عنوان: **بررسی جزئیات تزریق وابستگی‌ها در قالب پروژه WPF Framework**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۷ ۱۰:۴۱
آدرس: www.dotnettips.info
برچسب‌ها: Design patterns, AOP, Dependency Injection, Entity framework, MVVM, WPF

در قالب طراحی شده، نه در کدهای Viewهای اضافه شده و نه در ViewModelها، اثری از کدهای مرتبط با تزریق وابستگی‌ها و یا حتی وهله سازی ViewModel مرتبط با یک View مشاهده نمی‌شود. در ادامه قصد داریم جزئیات پیاده سازی آن‌را مرور کنیم.

مدیریت خودکار وهله سازی ViewModelها

اگر به فایل MVVM\ViewModelFactory.cs قرار گرفته در پروژه Common مراجعه کنید، کدهای کلاسی که کار وهله سازی ViewModelها را انجام می‌دهد، مشاهده خواهید کرد:

```
using System.Windows;
using StructureMap;

namespace WpfFramework1999.Common.MVVM
{
    /// <summary>
    /// Stitches together a view and its view-model
    /// </summary>
    public class ViewModelFactory
    {
        private readonly FrameworkElement _control;

        /// <summary>
        /// سازنده کلاس تزریق وابستگی‌ها به ویوو مدل و وهله سازی آن
        /// </summary>
        /// <param name="control">وهله‌ای از شیءایی که باید کار تزریق وابستگی‌ها در آن انجام شود</param>
        public ViewModelFactory(FrameworkElement control)
        {
            _control = control;
        }

        /// <summary>
        /// وهله متناظر با ویوو مدل
        /// </summary>
        public IViewModel ViewModelInstance { get; private set; }

        /// <summary>
        /// کار تزریق خودکار وابستگی‌ها و وهله سازی ویوو مدل مرتبط انجام خواهد شد
        /// </summary>
        public void WireUp()
        {
            var viewName = _control.GetType().Name;
            var viewModelName = string.Concat(viewName, "ViewModel"); //قرار داد نامگذاری ما است

            if (!_control.IsLoaded)
            {
                _control.Loaded += (s, e) =>
                {
                    setDataContext(viewModelName);
                };
            }
            else
            {
                setDataContext(viewModelName);
            }
        }

        private void setDataContext(string viewModelName)
        {
            //کار تزریق خودکار وابستگی‌ها و وهله سازی ویوو مدل مرتبط انجام خواهد شد
            ViewModelInstance = ObjectFactory.TryGetInstance<IViewModel>(viewModelName);
            if (ViewModelInstance == null) // این صفحه ویوو مدل ندارد
                return;

            _control.DataContext = ViewModelInstance;
        }
    }
}
```

در این کلاس، یک وهله از صفحه‌ای که توسط کاربر درخواست شده‌است، در سازنده کلاس دریافت گردیده و سپس در متد WireUp، بر اساس قرارداد نامگذاری که پیشتر نیز عنوان شد، ViewModel متناظر با نام View از IoC Container استخراج و وهله سازی می‌گردد. سپس این وهله به DataContext صفحه انتساب داده می‌شود.
چند سؤال مهم:
- IoC Container از کجا می‌داند که ViewModelها در کجا قرار دارند؟
- این کلاس ViewModelFactory چگونه به وهله‌ای از یک صفحه درخواستی توسط کاربر دسترسی پیدا می‌کند و در کجا؟

IoC Container از کجا می‌داند که ViewModelها در کجا قرار دارند؟

اگر بحث سری جاری را از ابتدا دنبال کرده باشید، عنوان شد که ViewModelها را در این قالب، باید مشتق شده از کلاس پایه‌ای به نام BaseViewModel تهیه کنیم. برای مثال:

```
/// <summary>
/// ویوو مدل افزودن و مدیریت کاربران
/// </summary>
public class AddNewUserViewModel : BaseViewModel
```

