

در این [پست](#) با مفاهیم اولیه این کتابخانه آشنا شدید. برای بررسی و پیاده سازی مثال، ابتدا یک Blank Solution را ایجاد نمایید. فرض کنید قصد پیاده سازی یک پروژه بزرگ ماژولار را داریم. برای این کار لازم است مراحل زیر را برای طراحی ساختار مناسب پروژه دنبال نمایید.

نکته: آشنایی اولیه با مفاهیم [MEF](#) از ملزومات این بخش است.

«ابتدا یک Class Library به نام Views ایجاد نمایید و اینترفیس زیر را به صورت زیر در آن تعریف نمایید. این اینترفیس رابط بین کنترلر و View از طریق ViewModel خواهد بود.

```
public interface IBookView : IView
{
    void Show();
    void Close();
}
```

اینترفیس IView در مسیر System.Waf.Applications قرار دارد. در نتیجه از طریق nuget اقدام به نصب Package زیر نمایید:

Install-Package WAF

«حال در Solution ساخته شده یک پروژه از نوع WPF Application به نام Shell ایجاد کنید. با استفاده از نیوگت، Waf Package را نصب نمایید؛ سپس ارجاعی از اسمبلی Views را به آن ایجاد کنید. output type اسمبلی Shell را به نوع ClassLibrary تغییر داده، همچنین فایل‌های موجود در آن را حذف نمایید. یک فایل Xaml جدید را به نام BookShell ایجاد نمایید و کدهای زیر را در آن کپی نمایید:

```
<Window x:Class="Shell.BookShell"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Book View" Height="350" Width="525">
    <Grid>
        <DataGrid ItemsSource="{Binding Books}" HorizontalAlignment="Left" Margin="10,10,0,0"
        VerticalAlignment="Top" Width="400" Height="200">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Code" Binding="{Binding Code}"
                Width="100"></DataGridTextColumn>
                <DataGridTextColumn Header="Title" Binding="{Binding Title}"
                Width="300"></DataGridTextColumn>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Window>
```

این فرم فقط شامل یک دیتاگرید برای نمایش اطلاعات کتاب‌هاست. دیتای آن از طریق ViewModel تامین خواهد شد، در نتیجه ItemsSource آن به خاصیتی به نام Books باید شده است. حال ارجاعی به اسمبلی System.ComponentModel.Composition دهید. سپس در Code behind این فرم کدهای زیر را کپی کنید:

```
[Export(typeof(IBookView))]
[PartCreationPolicy(CreationPolicy.NonShared)]
public partial class BookShell : Window, IBookView
{
    public BookShell()
    {
        InitializeComponent();
    }
}
```

کاملاً واضح است که این فرم اینترفیس IBookView را پیاده سازی کرده است. از آنجاکه کلاس Window به صورت پیش فرض دارای متدهای Show و Close است در نتیجه نیازی به پیاده سازی مجدد متدهای IBookView نیست. دستور Export باعث می‌شود که این کلاس به عنوان وابستگی به Composition Container اضافه شود تا در جای مناسب بتوان از آن وهله سازی کرد. نکته‌ی مهم این است که به دلیل آنکه این کلاس، اینترفیس IBookView را پیاده سازی کرده است در نتیجه نوع Export این کلاس حتماً باید به صورت صریح از نوع IBookView باشد.

«یک Class Library به نام Models بسازید و بعد از ایجاد آن، کلاس زیر را به عنوان مدل Book در آن کپی کنید:

```
public class Book
{
    public int Code { get; set; }

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

«یک Class Library دیگر به نام ViewModels ایجاد کنید و همانند مراحل قبلی، Package مربوط به WAF را نصب کنید. سپس کلاسی به نام BookViewModel ایجاد نمایید و کدهای زیر را در آن کپی کنید (ارجاع به اسمبلی‌های Views و Models را فراموش نکنید):

