عنوان: **اصول طراحی یک سیستم افزونه پذیر به کمک StructureMap**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۰۱ ۹:۳۷
آدرس: www.dotnettips.info
گروه‌ها: Design patterns, Dependency Injection, IoC

قصد داریم سیستمی را طراحی کنیم که افزونه‌های خود را در زمان اجرا از مسیری خاص خوانده و سپس وهله‌های آن‌هارا جهت استفاده در دسترس قرار دهد. برنامه‌ای که در اینجا مورد بررسی قرار می‌گیرد، یک برنامه‌ی WinForms ساده است؛ به نام WinFormsWithPluginSupport. اما اصول کلی مطرح شده، در تمام فناوری‌های دیگر دات نتی نیز کاربرد دارد و یکسان است.

تهیه قرارداد

یک پروژه‌ی Class library به نام PluginsBase را به Solution جاری اضافه کنید. به آن اینترفیس قرار داد پلاگین‌های برنامه خود را اضافه نمائید. برای مثال:

```
namespace PluginsBase
{
    public interface IPlugin
    {
        string Name { get; }
        void Run();
    }
}
```

هر پلاگین دارای یک نام یا توضیح خاص خود خواهد بود به همراه متدی برای اجرای منطق مرتبط با آن.

تهیه سه پلاگین جدید

به Solution جاری سه پروژه‌ی مجزای Class library با نام‌های plugin1 تا 3 را اضافه کنید. در ادامه به هر پلاگین، ارجاعی را به اسمبلی PluginsBase، برای دریافت قرارداد پیاده سازی منطق پلاگین، اضافه نمائید. هدف این است که اینترفیس IPlugin، در این اسمبلی‌ها قابل دسترسی شود.
هر پلاگین هم دارای برای مثال کدهایی مانند کد ذیل خواهد بود که در آن صرفا نام آن‌ها به 2 و 3 تنظیم می‌شود.

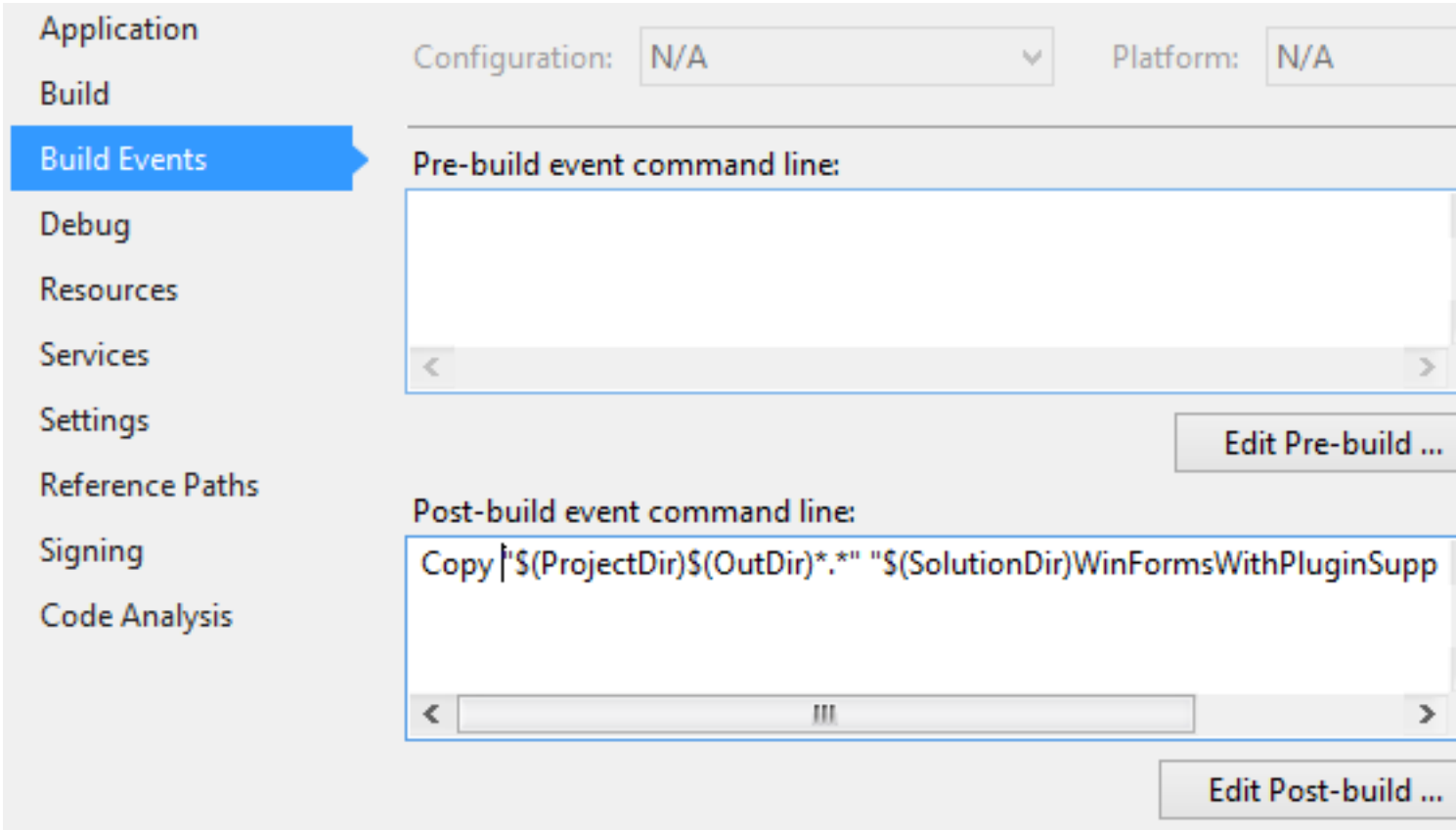
```
using PluginsBase;

namespace Plugin1
{
    public class Plugin1Main : IPlugin
    {
        public string Name
        {
            get { return "Test 1"; }
        }

        public void Run()
        {
            // todo: ...
        }
    }
}
```

