عنوان: **آشنایی با M.A.F - قسمت اول**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۳۰ ۲۱:۳۸:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: MAF

در طی چند مقاله قصد بررسی نحوه‌ی تولید برنامه‌های توسعه پذیر (extensible) را با استفاده از plug-ins و یا add-ins داریم.

افزونه‌ها عموما در سه گروه قرار می‌گیرند:
الف) افزونه، سرویسی را به هاست ارائه می‌دهد. برای مثال یک میل سرور نیاز به افزونه‌هایی برای ویروس یابی یا فیلتر کردن هرزنامه‌ها دارد؛ یا یک برنامه پردازش متنی نیاز به افزونه‌ای جهت بررسی غلط‌های املایی می‌تواند داشته باشد و یا یک مرورگر وب می‌تواند با کمک افزونه‌ها قابلیت‌های پیش فرض خود را به شدت توسعه و افزایش دهد (نمونه‌ی بارز آن فایرکس است که عمده‌ترین دلیل اقبال عمومی به آن سهولت توسعه پذیری آن می‌باشد).
ب) در گروه دوم، هاست، رفتار مشخصی را ارائه داده و سپس افزونه بر اساس آن، نحوه‌ی عملکرد هاست را مشخص می‌کند. در این حالت هاست است که سرویسی را به افزونه ارائه می‌دهد. نمونه‌ی بازر آن افزونه‌های آفیس هستند که امکان اتوماسیون فرآیندهای مختلف آن‌را میسر می‌سازند. به این صورت امکان توسعه‌ی یک برنامه به شکلی که در طراحی اولیه آن اصلا انتظار آن نمی‌رفته وجود خواهد داشت. همچنین در اینجا نیازی به داشتن سورس کد برنامه‌ی اصلی نیز نمی‌باشد.
ج) گروه سوم افزونه‌ها تنها از هاست جهت نمایش خود استفاده کرده و عملا استفاده‌ی خاصی از هاست ندارد. برای مثال نوار ابزاری که خود را به windows explorer متصل می‌کند و تنها از آن جهت نمایش خود بهره می‌جوید.

در حال حاضر حداقل دو فریم ورک عمده جهت انجام این‌کار و تولید افزونه‌ها برای دات نت فریم ورک مهیا است:
الف) managed addin framework یا MAF
ب) managed extensibility framework یا MEF

فضای نام جدیدی به دات نت فریم ورک سه و نیم به نام System.AddIn اضافه شده است که به آن Managed AddIn Framework یا MAF نیز اطلاق می‌شود. از این فریم ورک در VSTO (تولید افزونه برای مجموعه‌ی آفیس) توسط خود مایکروسافت استفاده شده است.

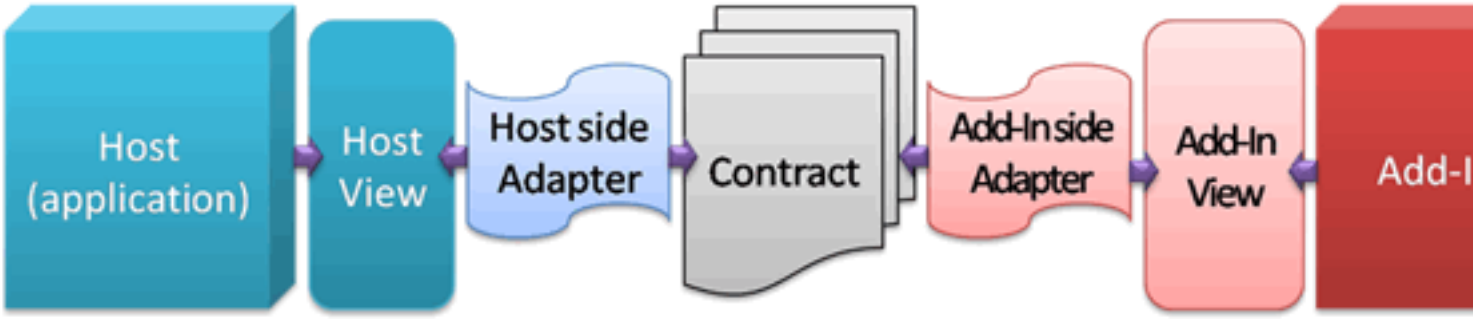
فریم ورک توسعه‌ی افزونه‌های مدیریت شده در دات نت فریم ورک سه و نیم، مزایای زیر را در اختیار ما خواهد گذاشت:
- امکانات load و unload افزونه‌های تولید شده
- امکان تغییر افزونه‌ها در زمان اجرای برنامه اصلی بدون نیاز به بستن آن
- ارائه‌ی محیطی ایزوله با ترسیم مرزی بین افزونه و برنامه اصلی
- مدیریت طول عمر افزونه
- مدیریت سازگاری با نگارش‌های قبلی و یا بعدی یک افزونه
- امکانات به اشتراک گذاری افزونه‌ها با برنامه‌های دیگر
- تنظیمات امنیتی و مشخص سازی سطح دسترسی افزونه‌ها
و ...

یک راه حل مبتنی بر MAF می‌تواند شامل 7 پروژه باشد (که به روابط تعریف شده در آن pipeline هم گفته می‌شود):

Host : همان برنامه‌ی اصلی است که توسط یک سری افزونه، توسعه یافته است.
Host View : بیانگر انتظارات هاست از افزونه‌ها است. به عبارت دیگر افزونه‌ها باید موارد لیست شده در این پروژه را پیاده سازی کنند.
Host Side Adapter : پل ارتباطی Host View و پروژه‌ی Contract است.
Contract: اینترفیسی است که کار برقراری ارتباط بین Host و افزونه‌ها را برعهده دارد.