```
[Export]
[Export(typeof(ViewModel<IBookView>))]
public class BookViewModel : ViewModel<IBookView>
{
    [ImportingConstructor]
    public BookViewModel(IBookView view)
        : base(view)
    {
    }

    public ObservableCollection<Book> Books { get; set; }
}
```

ViewModel مورد نظر از کلاس ViewModel of T ارث برده است. نوع این کلاس معادل نوع View مورد نظر ماست که در اینجا مقصود IBookView است. این کلاس شامل خاصیتی به نام ViewCore است که امکان فراخوانی متدها و خاصیت‌های View را فراهم می‌نماید. وظیفه اصلی کلاس پایه ViewModel، وهله سازی از View سپس ست کردن خاصیت DataContext در View مورد نظر به نمونه وهله سازی شده از ViewModel است. در نتیجه عملیات مقید سازی در Shell به درستی انجام خواهد شد. به دلیل اینکه سازنده پیش فرض در این کلاس وجود ندارد حتماً باید از ImportingConstructor استفاده نماییم تا CompositionContainer در هنگام عملیات وهله سازی Exception صادر نکند.

«بخش بعدی ساخت یک Class Library دیگر به نام Controllers است. در این Library نیز بعد از ارجاع به اسمبلی‌های زیر کتابخانه WAF را نصب نمایید.

Views

Models

ViewModels

System.ComponentModel.Composition

کلاسی به نام BookController بسازید و کدهای زیر را در آن کپی نمایید:

```
[Export]
public class BookController
{
    [ImportingConstructor]
    public BookController(BookViewModel viewModel)
    {
        ViewModelCore = viewModel;
    }

    public BookViewModel ViewModelCore
    {
    }
```

```

        get;
        private set;
    }

    public void Run()
    {
        var result = new List<Book>();
        result.Add(new Book { Code = 1, Title = "Book1" });
        result.Add(new Book { Code = 2, Title = "Book2" });
        result.Add(new Book { Code = 3, Title = "Book3" });

        ViewModelCore.Books = new ObservableCollection<Models.Book>(result);

        (ViewModelCore.View as IBookView).Show();
    }
}

```

نکته مهم این کلاس این است که BookViewModel به عنوان وابستگی این کنترلر تعریف شده است. در نتیجه در هنگام و هله سازی از این کنترلر Container مورد نظر یک و هله از BookViewModel را در اختیار آن قرار خواهد داد. در متد Run نیز ابتدا مقدار Book که به ItemsSource دیتا گرید در BookShell مقید شده است مقدار خواهد گرفت. سپس با فراخوانی متد Show از اینترفیس IBookView، متد Show در BookShell فراخوانی خواهد شد که نتیجه آن نمایش فرم مورد نظر است.

طراحی Bootstrapper

در پروژه‌های ماژولار Bootstrapper از ملزومات جدانشدنی این گونه پروژه هاست. برای این کار ابتدا یک WPF Application دیگر به نام Bootstrapper ایجاد نماید. سپس ارجاعی به اسمبلی‌های زیر را در آن قرار دهید:

«Controllers»

«Views»

«ViewModels»

«Shell»

«System.ComponentModel.Composition»

«نصب بسته WAF با استفاده از nuget»

حال یک کلاس به نام AppBootstrapper ایجاد نمایید و کدهای زیر را در آن کپی نمایید:

```

public class AppBootstrapper
{
    public CompositionContainer Container
    {
        get;
        private set;
    }

    public AggregateCatalog Catalog
    {
        get;
        private set;
    }

    public void Run()
    {
        Catalog = new AggregateCatalog();
        Catalog.Catalogs.Add(new AssemblyCatalog(Assembly.GetExecutingAssembly()));

        Catalog.Catalogs.Add(new AssemblyCatalog(String.Format("{0}\\{1}",
Environment.CurrentDirectory, "Shell.dll")));
        Catalog.Catalogs.Add(new AssemblyCatalog(String.Format("{0}\\{1}",
Environment.CurrentDirectory, "ViewModels.dll")));
        Catalog.Catalogs.Add(new AssemblyCatalog(String.Format("{0}\\{1}",
Environment.CurrentDirectory, "Controllers.dll")));

        Container = new CompositionContainer(Catalog);