این کلاس پایه که در فایل MVVM\BaseViewModel.cs پروژه Common قرار دارد، به نحو زیر آغاز شده است:

```
/// <summary>
/// کلاس پایه ویوو مدل‌های برنامه که جهت علامتگذاری آن‌ها برای سیم کشی‌های تزریق وابستگی‌های برنامه نیز استفاده می‌شود
/// </summary>
public abstract class BaseViewModel : DataErrorInfoBase, INotifyPropertyChanged, IViewModel
```

اگر دقت کنید در اینجا اینترفیس IViewModel نیز ذکر شده است. این اینترفیس برای علامتگذاری ViewModelها و یافتن خودکار آن‌ها توسط IoC Container مورد استفاده درنظر گرفته شده است. اگر به فایل Core\IocConfig.cs پروژه Infrastructure مراجعه کنید، چنین تنظیمی را در آن مشاهده خواهید نمود:

```
// Add all types that implement IView into the container,
// and name each specific type by the short type name.
scan.AddAllTypesOf<IViewModel>().NameBy(type => type.Name);
```

به این ترتیب StructureMap با اسکن اسمبلی Infrastructure کلیه کلاس‌های پیاده سازی کننده IViewModel را یافته و سپس آن‌ها را بر اساس نام متناظری که دارند، ذخیره می‌کند. با این تنظیم، اکنون در کلاس ViewModelFactory یک چنین کدی کار خواهد کرد:

```
//کار تزریق خودکار وابستگی‌ها و وهله سازی ویوو مدل مرتبط انجام خواهد شد
ViewModelInstance = ObjectFactory.TryGetInstance<IViewModel>(viewModelName);
```

کلاس ViewModelFactory چگونه به وهله‌ای از یک صفحه درخواستی توسط کاربر دسترسی پیدا می‌کند و در کجا؟

در اینجا قسمتی از کدهای فایل Core\FrameFactory.cs قرار گرفته در پروژه Infrastructure را ملاحظه می‌کنید:

```
namespace WpfFramework.Infrastructure.Core
{
    /// <summary>
    /// ایجاد یک کنترل فریم سفارشی که قابلیت تزریق وابستگی‌ها را به صورت خودکار دارد
    /// به همراه اعمال مسایل راهبری برنامه که از منوی اصلی دریافت می‌شوند
    /// </summary>
    public class FrameFactory : Frame
```

```
    {
        /// <summary>
        /// در اینجا می‌شود به وهله‌ای از صفحه‌ای که قرار است اضافه گردد دسترسی یافت
        /// </summary>
        protected override void OnContentChanged(object oldContent, object newContent)
        {
            base.OnContentChanged(oldContent, newContent);

            var newPage = newContent as FrameworkElement;
            if (newPage == null)
                return;

            _currentViewModelFactory = new ViewModelFactory(newPage);
            _currentViewModelFactory.WireUp(); //کار تزریق وابستگی‌ها و وهله سازی ویوو مدل مرتبط انجام
خواهد شد
        }
    }
}
```

در این کلاس، یک Frame سفارشی را طراحی کرده‌ایم؛ از این جهت که بتوان متد OnContentChanged آن‌را تحریف کرد. در این متد، newContent دقیقا وهله‌ای از صفحه جدیدی است که توسط کاربر درخواست شده‌است. خوب ... این وهله را داریم، بنابراین تنها کافی است آن‌را به کلاس ViewModelFactory ارسال کنیم و متد WireUp آن‌را بر روی وهله کلاس صفحه درخواستی فراخوانی نمائیم. به این ترتیب، صفحه‌ای نمایش داده خواهد شد که DataContext آن با وهله‌ای از ViewModel متناظر مقدار دهی شده‌است. از این جهت که این وهله سازی توسط IoC Container صورت می‌گیرد، کلیه وابستگی‌های تعریف شده در سازنده کلاس ViewModel نیز به صورت خودکار وهله سازی و مقدار دهی خواهند شد.

