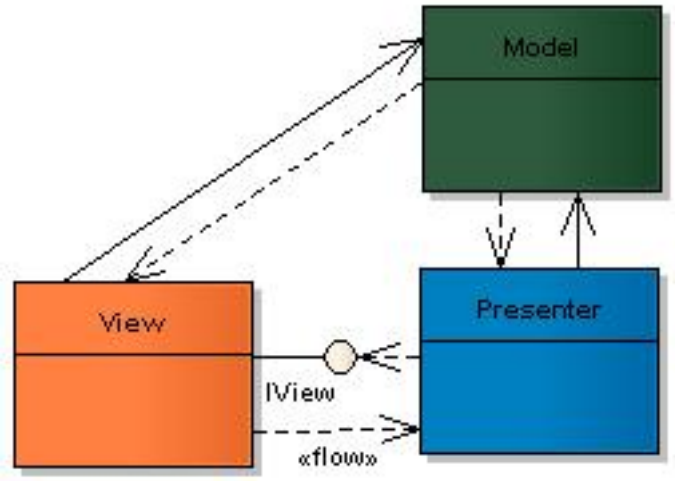
عنوان: **آشنایی با الگوی MVP**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۲۸ ۱۲:۴۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Design patterns

پروژه‌های زیادی را می‌توان یافت که اگر سورس کدهای آن‌ها را بررسی کنیم، یک اسپاگتی کد تمام عیار را در آن‌ها می‌توان مشاهده نمود. منطق برنامه، قسمت دسترسی به داده‌ها، کار با رابط کاربر، غیره و غیره همگی درون کدهای یک یا چند فرم خلاصه شده‌اند و آنچنان به هم گره خورده‌اند که هر گونه تغییر یا اعمال درخواست‌های جدید کاربران، سبب از کار افتادن قسمت دیگری از برنامه می‌شود.
همچنین از کدهای حاصل در یک پروژه، در پروژه‌های دیگر نیز نمی‌توان استفاده کرد (به دلیل همین در هم تنیده بودن قسمت‌های مختلف). حداقل نتیجه یک پروژه برای برنامه نویس، باید یک یا چند کلاس باشد که بتوان از آن به عنوان ابزار تسریع انجام پروژه‌های دیگر استفاده کرد. اما در یک اسپاگتی کد، باید مدتی طولانی را صرف کرد تا بتوان یک متد را از لابلای قسمت‌های مرتبط و گره خورده با رابط کاربر استخراج و در پروژه‌ای دیگر استفاده نمود. برای نمونه آیا می‌توان این کدها را از یک برنامه ویندوزی استخراج کرد و آن‌ها را در یک برنامه تحت وب استفاده نمود؟

یکی از الگوهایی که شیوه‌ی صحیح این جدا سازی را ترویج می‌کند، الگوی MVP یا Model-View-Presenter می‌باشد. خلاصه‌ی این الگو به صورت زیر است:



Model :
من می‌دانم که چگونه اشیاء برنامه را جهت حصول منطقی خاص، پردازش کنم.
من نمی‌دانم که چگونه باید اطلاعاتی را به شکلی بصری به کاربر ارائه داد یا چگونه باید به رخدادها یا اعمال صادر شده از طرف کاربر پاسخ داد.

View :
من می‌دانم که چگونه باید اطلاعاتی را به کاربر به شکلی بصری ارائه داد.
من می‌دانم که چگونه باید اعمالی مانند data binding و امثال آن را انجام داد.
من نمی‌دانم که چگونه باید منطق پردازشی موارد ذکر شده را فراهم آورم.

Presenter :

من می‌دانم که چگونه باید درخواست‌های رسیده کاربر به View را دریافت کرده و آن‌ها را به Model انتقال دهم.
من می‌دانم که چگونه باید اطلاعات را به Model ارسال کرده و سپس نتیجه‌ی پردازش آن‌ها را جهت نمایش در اختیار View قرار دهم.
من نمی‌دانم که چگونه باید اطلاعاتی را ترسیم کرد (مشکل View است نه من) و نمی‌دانم که چگونه باید پردازشی را بر روی اطلاعات انجام دهم. (مشکل Model است و اصلا ربطی به اینجانب ندارد!)