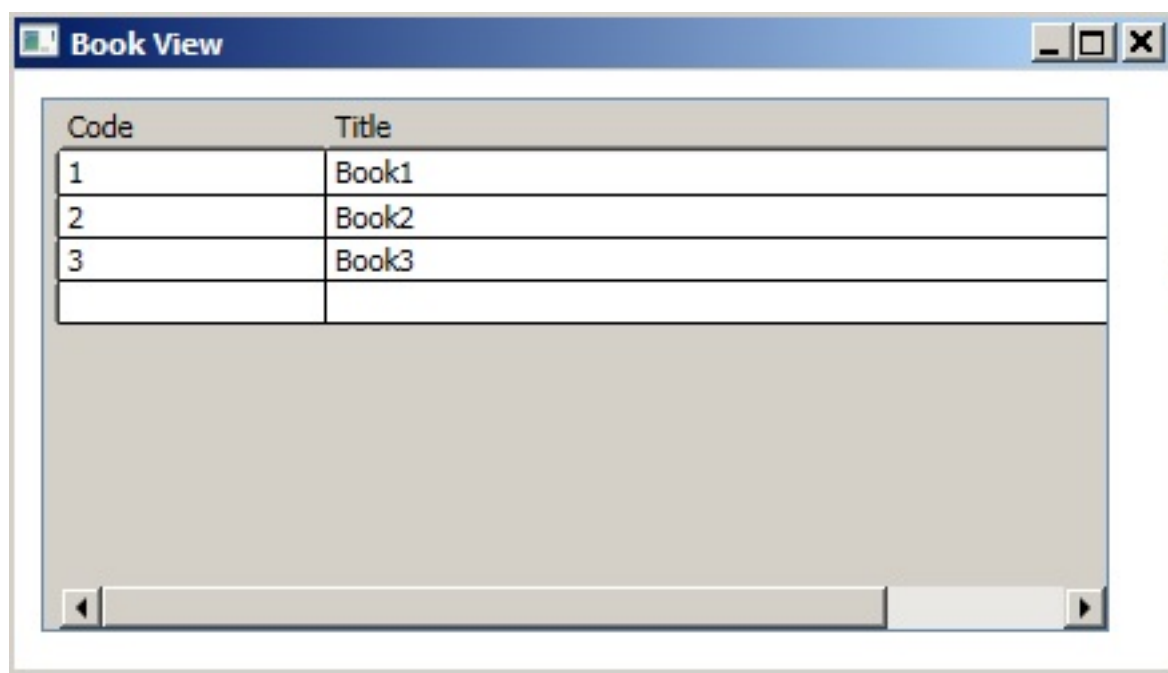
        var batch = new CompositionBatch();
        batch.AddExportedValue(Container);
        Container.Compose(batch);