Add-In Side Adapter : پل ارتباطی بین Add-In View و Contract است.
Add-In View : حاوی متدها و اشیایی است که جهت برقراری ارتباط با هاست از آن‌ها استفاده می‌شود.
Add-In : اسمبلی است که توسط هاست جهت توسعه‌ی قابلیت‌های خود بارگذاری می‌شود (به آن Plug-In ، Extension ، Add-On و Snap-In هم گفته می‌شود).

هدف از این جدا سازی‌ها ارائه‌ی راه حل loosely-coupledایی است که امکان ایزوله سازی، اعمال شرایط امنیتی ویژه و همچنین کنترل نگارش‌های مختلف را تسهیل می‌بخشد و این امر با استفاده از interface های معرفی شده میسر گردیده است. این pipeline از قسمت‌های ذیل تشکیل می‌شود:



قرار داد یا Contract
برای تولید یک افزونه نیاز است تا بین هاست و افزونه قراردادی بسته شود. با توجه به استفاده از MAF ، روش تعریف این قرار داد برای مثال در یک افزونه‌ی مترجم به صورت زیر باید باشد:

```
[AddInContract]
public interface ITranslator : IContract
{
   string Translate(string input);
}
```

استفاده از ویژگی AddInContract و پیاده سازی اینترفیس IContract جزو مراحل کاری استفاده از MAF است. MAF هنگام تولید پویای pipeline ذکر شده به دنبال ویژگی AddInContract می‌گردد. این موارد در فضای نام System.AddIn.Pipeline تعریف شده‌اند.

دیدگاه‌ها یا Views
دیدگاه‌ها کدهایی هستند که کار تعامل مستقیم بین افزونه و هاست را بر عهده دارند. هاست یا افزونه هر کدام می‌توانند دیدگاه خود را نسبت به قرار داد بسته شده داشته باشند. این موارد نیز همانند قرار داد در اسمبلی‌های مجزایی نگهداری می‌شوند.

دیدگاه هاست نسبت به قرار داد:

```
public abstract class TranslatorHostView
{
   public abstract string Translate(string input);
}
```

دیدگاه افزونه نسبت به قرار داد:

```
[AddInBase]
public abstract class TranslatorHostView
{
   public abstract string Translate(string input);
}
```

هر دو کلاس فوق بر اساس قرار موجود بنا می‌شوند اما وابسته به آن نیستند. به همین جهت به صورت کلاس‌هایی abstract تعریف شده‌اند. در سمت افزونه، کلاس تعریف شده دیدگاه آن با کلاس دیدگاه سمت هاست تقریبا یکسان می‌باشد؛ اما با ویژگی AddInBase تعریف شده در فضای نام System.AddIn.Pipeline مزین گردیده است.

وفق دهنده‌ها یا Adapters

آخرین قسمت pipeline ، وفق دهنده‌ها هستند که کار آن‌ها اتصال قرار داد به دیدگاه‌ها است و توسط آن مدیریت طول عمر افزونه و همچنین تبدیل اطلاعات بین قسمت‌های مختلف انجام می‌شود. شاید در نگاه اول وجود آن‌ها زائد به نظر برسد اما این جدا سازی کدها سبب تولید افزونه‌هایی خواهد شد که به نگارش هاست و برنامه اصلی وابسته نبوده و بر عکس (version tolerance). به دو کلاس زیر دقت نمائید:

کلاس زیر با ویژگی [HostAdapter] تعریف شده در فضای نام System.AddIn.Pipeline، مزین شده است و کار آن اتصال HostView به Contract می‌باشد. برای این منظور TranslatorHostView ایی را که پیشتر معرفی کردیم باید پیاده سازی نماید. علاوه بر این با ایجاد وهله‌ای از کلاس ContractHandle ، کار مدیریت طول عمر افزونه را نیز می‌توان انجام داد.

```
[HostAdapter]
public class TranslatorHostViewToContract : TranslatorHostView
{
   ITranslator _contract;
   ContractHandle _lifetime;

   public TranslatorHostViewToContract(ITranslator contract)
   {
       _contract = contract;
       _lifetime = new ContractHandle(contract);
   }

   public override string Translate (string inp)
   {
       return _contract.Translate(inp);
   }
}
```

کلاس سمت افزونه نیز بسیار شبیه قسمت قبل است و کار آن اتصال AddInView به Contract می‌باشد که با پیاده سازی ContractBase و Itranslator صورت خواهد گرفت. همچنین این کلاس به ویژگی AddInAdapter مزین گردیده است.

```
[AddInAdapter]
public class TranslatorAddInViewToContract : ContractBase, ITranslator
{
   TranslatorAddInView _view;

   public TranslatorAddInViewToContract(TranslatorView view)
   {
       _view = view;
   }

   public string Translate(string inp)
   {
       return _view.Translate(inp);
   }
}
```

قسمت عمده‌ای از این کدها تکراری است. جهت سهولت تولید این کلاس‌ها و پروژه‌های مرتبط، تیم مربوطه برنامه‌ای را به نام

pipeline builder ارائه داده است که از آدرس زیر قابل دریافت است:

http://www.codeplex.com/clraddins

این برنامه با دریافت اسمبلی مربوط بهcontract ، کار ساخت خودکار کلاس‌های adapters و views را انجام خواهد داد.

ایجاد افزونه

پس از ساخت قسمت‌های مختلف pipeline ، اکنون می‌توان افزونه را ایجاد نمود. هر افزونه باید add-in view را پیاده سازی کرده و با ویژگی AddIn مزین شود. برای مثال:

```
[AddIn("GoogleTranslator", Description="Universal translator",
Version="1.0.0.0", Publisher="YourName")]
public class GoogleAddIn : TranslatorAddInView
{
   public string Translate(string input)
   {
       ...
   }
}
```

ادامه دارد ....