

عنوان: آشنایی با M.A.F - قسمت دوم

نویسنده: وحید نصیری

تاریخ: ۱۶:۱۷:۰۰ ۱۳۸۸/۱۰/۰۲

آدرس: www.dotnettips.info

برچسب‌ها: MAF

قسمت قبل

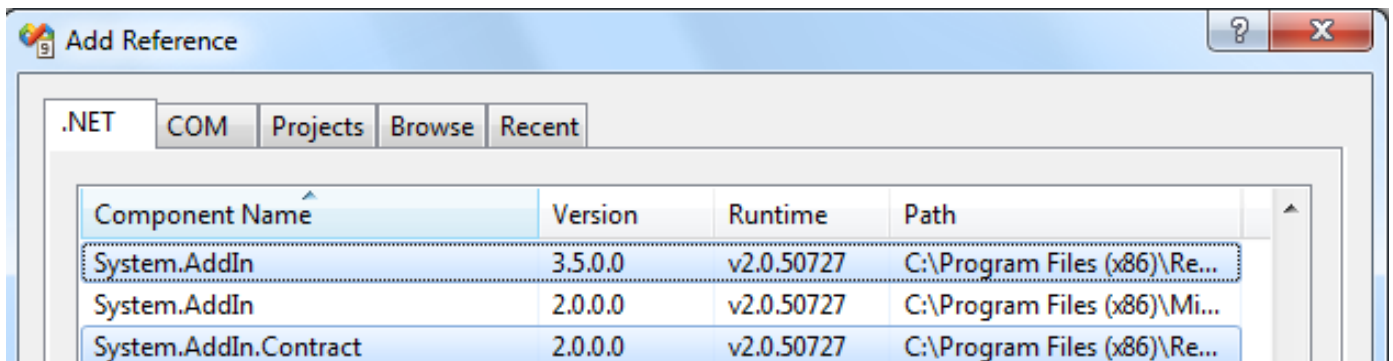
بیشتر آشنایی با یک سری از اصطلاحات مرتبط با فریم ورک MAF بود و همچنین نحوه‌ی کلی استفاده از آن. در این قسمت یک مثال ساده را با آن پیاده سازی خواهیم کرد و فرض قسمت دوم بر این است که افزونه‌ی [Visual Studio Pipeline Builder](#) را نیز نصب کرده‌اید.

یک نکته پیش از شروع:

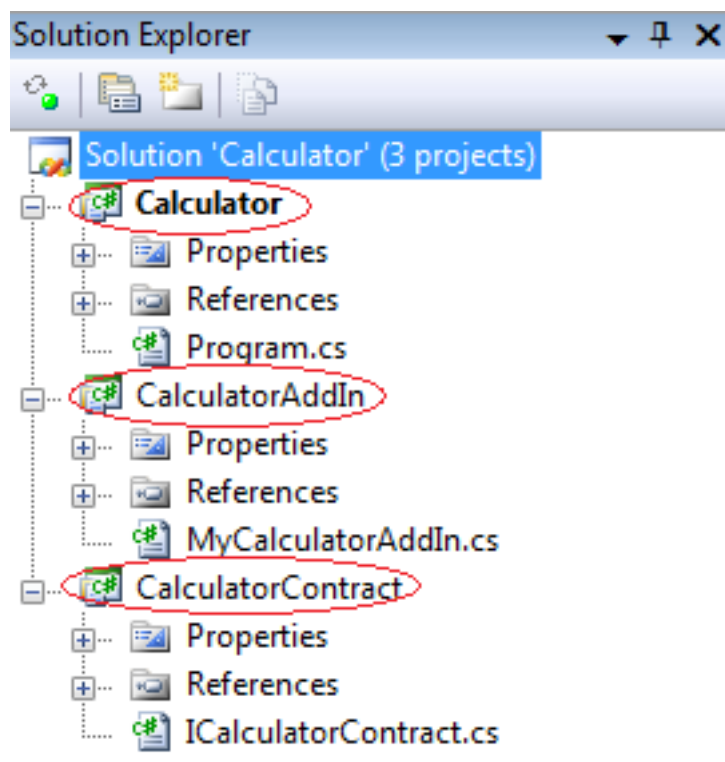
- اگر افزونه‌ی Visual Studio Pipeline Builder پس از نصب به منوی Tools اضافه نشده است، یک پوشه‌ی جدید را به نام Addins در مسیر Documents\Visual Studio 2008 ایجاد کرده و سپس فایل‌های آن را در این مسیر کپی کنید.

