[mvc与三层结构终极区别](http://blog.csdn.net/csh624366188/article/details/7183872)

http://blog.csdn.net/csh624366188/article/details/7183872

标签： [mvc](http://www.csdn.net/tag/mvc" \t "_blank)[asp.net](http://www.csdn.net/tag/asp.net)[webform](http://www.csdn.net/tag/webform)[设计模式](http://www.csdn.net/tag/%e8%ae%be%e8%ae%a1%e6%a8%a1%e5%bc%8f)[servlet](http://www.csdn.net/tag/servlet)[游戏](http://www.csdn.net/tag/%e6%b8%b8%e6%88%8f)

2012-01-07 17:18 67894人阅读 [评论](http://blog.csdn.net/csh624366188/article/details/7183872#comments)(27) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/csh624366188/article/details/7183872#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

设计模式（11） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

**注：本文章内所有内容都来自互联网，本人主要是起了一个收集的作用**

[**http://www.cnblogs.com/zhhh/archive/2011/06/10/2077519.html**](http://www.cnblogs.com/zhhh/archive/2011/06/10/2077519.html)

**又看到有人在问三层[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "_blank)和MVC的关系，感觉这种问题有点教条化了。因为它们都在逻辑上将应用程序划为三块，凑了一个数字3，就有人非要把它们联系到一起了。**

　　这两个东西我接触有几年了，有一点体会，表达一下：

　　三层是三层，MVC是MVC，它们毫无关系的。

**三层**是从整个应用程序架构的角度来分的三层（如果程序需要，还可以分多层）。

　　三层是为了解决整个应用程序中各个业务操作过程中不同阶段的代码封装的问题，为了使程序员更加专注的处理某阶段的业务逻辑。

　　比如将**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**操作代码封装到一层中，提供一些方法根据参数直接返回用户需要的相应数据，这样在处理具体的业务逻辑的时候，就不用关心数据的存储问题了。

**MVC**是在应用程序（BS结构）的视图层划分出来的不同功能的几个模块。

　　MVC主要是为了解决应用程序用户界面的样式替换问题，把展示数据的 HTML 页面尽可能的和业务代码分离。MVC把纯净的界面展示逻辑（用户界面）独立到一些文件中（Views），把一些和用户交互的程序逻辑（Controller）单独放在一些文件中，在 Views 和 Controller 中传递数据使用一些专门封装数据的实体对象，这些对象，统称为Models。

　　只所以说MVC和三层毫无关系，是因为它们二者使用范围不同：三层可以应用于任何语言、任何技术的应用程序；而MVC只是为了解决BS应用程序视图层各部分的耦合关系。它们互不冲突，可以同时存在，也可根据情况使用其中一种。

[**http://blog.csdn.net/beijiguangyong/article/details/7029257**](http://blog.csdn.net/beijiguangyong/article/details/7029257)

**三层架构就是MVC！**

起初老师总说三层MVC，MVC三层架构……

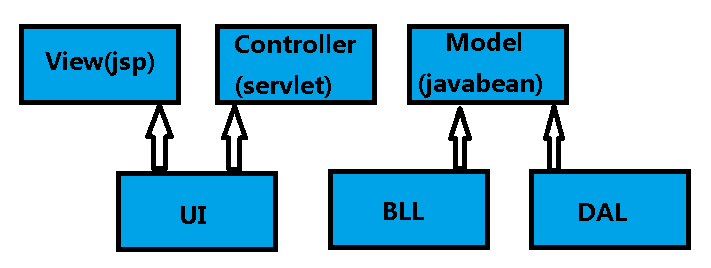
所以开始的时候脑子就一个概念：三层就是MVC，MVC就是三层架构。而且想想也合理啊，都是“三”。MVC是三个字母，三层架构也是“三”，理所应当的就对应上了。然后就这么一直“错”了很长时间。

**三层架构绝不是MVC！！**

后来学习了J2EE之后发现老师说的好像不对，MVC和三层架构不是一个东西。三层架构是界面层（UI）业务逻辑层（BLL）和数据访问层（DAL）构成的，而MVC是模型层（M）界面层（View）和控制层（Controller）构成的，而且他们之间也不对应。

如果硬要给他们对应的话，那么三层架构中的UI对应MVC中的view（jsp），都是用于显示以及获取界面的数据；三层架构中的BLL层和DAL层对应MVC中的Model（javabean）层都是用于处理上层传递来的数据以及从数据库获取的数据的；MVC中的Controller（Servlet）最多算是三层架构中的UI的一部分，也就我们常说的是Servlet。

如下图所示：



顿时感到世界明朗了，对分层又深入了解了一步。

**其实三层架构和MVC还是一个东西！！！**

这几天一直在思考三层架构和MVC到底是个什么关系，老师为什么起初会放在一起说嘞？然后恍然大悟：其实三层架构和MVC是一样的！！！我们所看到的不一样只是表面上的不一样。核心的东西是一致的，那么什么是核心？