نهایتا فراخوانی متد IocConfig.Init، در فایل App.xaml.cs پروژه ریشه، در آغاز برنامه قرار گرفته است.

نظرات خوانندگان

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱:۳۴

وقتی پیج تغییر می‌کنه ، برای پیج قبلی چه اتفاقی میفته؟
چطور می‌تونیم یک page manager بنویسیم که بین صفحات باز سوییچ کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۱ ۸:۵۰

- از بین میره. تمام منابع مرتبط با اون هم مانند DbContext رها خواهند شد.
- به پروژه Infrastructure مراجعه کنید. یک کلاس Redirect برای هدایت به صفحات مختلف با برنامه نویسی طراحی شده. نمونه‌ای از استفاده از این کلاس رو در ViewModel مرتبط با لاگین به سیستم می‌تونید مشاهده کنید.

نویسنده: علیرضا پایدار
تاریخ: ۱۳۹۲/۰۸/۰۳ ۱۷:۳۱

من توی سازنده کلاس Locator کد زیر را نوشتم

```
SimpleIoc.Default.Register<IUnitOfWork, SampleContext>();
```

و بعد هر کجا نیاز دارم از این کد استفاده می‌کنم:

```
_uow = ServiceLocator.Current.GetInstance<IUnitOfWork>();
```

مشکلی که دارم وقتی SaveChanges را فراخوانی می‌کنم داخلش this.GetValidationErrors را فراخوانی کردم چون تنها یک instance از SampleContext ساخته میشه خطاهای قبلی دوباره وجود داره.
راهکاری هست خطاها را پاک کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۳ ۱۸:۴۷

- بله. باید Context تخریب و بازسازی مجدد شود؛ یا باید از سیستم Tracking آن کوئری بگیرید و سپس مواردی را که تغییر کرده‌اند به حالت اول برگردانید و یا روش سوم استفاده از متد Reload بر روی یک dbContext.Entry است.
+ طول عمر یک Context باید کوتاه باشد یا حداکثر در حد طول عمر یک فرم و نه طول عمر یک برنامه.
+ دو مطلب مرتبط
بایدها و نبایدهای استفاده از IoC Containers
نکته‌ای در مورد مدیریت طول عمر اشیاء در حالت HybridHttpOrThreadLocalScoped در برنامه‌های دسکتاپ

برای اینکه با SimpleIoc هم بتوانید وهله‌های متفاوتی را دریافت کنید و نه الزاما استفاده به صورت سینگلتون، باید به متد GetInstance آن یک کلید مشخص را ارسال کنید (مثلا نام فرم جاری یا آدرس صفحه جاری)؛ اگر کلیدی ارسال نشود، سینگلتون عمل می‌کند. در کل بهتر است از یک IoC Container بهتر استفاده کنید که در مورد طول عمر اشیاء راه‌حل‌های متفاوتی را به همراه دارد؛ مانند StrcutureMap. همچنین استفاده از آن به الگوی Service locator محدود نباشد.

نویسنده: علیرضا پایدار
تاریخ: ۱۳۹۲/۰۸/۰۳ ۱۹:۴۳

با کد زیر مشکلم حل میشه؟

```
ObjectFactory.Configure(cfg =>
            {
 cfg.For<IUnitOfWork>().Use(() => new SampleContext());
});
```

البته با استفاده از StrcutureMap

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۳ ۱۹:۴۸

در مورد سینگلتون نبودن، بله. نیازی به new هم نداره (حالت معمولی ذکر آن کافی است). در حالت پیش فرض، به ازای هربار فراخوانی، یک وهله جدید را ایجاد می‌کند. مگر اینکه در همینجا صریحا ذکر کنید که سینگلتون نیز مدنظر شما است یا مثلا حالت زنده نگه داشتن شیء در طول عمر یک درخواست وب، مهم است.