ساختار اولیه یک پروژه MAF

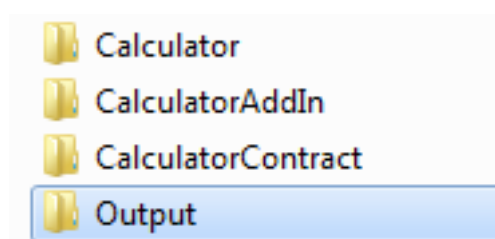
- پروژه‌هایی که از MAF استفاده می‌کنند، نیاز به ارجاعاتی به دو اسمبلی استاندارد System.AddIn.dll و System.AddIn.Contract.dll دارند (مطابق شکل زیر):



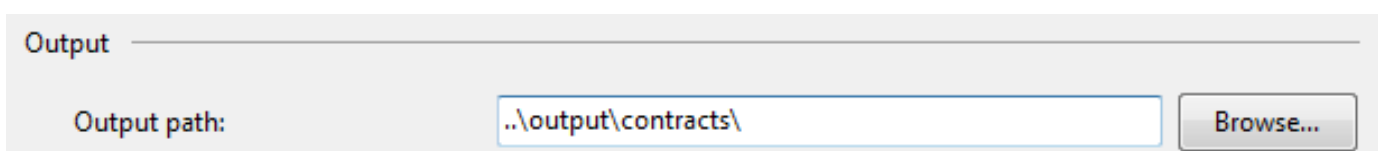
- ساختار آغازین یک پروژه MAF از سه پروژه تشکیل می‌شود که توسط افزونه‌ی Visual Studio Pipeline Builder به 7 پروژه بسط خواهد یافت. این سه پروژه استاندارد آغازین شامل موارد زیر هستند:



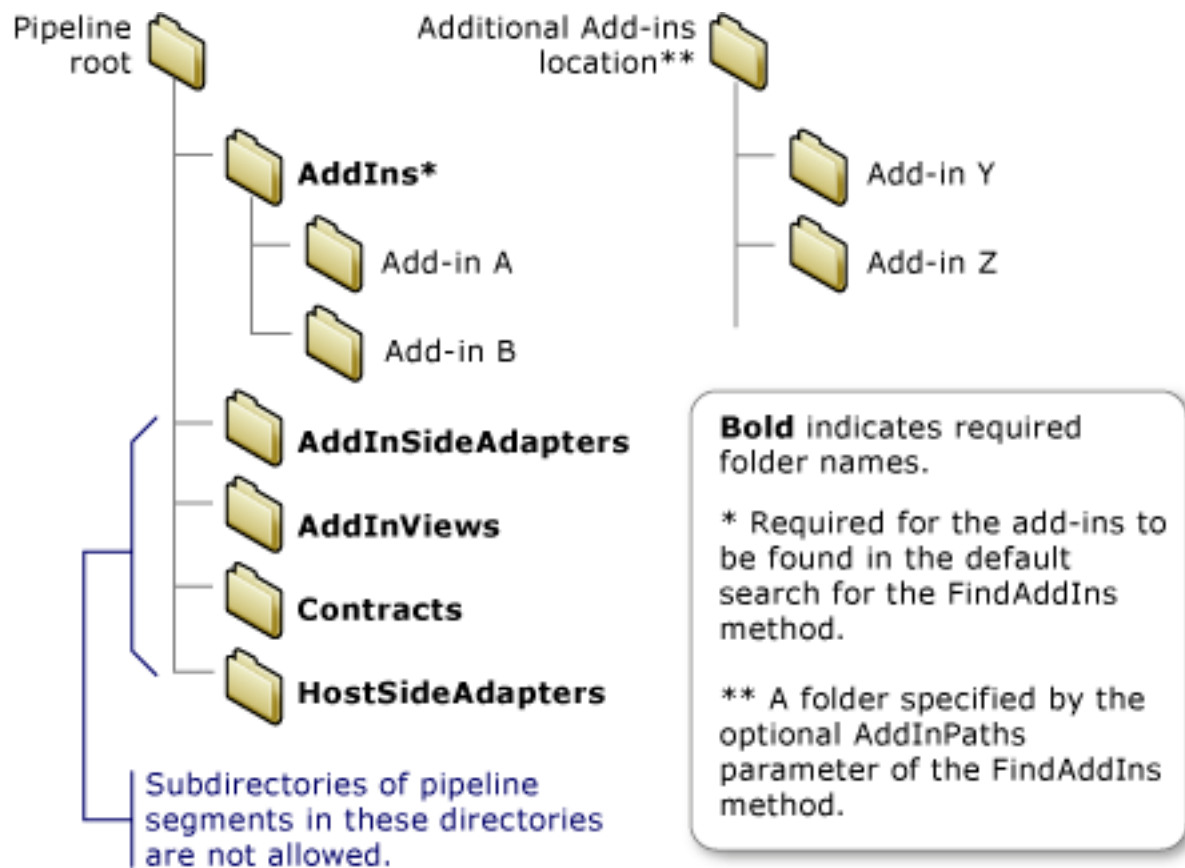
- هاست: همان برنامه‌ی اصلی که قرار است از افزونه استفاده کند.
- قرار داد: نحوه‌ی تعامل هاست و افزونه در این پروژه تعریف می‌شود. (یک پروژه از نوع class library)
- افزونه: کار پیاده سازی قرار داد را عهده دار خواهد شد. (یک پروژه از نوع class library)
- همچنین مرسوم است جهت مدیریت بهتر خروجی‌های حاصل شده یک پوشه Output را نیز به این solution اضافه کنند:



اکنون با توجه به این محل خروجی، به خواص Build سه پروژه موجود مراجعه کرده و مسیر Build را اندکی خواهیم کرد (هر سه مورد بهتر است اصلاح شوند)، برای مثال:



نکته‌ی مهم هم اینجا است که خروجی host باید به ریشه این پوشه تنظیم شود و سایر پروژه‌ها هر کدام خروجی خاص خود را در پوشه‌ای داخل این ریشه باید ایجاد کنند.



