

我们都是和自己赛跑的人！代码应该依赖于抽象，而不是具体的实现，只有你经历过了也能体会到...

追求：代码之美，对程序的认知：编程不仅是一种技术,更是一门艺术!
爱好：架构设计，设计模式，语言规范，代码重构，代码艺术，面向对象，面向服务，面向接口，大数据，分布式，数据库集群，读写分离，企业级架构设计
我的QQ：853066980，我的Email：bfyxzls@sina.com (C#1.0托管代码→C#2.0泛型→C#3.0LINQ→C#4.0动态语言→C#5.0异步编程)
我的成就：自主开发EF架构,LinqToSql架构,DDD架构,MVVM架构,同时也是Knockout,AngularJS,Bootstrap的追求者。
我的产品：仓储大叔框架，大叔框架群437541737，数据库集群，NoSql集群，分布式数据集缓存，分布式Session存储，分布式文件存储，消息中间件，通知策略。



姓名：张占岭
大叔框架群：437541737
大叔商城在线Demo演示
信仰：没有人能随随便便成功
追求：代码之美，深入体会面向对象
梦想：拥有自己的软件公司, 提供SaaS
回味：代码的味道...
洋名：Lind
花名：仓储大叔
职位：系统架构师/分析师，高级讲师
《大叔Lind. DDD敏捷框架`New》
《大叔企业级框架介绍》
《大叔讲自己的框架》
工作经历：
8年B2C/C2C系统架构与开发经验
2年OA, CMS系统架构与开发经验
2年asp., net, sql教学经验
关注大叔微信公众号
更多架构资源等着你



技术资料
infoq
cnblogs
csdn
codeplex
github
msdn
stackoverflow
博客统计
昵称：张占岭
园龄：5年11个月
粉丝：1926
关注：32
+加关注

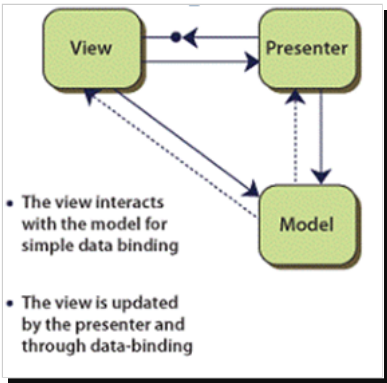
随笔分类
.NET MVC (182)
.NET Remoting (1)
AOP (14)

MVVM架构~mvc,mvp,mvvm大话开篇

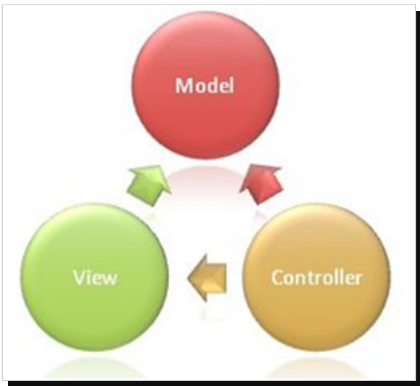
[返回目录](#)

百度百科的定义：

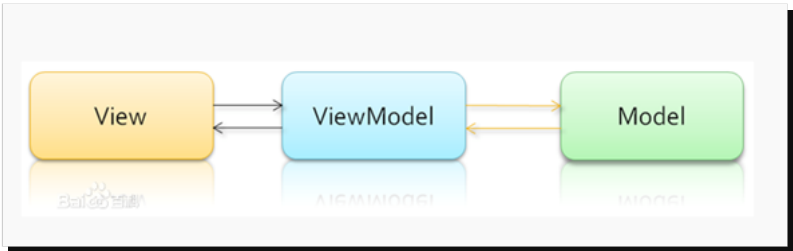
MVP 是从经典的模式MVC演变而来，它们的基本思想有相通的地方：Controller/Presenter负责逻辑的处理，Model提供负责显示。作为一种新的模式，MVP与MVC有着一个重大的区别：在MVP中View并不直接使用Model，它们之间的通信是通过(MVC中的Controller)来进行的，所有的交互都发生在Presenter内部，而在MVC中View会从直接Model中读取数据而不是Controller。



在MVC里，View是可以直接访问Model的！从而，View里会包含Model信息，不可避免的还要包括一些 业务逻辑。在MV注的Model的不变，而同时有多个对Model的不同显示，及View。所以，在MVC模型里，Model不依赖于View，但是 View Model的。不仅如此，因为有一些业务逻辑在View里实现了，导致要更改View也是比较困难的，至少那些业务逻辑是无法



