportsSatisfiedNotification را پیاده سازی کرده است.