تا اینجا قالب اصلی کار آماده شده است. قرارداد ما هم به شکل زیر است (ویژگی AddInContract آن نیز نباید فراموش شود):

```
using System.AddIn.Pipeline;
using System.AddIn.Contract;

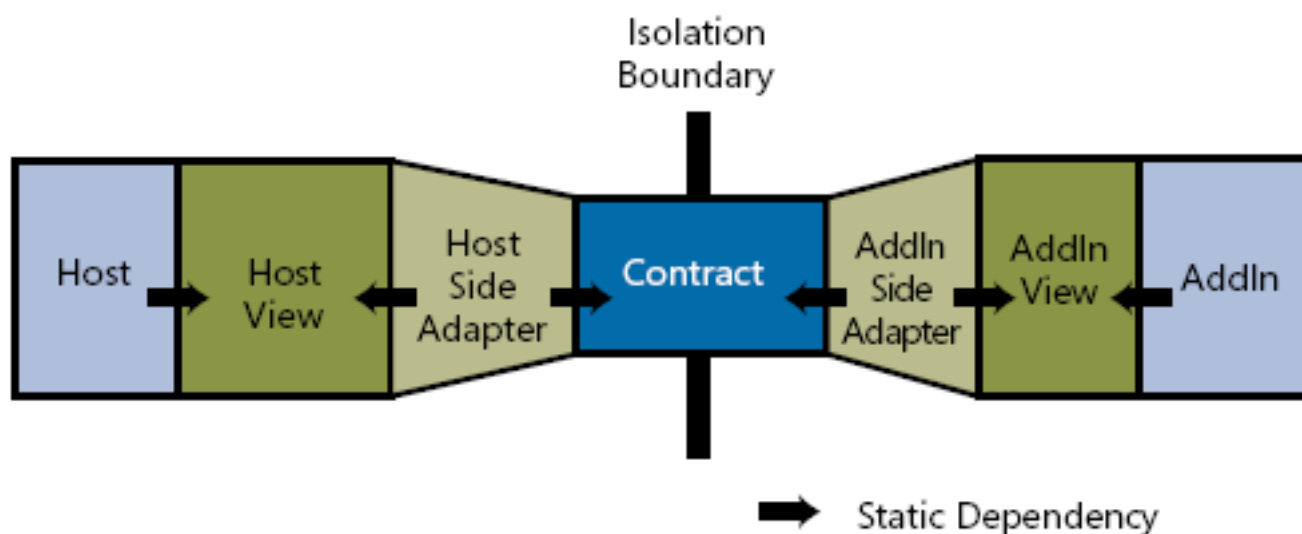
namespace CalculatorContract
{
    [AddInContract]
    public interface ICalculatorContract : IContract
    {
        double Operate(string operation, double a, double b);
    }
}
```

به عبارت دیگر برنامه‌ای محاسباتی داریم (هاست) که دو عدد double را در اختیار افزونه‌های خودش قرار می‌دهد و سپس این افزونه‌ها یک عملیات ریاضی را بر روی آن‌ها انجام داده و خروجی را بر می‌گردانند. نوع عملیات توسط آرگومان operation مشخص می‌شود. این آرگومان به کلیه افزونه‌های موجود ارسال خواهد شد و احتمالاً یکی از آن‌ها این مورد را پیاده سازی کرده است. در غیر اینصورت یک استثنای عملیات پیاده سازی نشده صادر می‌شود.

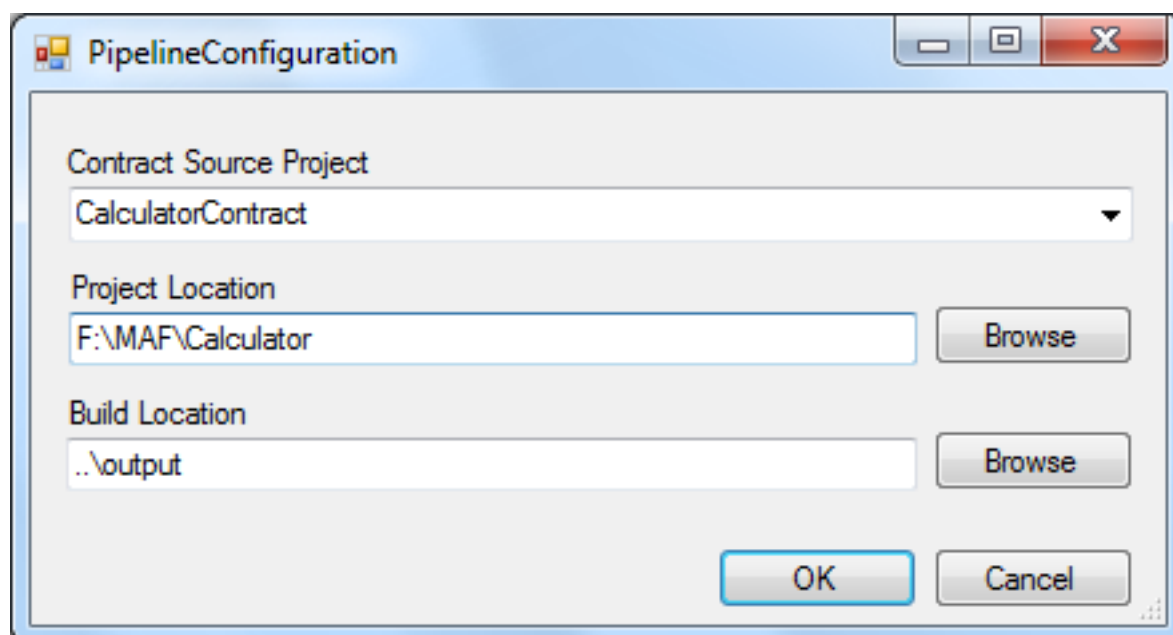
البته روش بهتر طراحی این افزونه، اضافه کردن متد یا خاصیتی جهت مشخص کردن نوع و یا انواع عملیات پشتیبانی شده توسط افزونه است که جهت سادگی این مثال، به این طراحی ساده اکتفا می‌شود.