یک مثال ساده از پیاده سازی این روش
برنامه‌ای وبی را بنویسید که پس از دریافت شعاع یک دایره از کاربر، مساحت آن‌را محاسبه کرده و نمایش دهد.
یک تکست باکس در صفحه قرار خواهیم داد (txtRadius) و یک دکمه جهت دریافت درخواست کاربر برای نمایش نتیجه حاصل در یک برچسب به نام lblResult

الف) پیاده سازی به روش متداول (اسپاگتی کد)

```
protected void btnGetData_Click(object sender, EventArgs e)
      {
          lblResult.Text = (Math.PI * double.Parse(txtRadius.Text) *
double.Parse(txtRadius.Text)).ToString();
      }
```

بله! کار می‌کنه!
اما این مشکلات را هم دارد:
- منطق برنامه (روش محاسبه مساحت دایره) با رابط کاربر گره خورده.
- کدهای برنامه در پروژه‌ی دیگری قابل استفاده نیست. (شما متد یا کلاسی را این‌جا با قابلیت استفاده مجدد می‌توانید پیدا می‌کنید؟ آیا یکی از اهداف برنامه نویسی شیءگرا تولید کدهایی با قابلیت استفاده مجدد نبود؟)
- چگونه باید برای آن آزمون واحد نوشت؟

ب) بهبود کد و جدا سازی لایه‌ها از یکدیگر

در روش MVP متداول است که به ازای هر یک از اجزاء ابتدا یک interface نوشته شود و سپس این اینترفیس‌ها پیاده سازی گردد.

پیاده سازی منطق برنامه:

1- ایجاد Model :
یک فایل جدید را به نام CModel.cs به پروژه اضافه کرده و کد زیر را به آن خواهیم افزود:

```
using System;

namespace MVPTest
{
  public interface ICircleModel
  {
      double GetArea(double radius);
  }

  public class CModel : ICircleModel
  {
      public double GetArea(double radius)
      {
          return Math.PI * radius * radius;
      }
  }
}
```

همانطور که ملاحظه می‌کنید اکنون منطق برنامه از موارد زیر اطلاعی ندارد:

- خبری از textbox و برچسب و غیره نیست. اصلا نمی‌داند که رابط کاربری وجود دارد یا نه.
- خبری از رخدادهای برنامه و پاسخ دادن به آن‌ها نیست.
- از این کد می‌توان مستقیما و بدون هیچ تغییری در برنامه‌های دیگر هم استفاده کرد.
- اگر باگی در این قسمت وجود دارد، تنها این کلاس است که باید تغییر کند و بلافاصله کل برنامه از این بهبود حاصل شده می‌تواند بدون هیچگونه تغییری و یا به هم ریختگی استفاده کند.
- نوشتن آزمون واحد برای این کلاس که هیچگونه وابستگی به UI ندارد ساده است.

-2 ایجاد View :
فایل دیگری را به نام CView.cs را به همراه اینترفیس زیر به پروژه اضافه می‌کنیم:

```
namespace MVPTest
{
  public interface IView
  {
      string RadiusText { get; set; }
      string ResultText { get; set; }
  }
}
```

کار View دریافت ابتدایی مقادیر از کاربر توسط RadiusText و نمایش نهایی نتیجه توسط ResultText است البته با یک اما.
View نمی‌داند که چگونه باید این پردازش صورت گیرد. حتی نمی‌داند که چگونه باید این مقادیر را به Model جهت پردازش برساند یا چگونه آن‌ها را دریافت کند (به همین جهت از اینترفیس برای تعریف آن استفاده شده).

-3 ایجاد Presenter :
در ادامه فایل جدیدی را به نام CPresenter.cs با محتویات زیر به پروژه خواهیم افزود:

```
namespace MVPTest
{
  public class CPresenter
  {
      IView _view;

      public CPresenter(IView view)
      {
          _view = view;
      }

      public void CalculateCircleArea()
      {
          CModel model = new CModel();
          _view.ResultText = model.GetArea(double.Parse(_view.RadiusText)).ToString();
      }
  }
}
```

کار این کلاس برقراری ارتباط با Model است.
می‌داند که چگونه اطلاعات را به Model ارسال کند (از طریق _view.RadiusText) و می‌داند که چگونه نتیجه‌ی پردازش را در اختیار View قرار دهد. (با انتساب آن به _view.ResultText)
نمی‌داند که چگونه باید این پردازش صورت گیرد (کار مدل است نه او). نمی‌داند که نتیجه‌ی نهایی را چگونه نمایش دهد (کار View است نه او).
روش معرفی View به این کلاس به constructor dependency injection معروف است.