MVVM在概念上是真正将页面与数据逻辑分离的模式，在开发方式上，它是真正将前台代码开发者（JS+HTML）与后台代模式（asp,asp.net,php,jsp）。



ASP.NET (136)
C# (227)
DDD(领域驱动设计) (26)
Entity Framework (72)
EXCEL VBA (1)
IoC(控制反转) (13)
JAVASCRIPT/jQuery/HTML (120)
LINQ (49)
MSMQ (2)
MVVM (26)
Nginx (3)
NodeJS (10)
NoSQL (42)
Objective-C (IOS) (1)
Orchard (2)
Thrift (5)
WCF (2)
Web Api & CORS (13)
WebService (2)
WebSocket (2)
WinForm (1)
Xamarin (11)
不说技术 (21)
大牛们说过的话，我们应该去参考 (2)
代码重构 (8)
分布式文件存储 (DFS) (4)
开发工具 (18)
流媒体
面向对象 (77)
面向服务 (7)
全文检索 (4)
设计模式 (17)
数据结构 (15)
数据库 (43)
网络编程 (4)
系统架构 (203)
压力测试 (2)
异步和并行 (9)
异常错误与解决 (5)
积分与排名
积分 - 931089
排名 - 58
最新评论
1. Re: 压力测试`测试工具的使用
迷失的疯丫头特定数量的用户，在某
个时间段，向一个固定的URL，
发送请求！...
——张占岭
2. Re: 实时监控Cat之旅`介绍与自定义
类型在哪里
是呀，乱世出英雄，现在的互联网时代
就是这样，一个成功的项目比一个
成功的公司更能培养出人才来！
——张占岭
3. Re: 压力测试`测试工具的使用
压力测试是种什么样的测试呀，想了解
下这个
——迷失的疯丫头
4. Re: WebApi系列`通过HttpClient来
调用Web Api接口`续`实体参数的传
递

MVP与传统MVC的区别，但目前我们提倡的MVC已经走向MVP了

在MVP里，Presenter完全把Model和View进行了分离，主要的程序逻辑在Presenter里实现。而且，Presenter与具体的V关联的，而是通过定义好的接口进行交互，从而使得在变更View时候可以保持Presenter的不变，即重用！

不仅如此，我们还可以编写测试用的View，模拟用户的各种操作，从而实现对Presenter的测试--而不需要使用自动化的测试。至可以在Model和View都没有完成时候，就可以通过编写Mock Object（即实现了Model和View的接口，但没有具体的内部Presenter的逻辑。

在MVP里，应用程序的逻辑主要在Presenter来实现，其中的View是很薄的一层。因此就有人提出了Presenter First的设计。用User Story来首先设计和开发Presenter。在这个过程中，View是很简单的，能够把信息显示清楚就可以了。在后面，根据Model和View，而对Presenter没有任何的影响了。如果要实现的UI比较复杂，而且相关的显示逻辑还跟Model有关系，就可以在View和Presenter之间放置一个Adapter。由这个Adapter来访问Model和View，避免两者之间的关联。而同时，因为Adapter实现了Model和View的接口，从而可以保证与Presenter之间接口的不变。这样就可以保证View和Presenter之间接口的简洁，又不失去UI的灵活性。里，View只应该有简单的Set/Get的方法，用户输入和设置界面显示的内容，除此就不应该有更多的内容，绝不容许直接访问Model，这是与MVC很大的不同之处。

目前我们提倡的MVC已经与MVP没有太大区别，View依然是很薄的一层，不进行与Model的逻辑处理，只进行简单的页面渲染。

MVVM是Model-View-ViewModel的简写。

微软的WPF带来了新的技术体验，如Sliverlight、音频、视频、3D、动画.....，这导致了软件UI层更加细节化、可定制化。在WPF（原来winform那边的）也带来了诸如Binding、Dependency Property、Routed Events、Command、DataTemplate等新特性。MVVM（Model-View-ViewModel）框架的由来便是MVP（Model-View-Presenter）模式应用方式时发展演变过来的一种新型架构框架。它立足于原有MVP框架并且把WPF的新特性揉合进去，以应对客户日益复杂的需求。

MVC中引入MVVM的意义

前端工程师们与程序开发人员们实现了真正的分离，当然，为了减少重复代码量，前端人员有必要了解一个数据库结构。

前端进行MVVM的JS插件，knockoutjs

Knockoutjs可以灵活的实现数据的绑定工作，对于前台开发人员来说，不需要关心后台数据以何种方式产生，它们只需要关心数据即可，对于程序开发人员来说，它们关心的是数据产生的接口，包括简单的CURD操作接口，程序开发人员完成可以把这些接口封装成JS文件中，由HTML页面指定引用，真实数据即可完成绑定工作。