ایجاد pipeline

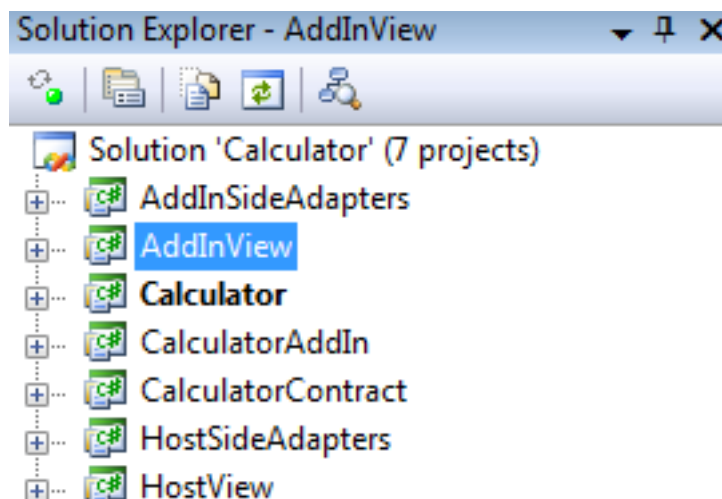
اگر قسمت قبل را مطالعه کرده باشید، یک راه حل مبتنی بر MAF از 7 پروژه تشکیل می‌شود که عمده‌ترین خاصیت آن‌ها مقاوم کردن سیستم در مقابل تغییرات نگارش قرارداد است. در این حالت اگر قرار داد تغییر کند، نه هاست و نه افزونه‌ی قدیمی، نیازی به تغییر در کدهای خود نخواهند داشت و این پروژه‌های میانی هستند که کار وفق دادن (adapters) نهایی را برعهده می‌گیرند.



برای ایجاد خودکار View ها و همچنین Adapters ، از افزونه‌ی Visual Studio Pipeline Builder که پیشتر معرفی شد استفاده خواهیم کرد.



سه گزینه‌ی آن هم مشخص هستند. نام پروژه‌ی قرارداد، مسیر پروژه‌ی هاست و مسیر خروجی نهایی معرفی شده. پیش از استفاده از این افزونه نیاز است تا یکبار solution مورد نظر کامپایل شود. پس از کلیک بر روی دکمه‌ی OK، پروژه‌های ذکر شده ایجاد خواهند شد:



پس از ایجاد این پروژه‌ها، نیاز به اصلاحات مختصری در مورد نام اسمبلی و فضای نام هر کدام می‌باشد؛ زیرا به صورت پیش فرض هر کدام به نام template نامگذاری شده‌اند:

Assembly name:	Default namespace:
AddInView	AddInView

پیاده سازی افزونه

قالب کاری استفاده از این فریم ورک آماده است. اکنون نوبت به پیاده سازی یک افزونه می‌باشد. به پروژه AddIn مراجعه کرده و ارجاعی را به اسمبلی AddInView خواهیم افزود. به این صورت افزونه‌ی ما به صورت مستقیم با قرارداد سروکار نداشته و ارتباطات، در راستای همان pipeline تعریف شده، جهت مقاوم شدن در برابر تغییرات صورت می‌گیرد:

```
using System;
using CalculatorContract.AddInView;
using System.AddIn;

namespace CalculatorAddIn
{
    [AddIn]
    public class MyCalculatorAddIn : ICalculator
    {
        public double Operate(string operation, double a, double b)
        {
            throw new NotImplementedException();
        }
    }
}
```

```
    }
}
}
```

در اینجا افزونه‌ی ما باید اینترفیس ICalculator مربوط به AddInView را پیاده سازی نماید که برای مثال خواهیم داشت:

```
using System;
using CalculatorContract.AddInView;
using System.AddIn;

namespace CalculatorAddIn
{
    [AddIn("افزونه یک", Description = "توضیحات", Publisher = "نویسنده", Version = "نگارش یک")]
    public class MyCalculatorAddIn : ICalculator
    {
        public double Operate(string operation, double a, double b)
        {
            switch (operation)
            {
                case "+":
                    return a + b;
                case "-":
                    return a - b;
                case "*":
                    return a * b;
                default:
                    throw new NotSupportedException("عملیات مورد نظر توسط این افزونه پشتیبانی نمی‌شود");
            }
        }
    }
}
```

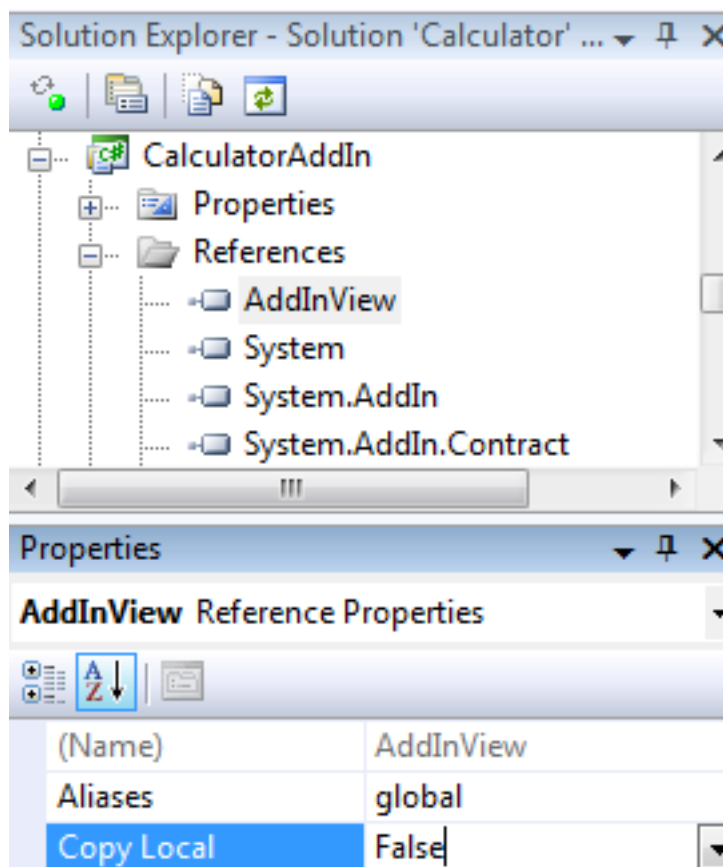
همانطور که در قسمت قبل نیز ذکر شد، این کلاس باید با ویژگی AddIn مزین شود که توسط آن می‌توان توضیحاتی در مورد نام ، نویسنده و نگارش افزونه ارائه داد.