        var bookController = Container.GetExportedValue<BookController>();
    }
}

```

```
bookController.Run();  
}  
}
```

اگر با MEF آشنا باشید کدهای بالا نیز برای شما مفهوم مشخصی دارند. در متد Run این کلاس ابتدا Catalog ساخته می‌شود. سپس با اسکن اسمبلی‌های مورد نظر تمام Exportها و Importهای لازم واکنشی شده و به Conrtainer مورد نظر رجیستر می‌شوند. در انتها نیز با وهله سازی از BookController و فراخوانی متد Run آن خروجی زیر نمایان خواهد شد.

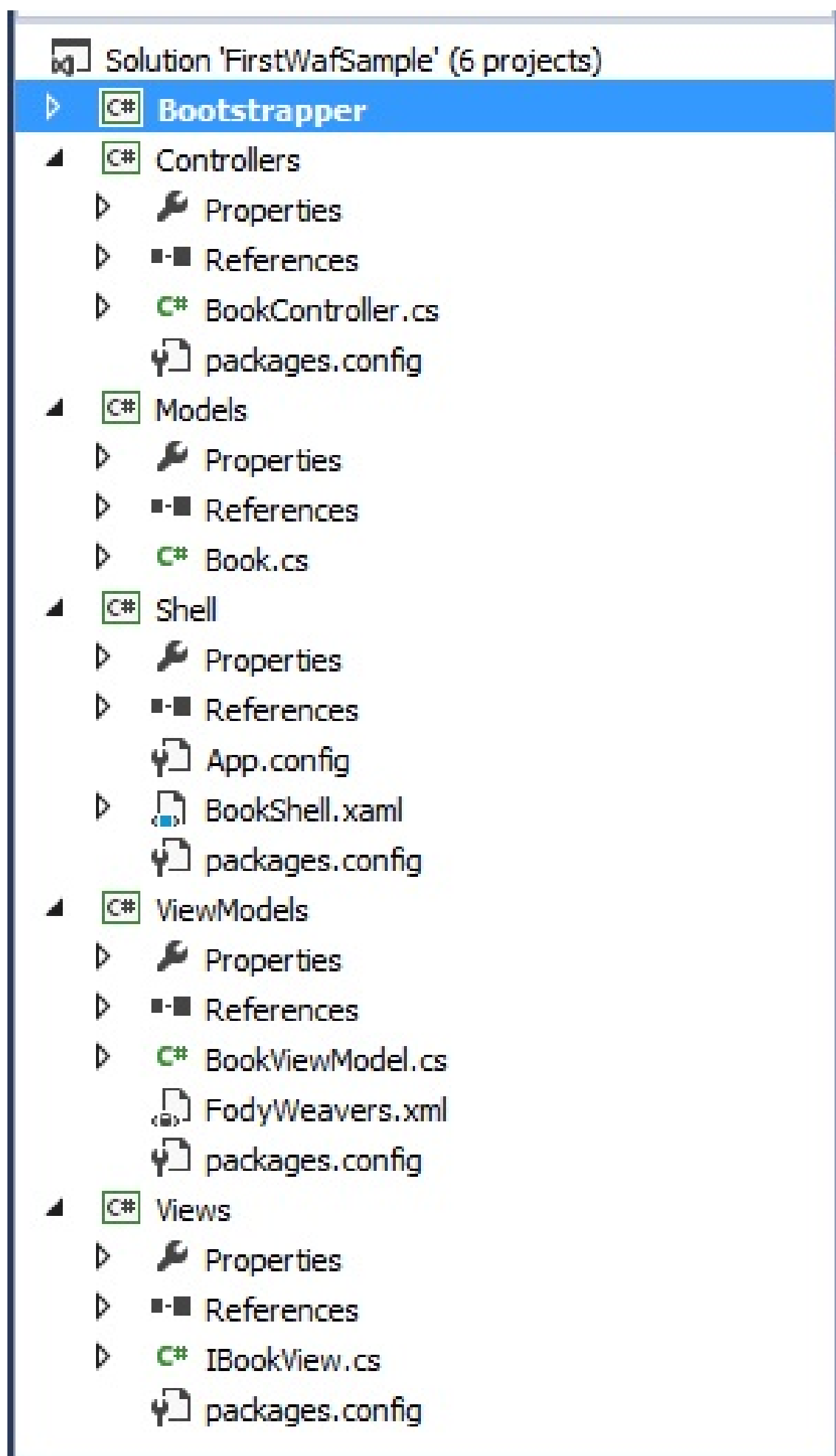


Code	Title
1	Book1
2	Book2
3	Book3

نکته بخش Startup را از فایل App.Xaml حذف نمایید و در متد Startup این فایل کد زیر را کپی کنید:

```
public partial class App : Application  
{  
    protected override void OnStartup(StartupEventArgs e)  
    {  
        new Bootstrapper.AppBootstrapper().Run();  
    }  
}
```

در پایان، ساختار پروژه به صورت زیر خواهد شد:



نکته: می‌توان بخش اسکن اسمبلی‌ها را توسط یک DirectoryCatalog به صورت زیر خلاصه کرد:

```
Catalog.Catalogs.Add(new DirectoryCatalog(Environment.CurrentDirectory));
```

در این صورت تمام اسمبلی‌های موجود در این مسیر اسکن خواهند شد.

نکته: می‌توان به جای جداسازی فیزیکی لایه‌ها آن‌ها را از طریق Directoryها به صورت منطقی در قالب یک اسمبلی نیز مدیریت کرد.

نکته: بهتر است به جای رفرنس مستقیم اسمبلی‌ها به Bootstrapper با استفاده از Pre post build در قسمت Build Event، اسمبلی‌های مورد نظر را در یک مسیر Build کپی نمایید که روش آن به تفصیل در این [پست](#) و این [پست](#) شرح داده شده است.

[دانلود سورس پروژه](#)