使用knockoutjs来实现MVC中的MVVM模式

html代码：

```
<table>
  <thead>
    <tr>
      <th>编号</th>
      <th>姓名</th>
      <th>电话</th>
      <th>操作</th>
    </tr>
  </thead>
  <tbody data-bind="template:{name:'list',foreach: lines}">
  </tbody>
  <tfoot>
    <tr>
      <td colspan="4">
        <button data-bind='click: addLine'>Add</button>
      </td>
    </tr>
  </tfoot>
</table>
<script type="text/html" id="list">
  <tr>
    <td><span data-bind="text:Id"></span></td>
    <td>
      <span data-bind="text: Name"></span>
    </td>
    <td>
    </td>
    <td>
      <span data-bind="text: PhoneNo"></span>
    </td>
  </tr>
</script>
```

您好 我有个问题关于webclient 请求 web services 方法的问题我的 webservices 的WebMethod 需要一个 实体对象参数 我如何用 web client post方式.....

——逗比大琳琳

5. Re:实时监控Cat之旅`介绍与自定义类型在哪里

看来点评也培养很多技术精英嘛 我们公司现在在用点评的CAT ,Lion, zebra, pigeon.... 确实很好用...

——回车键

- 阅读排行榜
- 1. NodeJS系列`第四个小例子, NodeJs 处理Get请求和Post请求 (22225)
 - 2. WebApi系列`基于RESTful标准的Web Api (19991)
 - 3. WebApi系列`通过HttpClient来调用 Web Api接口 (18422)
 - 4. SignalR实现服务器与客户端的实时通信 (17512)
 - 5. NodeJS系列`第三个小例子, NodeJs 与Redis实现高并发的队列存储 (17143)

- 评论排行榜
- 1. SignalR实现服务器与客户端的实时通信 (30)
 - 2. 面向对象的故事`数据底层操作告诉了我们接口, 抽象类, 继承与多态性的使用 (23)
 - 3. IoC`MVC3+EF+Autofac实现松耦合的系统架构 (21)
 - 4. 产品介绍`大叔管理系统 (21)
 - 5. EF架构`一个规范, 两个实现 (16)

```
<a href='#' data-bind='click: $parent.removeLine'>Remove</a>
</td>
</tr>
</script>
```

controller层C#代码得到数据：

```
public IEnumerable<Contact> Get()
{
    Contact[] contacts = new Contact[]
    {
        new Contact{Id=1, Name="张三", PhoneNo="123", EmailAddress="zhangsan@gmail.com"},
        new Contact{Id=2, Name="李四", PhoneNo="456", EmailAddress="lisi@gmail.com"},
        new Contact{Id=3, Name="王五", PhoneNo="789", EmailAddress="wangwu@gmail.com"},
    };
    return contacts;
}
```

knockoutjs代码用来绑定数据和实现页面的交互

```
<script type="text/javascript">
    var CartLine = function () {
        var self = this;
        self.Id = ko.observable();
        self.Name = ko.observable();
        self.PhoneNo = ko.observable(135);
        self.EmailAddress = ko.observable("@");
    };

    $.getJSON("http://localhost:2166/api/values/", function (data) {
        var Cart = function () {
            // Stores an array of lines, and from these, can work out the grandTotal
            var self = this;
            //self.lines = ko.observableArray([new CartLine()]);
            self.lines = ko.observableArray(data);
            // Operations
            self.addLine = function () { self.lines.push(new CartLine()); };
            self.removeLine = function (line) { self.lines.remove(line); };
        }
        ko.applyBindings(new Cart());
    });
```

OK，到了这里，一个MVVM风格的实例已经讲解完了，下回我们将逐步来学习knockoutjs。

[返回目录](#)

分类: [MVVM](#)

好文要顶

关注我

收藏该文

张占岭
关注 - 32
粉丝 - 1926
[+加关注](#)

« 上一篇: [JS~delegate与live](#)
» 下一篇: [MVVM架构~Knockoutjs系列之text,value,attr,visible,with的数据绑定](#)

评论列表

#1楼 2015-08-15 17:36 飞鹰008

好文章，赞一个！

支

#2楼 2015-11-11 22:47 Lumia1020

这篇文章好 因为在 View 中可以直接调用方法之类的
所以在 MVC 导致 MVC 中 V 与 M 又形成了聚合

支

[刷新评论](#) [刷新](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。