استفاده از افزونه‌ی تولید شده

هاست برای استفاده از افزونه‌هایی با قرارداد ذکر شده، مطابق pipeline پروژه، نیاز به ارجاعی به اسمبلی HostView دارد و در اینجا نیز هاست به صورت مستقیم با قرارداد کاری نخواهد داشت. همچنین هاست هیچ ارجاع مستقیمی به افزونه‌ها نداشته و بارگذاری و مدیریت آن‌ها به صورت پویا انجام خواهد شد.

نکته‌ی مهم!

در هر دو ارجاع به HostView و یا AddInView باید خاصیت Copy to local به false تنظیم شود، در غیر اینصورت افزونه‌ی شما بارگذاری نخواهد شد.



پس از افزودن ارجاعی به `HostView`، نمونه‌ای از استفاده از افزونه‌ی تولید شده به صورت زیر می‌تواند باشد که توضیحات مربوطه به صورت کامنت آورده شده است:

```
using System;
using System.AddIn.Hosting;
using CalculatorContract.HostViews;

namespace Calculator
{
    class Program
    {
        private static ICalculator _calculator;

        static void doOperation()
        {
            Console.WriteLine("1+2: {0}", _calculator.Operate("+", 1, 2));
        }

        static void Main(string[] args)
        {
            //مسیر پوشه ریشه مربوطه به خط لوله افزونه‌ها
            string path = Environment.CurrentDirectory;

            //مشخص سازی مسیر خواندن و کش کردن افزونه‌ها
            AddInStore.Update(path);

            //یافتن افزونه‌هایی سازگار با شرایط قرارداد پروژه
            //در اینجا هیچ افزونه‌ای بارگذاری نمی‌شود
            var addIns = AddInStore.FindAddIns(typeof(ICalculator), path);

            //اگر افزونه‌ای یافت شد
            if (addIns.Count > 0)
            {
                var addIn = addIns[0]; //استفاده از اولین افزونه
                Console.WriteLine("1st addIn: {0}", addIn.Name);

                //فعال سازی افزونه و همچنین مشخص سازی سطح دسترسی آن
            }
        }
    }
}
```

```

        _calculator = addIn.Activate<ICalculator>(AddInSecurityLevel.Intranet);
        // یک نمونه از استفاده آن
        doOperation();
    }

    Console.WriteLine("Press a key...");
    Console.ReadKey();
}
}
}

```

چند نکته جالب توجه در مورد قابلیت‌های ارائه شده:

- مدیریت load و unload پویا
- امکان تعریف سطح دسترسی و ویژگی‌های امنیتی اجرای یک افزونه
- امکان ایزوله سازی پروسه اجرای افزونه از هاست (در ادامه توضیح داده خواهد شد)
- مقاوم بودن پروژه به نگارش‌های مختلف قرارداد

اجرای افزونه در یک پروسه مجزا

حتما با امکانات مرورگر کروم و یا IE8 در مورد اجرای هر tab آن‌ها در یک پروسه‌ی مجزا از پروسه اصلی هاست مطلع هستید. به این صورت پروسه‌ی هاست از رفتار tab ها محافظت می‌شود، همچنین پروسه‌ی هر tab نیز از tab دیگر ایزوله خواهد بود. یک چنین قابلیتی در این فریم ورک نیز پیش بینی شده است.

```

// فعال سازی افزونه و همچنین مشخص سازی سطح دسترسی آن
// همچنین جدا سازی پروسه اجرایی افزونه از هاست
_calculator = addIn.Activate<ICalculator>(
    new AddInProcess(),
    AddInSecurityLevel.Intranet);

```

در این حالت اگر پس از فعال شدن افزونه، یک break point قرار دهیم و به task manager ویندوز مراجعه نمایم، پروسه‌ی مجزای افزونه قابل مشاهده است.