کپی خودکار پلاگین‌ها به پوشه‌ی مخصوص آن‌ها

به پروژه‌ی WinFormsWithPluginSupport مراجعه کنید. در پوشه‌ی bin\debug آن یک پوشه‌ی جدید به نام Plugins ایجاد نمائید. بدیهی است هربار که پلاگین‌های برنامه تغییر کنند نیاز است اسمبلی‌های نهایی آن‌ها را به این پوشه کپی نمائیم. اما راه بهتری نیز وجود دارد. به خواص هر کدام از پروژه‌های پلاگین مراجعه کرده و برگه‌ی Build events را باز کنید.



در اینجا قسمت post-build event را به نحو ذیل تغییر دهید:

```
Copy "$(ProjectDir)$(OutDir)$(TargetName).*"
"$(SolutionDir)WinFormsWithPluginSupport\bin\debug\Plugins"
```

این کار را برای هر سه پلاگین تکرار کنید.
به این ترتیب هربار که پلاگین جاری کامپایل شود، پس از آن به صورت خودکار به پوشه‌ی plugins تعیین شده، کپی می‌شود و دیگر نیازی به کپی دستی نخواهد بود.
تنظیم فوق، تنها اسمبلی اصلی پروژه را به پوشه‌ی bin\debug\plugins کپی می‌کند. اگر می‌خواهید تمام فایل‌ها کپی شوند، از تنظیم ذیل استفاده کنید:

```
Copy "$(ProjectDir)$(OutDir)*.*" "$(SolutionDir)WinFormsWithPluginSupport\bin\debug\Plugins"
```

اضافه کردن وابستگی‌های اصلی پروژه‌ی WinForms

در ادامه بسته‌ی نیوگت StructureMap را به پروژه‌ی WinForms از طریق دستور ذیل اضافه کنید:

```
PM> install-package structuremap
```

همچنین این پروژه تنها نیاز دارد ارجاع مستقیمی را به اسمبلی PluginsBase ابتدای مطلب داشته باشد. از آن، جهت تنظیمات اولیه یافتن افزونه‌ها استفاده می‌کنیم.

تعریف محل ثبت پلاگین‌ها

روش‌های متفاوتی برای کار با StructureMap وجود دارد. یکی از آن‌ها تعریف کلاسی است مشتق شده از کلاس Registry آن به نحو ذیل:

```
using System.IO;
using System.Windows.Forms;
using PluginsBase;
using StructureMap.Configuration.DSL;
using StructureMap.Graph;

namespace WinFormsWithPluginSupport.Core
{
    public class PluginsRegistry : Registry
    {
        public PluginsRegistry()
        {
            this.Scan(scanner =>
            {
                scanner.AssembliesFromPath(
                    path: Path.Combine(Application.StartupPath, "plugins"),
                    // یک اسمبلی نباید دوبار بارگذاری شود
                    assemblyFilter: assembly =>
                    {
                        return !assembly.FullName.Equals(typeof(IPlugin).Assembly.FullName);
                    });
                scanner.AddAllTypesOf<IPlugin>().NameBy(item => item.FullName);
            });
        }
    }
}
```

در اینجا مشخص کرده‌ایم که اسمبلی‌های پوشه plugins را که یک سطح پایین‌تر از پوشه‌ی اجرایی برنامه قرار می‌گیرند، خوانده و در این بین آن‌هایی را که پیاده سازی از اینترفیس IPlugin دارند، در دسترس قرار دهد.

