عنوان: **استفاده از Factories برای حذف Service locators در برنامه‌های WinForms**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۳ ۲۳:۳۴
آدرس: www.dotnettips.info
گروه‌ها: Design patterns, Dependency Injection, IoC

یک برنامه‌ی WinForms را درنظر بگیرید که از دو فرم تشکیل شده است.
فرم اول کار نمایش فرم 2 را به عهده دارد.
فرم دوم کار ارسال ایمیل را انجام می‌دهد. این ایمیل نیز از طریق سرویس ذیل فراهم می‌شود:

```
namespace WinFormsIoc.Services
{
    public interface IEmailsService
    {
        void SendEmail(string from, string to, string title, string message);
    }
}

namespace WinFormsIoc.Services
{
    public class EmailsService : IEmailsService
    {
        public void SendEmail(string from, string to, string title, string message)
        {
            //todo: ...
        }
    }
}
```

پیاده سازی متد SendEmail در اینجا مدنظر نیست. نکته‌ی مهم، مدیریت تامین و تزریق وابستگی‌های تعریف شده در سازنده‌ی آن است:

```
public partial class Form2 : Form
    {
        private readonly IEmailsService _emailsService;
        public Form2(IEmailsService emailsService)
        {
            _emailsService = emailsService;
            InitializeComponent();
        }
```

احتمالا شاید عنوان کنید که در فرم اول، زمانیکه نیاز است فرم دوم نمایش داده شود، می‌نویسیم new Form2 و در پارامتر آن با استفاده از متد ObjectFactory.GetInstance سازنده‌ی آن‌را فراهم می‌کنیم:

```
var form2 = new Form2(ObjectFactory.GetInstance<IEmailsService>());
form2.Show();
```

و یا اگر مدتی با IoC Containers کار کرده باشید، شاید پیشنهاد دهید که فقط بنویسید:

```
var form2 = ObjectFactory.GetInstance<Form2>();
form2.Show();
```

و همین! به صورت خودکار اگر n پارامتر تزریق شده هم در سازنده‌ی فرم دوم وجود داشته باشند، بر اساس تنظیمات اولیه‌ی IoC Container مورد استفاده، نمونه سازی شده و برنامه بدون مشکل کار خواهد کرد.

مشکل! این دو راه حل هیچکدام به عنوان تزریق وابستگی‌ها شناخته نمی‌شوند و به الگوی Service locator معروف هستند.
مشکل آن‌ها این است که کدهای ما در حال حاضر وابستگی مستقیمی به IoC container مورد استفاده پیدا کرده‌اند. در حالت اول ما خودمان دستی درخواست داده‌ایم که کدام وابستگی باید وهله سازی شود و در حالت دوم همانند حالت اول، کدهای ObjectFactory.GetInstance، مختص به یک IoC Container خاص است. نحوه‌ی صحیح کار با IoC Containerها باید به این نحو

باشد که یکبار در آغاز برنامه تنظیم شوند و در ادامه سایر کلاس‌های برنامه طوری کار کنند که انگار IoC Container ایی وجود خارجی ندارد.

راه حل: ObjectFactory.GetInstance را کپسوله کنید.

```
using System.Windows.Forms;

namespace WinFormsIoc.IoC
{
    public interface IFormFactory
    {
        T Create<T>() where T : Form;
    }
}

using System.Windows.Forms;
using StructureMap;

namespace WinFormsIoc.IoC
{
    public class FormFactory : IFormFactory
    {
        public T Create<T>() where T : Form
        {
            return ObjectFactory.GetInstance<T>();
        }
    }
}
```

در اینجا یک اینترفیس را تعریف کرده‌ایم که متد ایجاد وهله‌ای از یک فرم را ارائه می‌دهد. پیاده سازی آن در برنامه‌ای که از StructureMap استفاده می‌کند، مطابق کلاس FormFactory است. اگر IoC Container دیگری باشد، فقط باید این پیاده سازی را تغییر دهید و نه کل برنامه را. اکنون برای استفاده از آن، IFormFactory را در سازنده‌ی کلاسی که نیاز دارد فرم‌های دیگر را نمایش دهد، تزریق می‌کنیم:

```
using System;
using System.Windows.Forms;
using WinFormsIoc.IoC;

namespace WinFormsIoc
{
    public partial class Form1 : Form
    {
        private readonly IFormFactory _formFactory;
        public Form1(IFormFactory formFactory)
        {
            _formFactory = formFactory;
            InitializeComponent();
        }

        private void btnShowForm2_Click(object sender, EventArgs e)
        {
            var form2 = _formFactory.Create<Form2>();
            form2.Show();
        }
    }
}
```

در کدهای فوق، فرم اول برنامه را ملاحظه می‌کنید که قرار است فرم دوم را نمایش دهد. IFormFactory در سازنده‌ی آن تزریق شده‌است. با فراخوانی متد Create آن، فرم دوم برنامه به همراه تمام وابستگی‌های تزریق شده‌ی در سازنده‌ی آن وهله سازی می‌شوند.
نکته‌ی مهم این کدها عدم وابستگی مستقیم آن به هیچ نوع IoC Container خاصی است. این فرم اصلا نمی‌داند که IoC Container ایی در برنامه وجود دارد یا خیر.

مشکل! با تغییر سازنده‌ی Form1 برنامه دیگر کامپایل نمی‌شود!

اگر فایل Program.cs را باز کنید، یک چنین سطری را دارد:

```
Application.Run(new Form1());
```

چون سازنده‌ی فرم یک، اکنون پارامتر جدیدی پیدا کرده‌است، در اینجا می‌توان ObjectFactory.GetInstance را مستقیما بکار برد (در این حالت خاص که مرتبط است به کلاس آغازین برنامه، با توجه به اینکه وهله سازی آن مستقیما و خارج از کنترل ما انجام می‌شود، دیگر چاره‌ای نداریم و مجبور هستیم از الگوی Service locator استفاده کنیم).

```
Application.Run(ObjectFactory.GetInstance<Form1>());
```

مثال کامل این بحث را از اینجا می‌توانید دریافت کنید

WinFormsIoc.zip