اکنون کد وب فرم ما که در قسمت (الف) معرفی شده به صورت زیر تغییر می‌کند:

```
using System;
```

```
namespace MVPTest
{
  public partial class _Default : System.Web.UI.Page, IView
  {
      protected void Page_Load(object sender, EventArgs e)
      {
      }

      public string RadiusText
      {
          get { return txtRadius.Text; }
          set { txtRadius.Text = value; }
      }
      public string ResultText
      {
          get { return lblResult.Text; }
          set { lblResult.Text = value; }
      }

      protected void btnGetData_Click(object sender, EventArgs e)
      {
          CPresenter presenter = new CPresenter(this);
          presenter.CalculateCircleArea();
      }
  }
}
```

در این‌جا یک وهله از Presenter برای برقراری ارتباط با Model ایجاد می‌شود. همچنین کلاس وب فرم ما اینترفیس View را نیز پیاده سازی خواهد کرد.

نظرات خوانندگان

نویسنده: زوزو
تاریخ: ۱۳۸۸/۰۵/۲۸ ۱۷:۵۶:۰۸

آیا این همون مدل 3 لایه برنامه نویسی هست که در اینجا بدون استفاده از دیتابیس به این شکل مطرح شده است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۲۸ ۱۸:۳۰:۲۱

سلام
احتمالا tier 3 را شنیده‌اید که به این صورت مطرح کردید.
n-tier نوعی معماری است که به شما در تهیه برنامه‌های توزیع شده کمک می‌کند و مهم‌ترین مزیت آن قابلیت بسط پذیری سیستم است. Tiering در مورد تخصیص منابع و نحوه‌ی توزیع آن‌ها بحث می‌کند. برای مثال دیتابیس سرور شما جدا است، منطق برنامه در سروری دیگر توسط یک وب سرویس قابل دسترسی است و سروری دیگر کار دریافت و ارائه این اطلاعات را به عهده خواهد داشت.
MVC که در ابتدا پدید آمد و بعد از آن MVP ، یک نوع الگوی برنامه نویسی شیءگرا هستند که به شما کمک خواهند کرد تا برنامه‌ی n-tier ایی با حداقل گره خوردگی و به هم پیچیدگی که اصطلاحا به آن Loosely coupled نیز گفته می‌شود، تولید کنید.

نویسنده: میثم جوادی
تاریخ: ۱۳۸۸/۰۵/۲۹ ۰۱:۵۲:۵۳

سلام، جناب نصیری اگه برنامه بزرگ بشه نیاز به الگوی Facade بیشتر نمیشه؟منظورم استفاده اش تو این الگو.
یه ازای همه کلاس ها باید اینترفیس تعریف کنیم؟
(حتی واسه گوگل کلمه MVP تازه است به طوری که did you mean... گوگل کلمه MVC رو پیشنهاد میکنه!)

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۲۹ ۱۰:۲۵:۲۶

سلام
MVP جدید نیست و اولین مقاله در مورد آن به سال 1996 بر می‌گرده
http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

- در این روش Presenter شما می‌تونه با یک Facade یا Service object جهت دریافت اطلاعات Model نیز در ارتباط باشه.

نویسنده: حسن
تاریخ: ۱۳۸۸/۰۵/۳۰ ۰۲:۴۸:۱۱

از کدهایی که نوشته بودید هیچ سر در نیاوردم (.net بود؟) ولی با MVC در phpآشنام، اینا فرق دارن یا فقط اسماشون در محیط‌های مختلف متفاوته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۳۰ ۱۱:۳۴:۵۵

فرق دارند (و البته در انحصار پلتفرم خاصی هم نیستند چون یک نوع الگوی برنامه نویسی‌اند).
سیر تکاملی این‌ها رو در تصویر زیر می‌تونید مشاهده کنید:
http://vahid.nasiri.googlepages.com/mvcmvp.png

تفاوت‌ها:

در MVP
view و model کاملا از هم جدا شده‌اند.
presenter کار رخدادگردانی عناصر UI را انجام می‌دهد
presenter کار به روز رسانی view را از طریق فراخوانی اینترفیس آن انجام می‌دهد

در MVC
view و model کاملا از هم جدا نیستند.
View کار رخدادگردانی عناصر UI را انجام می‌دهد
controller مدل را به view ارسال کرده و سپس view بر این اساس خودش را به روز می‌کند

نویسنده: مسعود
تاریخ: ۱۳۸۸/۰۵/۳۰ ۱۳:۰۴:۴۳

آقای نصیری اول که خسته نباشید.