یک نکته‌ی مهم

در قسمت assemblyFilter تعیین کرده‌ایم که اسمبلی تکراری PluginBase بارگذاری نشود. چون این اسمبلی هم اکنون به برنامه‌ی WinForms ارجاع دارد. رعایت این نکته جهت رفع تداخلات آتی بسیار مهم است. همچنین این فایل در پوشه‌ی Plugins نیز نباید حضور داشته باشد وگرنه شاهد بارگذاری افزونه‌ها نخواهید بود.

سپس نیاز به وهله سازی Container آن و معرفی این کلاس PluginsRegistry می‌باشد:

```
using System;
using System.Threading;
using StructureMap;

namespace WinFormsWithPluginSupport
{
    public static class IocConfig
    {
        private static readonly Lazy<Container> _containerBuilder =
            new Lazy<Container>(defaultContainer, LazyThreadSafetyMode.ExecutionAndPublication);

        public static IContainer Container
        {
            get { return _containerBuilder.Value; }
        }

        private static Container defaultContainer()
        {
            return new Container(x =>
            {
                x.AddRegistry<PluginsRegistry>();
            });
        }
    }
}
```

تنظیمات ابتدایی WinForms برای دسترسی به امکانات StructureMap

به فرم اصلی برنامه مراجعه کرده و به سازنده‌ی آن IContainer را اضافه کنید. از این اینترفیس جهت دسترسی به پلاگین‌های برنامه استفاده خواهیم کرد.

```
using System.Windows.Forms;
using StructureMap;

namespace WinFormsWithPluginSupport
{
    public partial class FrmMain : Form
    {
        private readonly IContainer _container;

        public FrmMain(IContainer container)
        {
            _container = container;
            InitializeComponent();
        }
    }
}
```

اکنون برنامه دیگر کامپایل نخواهد شد؛ چون در فایل Program.cs وهله سازی مستقیمی از FrmMain وجود دارد. این وهله سازی را اکنون به StructureMap محول می‌کنیم تا مشکل برطرف شود:

```
using System;
using System.Windows.Forms;

namespace WinFormsWithPluginSupport
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(IocConfig.Container.GetInstance<FrmMain>());
        }
    }
}
```

زمانیکه از متد IocConfig.Container.GetInstance استفاده می‌شود، تا هر تعداد سطحی که تعریف شده، سازنده‌های کلاس‌های مرتبط وهله سازی می‌شوند. در اینجا نیاز است سازنده‌ی کلاس FrmMain وهله سازی شود. چون IContainer اینترفیس اصلی خود StructureMap است، آن‌را شناخته و به صورت خودکار وهله سازی می‌کند. اگر اینترفیس دیگری را ذکر کنید، نیاز است مطابق معمول آن‌را در کلاس IocConfig و متد defaultContainer آن معرفی نمائید.

بارگذاری و اجرای افزونه‌ها

دو دکمه‌ی Run و ReLoad را به فرم اصلی برنامه با کدهای ذیل اضافه کنید:

```
using System.Linq;
using System.Windows.Forms;
using PluginsBase;
using StructureMap;
using WinFormsWithPluginSupport.Core;

namespace WinFormsWithPluginSupport
{
    public partial class FrmMain : Form
    {
        private readonly IContainer _container;

        public FrmMain(IContainer container)
        {
```

```
            _container = container;
            InitializeComponent();
        }

        private void BtnRun_Click(object sender, System.EventArgs e)
        {
            var plugins = _container.GetAllInstances<IPlugin>().ToList();
            foreach (var plugin in plugins)
            {
                plugin.Run();
            }
        }

        private void BtnReload_Click(object sender, System.EventArgs e)
        {
            _container.EjectAllInstancesOf<IPlugin>();
            _container.Configure(x =>
                x.AddRegistry<PluginsRegistry>()
            );
        }
    }
}
```

در اینجا توسط متد container.GetAllInstances می‌توان به تمامی وهله‌های پلاگین‌های بارگذاری شده، دسترسی یافت و سپس آن‌ها را اجرا کرد.

همچنین در متد ReLoad نحوه‌ی بارگذاری مجدد این پلاگین‌ها را در صورت نیاز مشاهده می‌کنید.

اگر برنامه را اجرا کردید و پلاگینی بارگذاری نشد، به دنبال اسمبلی‌های تکراری بگردید. برای مثال PluginsBase نباید هم در پوشه‌ی اصلی اجرایی برنامه حضور داشته باشد و هم در پوشه‌ی پلاگین‌ها.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

WinFormsWithPluginSupport.zip