دوم اینکه من اینو درک نمی کنم که وقتی من اینتر فیس برای بخشی نوشتم و اونوimplements کردم و از این نوع معاری استفاده کردم ، چرا بهتره.

من سوالم اینجاست که اگه من اینترفیس رو متناسب با نیازی که بعدا به وجود میاد تغییراتی بدم کم ترین کاری که باید بکنم اینه که کلاس مربوطه رو دست کاری کنم.
این که زمان بیشتری میگیره...

البته منکر reusable شدن و ... این مدل نیستم...

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۳۰ ۱۴:۱۸:۱۳

کیوان نیری یک دمو در مورد ASP.Net MVC درست کرده و روش متداول و روش جدید را در ابتدای این دمو با هم مقایسه کرده (حداقل برای MVC الان یک فریم ورک خوب هست).
http://nayyeri.net/files/media/file/Talks/ASPNETMVC10Presentation.pptx

مزایا و معایب هر کدام را توضیح داده که بد نیست یک نگاهی بیندازید.

نویسنده: افشار محبی
تاریخ: ۱۳۸۸/۰۵/۳۰ ۱۶:۵۴:۳۶

MVP به نوعی با SOA (معماری سرویس‌گرا) هم شبیه است

نویسنده: ahmad
تاریخ: ۱۳۸۹/۰۴/۱۸ ۰۰:۰۲:۳۳

سلام
بالاخره در Windows Application از کدام روش استفاده کنیم؟ MVP یا n-tier ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۴/۱۸ ۰۱:۱۰:۳۶

شما در یک سیستم n-tier می‌تونید از MVP هم استفاده کنید.

نویسنده: Mohsen
تاریخ: ۱۳۸۹/۰۶/۱۸ ۰۴:۳۵:۲۳

واقعا این چند لایه آدم رو کلافه می کنه
10 جا یک تیکه کد رو با کمی تغییر باید بنویسی

MVC رو تست کردم عالی بود

MVP رو هنوز تست نکردم
برای MVP در دات نت فریمورک وبی داریم مثل MVC

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۶/۱۸ ۱۰:۱۰:۲۸

Better Web Forms with the MVP Pattern

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۲۸ ۰۰:۳۷:۵۰

Model View Presenter Pattern Implementation in ASP.NET

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۲۸ ۰۰:۴۰:۲۳

ASP.Net MVP Framework

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۲۸ ۰۱:۰۷:۳۵

Web Forms MVP

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۰ ۱۷:۲۸:۰۹

یک مثال جالب دیگر در این مورد: (+)

نویسنده: shahin kiassat
تاریخ: ۱۳۹۰/۰۱/۱۱ ۲۳:۱۴:۰۸

سلام.
آقای نصیری هنگامی که می خواستم این آدرس رو باز کنم صحفه منتظر پاسخ از بلاگر می مونه :
http://www.dotnettips.info/2009/08/mvp.html#disqus_thread

در مورد این لینک جدید ام وی پی من برای دانلود فریمورک
webformsmvp
به این صحفه رسیدم
http://nuget.org/List/Packages/WebFormsMvp
اما با وجود ثبت نام در این سایت موفق به دانلود نمیشم.
یعنی هیچ جایی برای دانلود پیدا نکردم.
در سایت پروژه در کد پلکس هم چیزی برای دانلود نبود.
اگر فرصت داشتید راهنمایی کنید.

ممنون
پ ن : قبلا می شد کامنت راست به چپ نوشت الان انگار این امکان وجود نداره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۱ ۲۳:۵۵:۰۶

در مورد آدرس مشکلی نبود الان حالا شاید در اون لحظه مشکل ارتباطی وجود داشته

این پکیج رو برای نوگت کامپایل کردن ولی اگر علاقمند بودید می‌شود سورس را از آدرس زیر دریافت و سپس خودتون کامپایل کنید
http://webformsmvp.codeplex.com/SourceControl/list/changesets

این راست به چپ فعلا در دیسکاس نیست یا من ندیدم حالا تا بعد