答曰：分层，解耦！

如果从解耦的角度来看三层架构和MVC其实他们是一致的，只不过划分的方法不一样罢了，就像上面的图所示。从这一点说他们可以说是一个东西。这就相当于我们看到馒头和面条一样，表面上看他们不一样（注意仅仅是表面）但是他们核心是一致的，都是面……

知识的学习过程就要像老牛反刍一样，需要不断的加深认识，最终才能真正领悟

对事物的认识是从感性到理性的，是一步一步的加深的，每一步的加深也许会推翻以前的自己，也许会更加赞同以前的自己。如果是推翻以前的自己那么代表对这个事物的认识发生了翻天地覆的变化，但是如果赞许以前的自己也并不代表自己的观点没有变化，往往表面上看起来一致的东西其实内核并一定是相同的。就像刚开始的时候认为三层架构和MVC是一个东西到最后同样是认为这两是一个东西，但是理解的层次绝对是不一样的。

至于以后会不会再次推翻自己的观点我不晓得，只能说每次推翻都代表着进步，代表着理解的更深一层，所以我期望着下次的否定自己

<http://baike.baidu.com/view/687468.htm>

与MVC的区别　　MVC（模型Model-视图View-控制器Controller）是一种设计模式，我们可以用它来创建在域对象和UI表示层对象之间的区分。

　　同样是架构级别的，相同的地方在于他们都有一个表现层，但是他们不同的地方在于其他的两个层。

　　在三层架构中没有定义Controller的概念。这是我认为最不同的地方。而MVC也没有把业务的逻辑访问看成两个层，这是采用三层架构或MVC搭建程序最主要的区别。当然了。在三层中也提到了Model，但是三层架构中Model的概念与MVC中Model的概念是不一样的，“三层”中典型的Model层是以实体类构成的，而MVC里，则是由业务逻辑与访问数据组成的。

<http://topic.csdn.net/u/20090710/14/07f4447d-60c4-4b74-af1c-ea1b6e2e9ca7.html>

当然啊，你要明白三层架构的MVC的区别和联系：  
  
三层架构是最基本的项目分层结果，而MVC则是三层架构的一个变体，MVC是一种好的开发模式。首先你要明白MVC分别代表的是什么意思.  
M 即Model(模型层)，主要负责出来业务逻辑以及数据库的交互  
V 即View(视图层)，主要用于显示数据和提交数据  
C 即Controller(控制器)，主要是用作捕获请求并控制请求转发  
  
三层:UI 界面层 BLL 业务逻辑层，DAL数据访问层，Model 实体层  
MVC中的的M 不是三层中的Model(实体层)，他其实包括三层中的 BLL，DAL,Model,这是非常要注意的，这也是他们之间的区别的关键所在  
  
其有点有如下：  
低耦合性  
高重用性和可适用性  
较低的生命周期成本  
快速的部署  
可维护性  
有利于软件工程化管理  
  
当然优点也有缺点，那就是内部结构复杂，不容易理解，文件数量大，管理难度自然也就大

MVC设计模式…  
三层架构…  
他们细分之后得到的是：View(UI)、BIZ（BLL）、DAO（DAL）、Entity（Model）、Controller  
MVC把 BIZ（BLL）、DAO（DAL）、Model（Entity） 统一称之为 模型(MODEL)，得到：View、Controller、模型(MODEL)  
三层 在我使用中 暂未体会到控制器的存在，完全是：UI、DAO、BLL  
  
他们相同的设计理念就是：把视图设计与数据持久化进行分离，从而降低耦合性，易于扩展，提高团队开发效率。

三层是基于业务逻辑来分的，而mvc是基于页面来分的  
根本就没有什么可比性。  
其实两个一起用我感觉很好

MVC模式是一种复合设计模式，一种解决方案  
三层是种软件架构，通过接口实现编程  
三层模式是体系结构模式，MVC是设计模式  
三层模式又可归于部署模式，MVC可归于表示模式

引用楼主 daijun17 的回复:  
如何在三层架构和mvc之间进行取舍呢？

没有什么取舍的，说的根本不是一回事。在所谓的“三层”中，它要求你将BLL层独立出来，它只是告诉你表示层和业务逻辑层之间的静态关系。而MVC则告诉你在这个具体的地方如何处理其动态驱动流程，尽管mvc仍然粗糙（甚至mvp也是粗糙的），但是已经比所谓三层更细致一些了。

谢谢大家的关注，这几天都在忙面试，没来结贴。再次谢谢大家。  
我大概明白了：三层架构和mvc设计模式侧重点不一样，三层是一种笼统的架构思想，没有限制具体的设计；而mvc就比较具体的说明它的设计方法。  
还是自己动手做一下，理解会更深一些，以前都是用三层架构的方法

---------------------------------------------------------------------------------------------------------------

<http://apps.hi.baidu.com/share/detail/15057889>

**ASP**[**.NET**](http://lib.csdn.net/base/dotnet)**MVC 与 WebForm 三层架构 区别**

1.**传统WebForm开发中存在的一些问题**

传统的ASP[**.Net**](http://lib.csdn.net/base/dotnet)开发中，微软的开发团队为开发者设计了一个在可视化设计器中拖放控件，编写代码响应事件的快速开发环境。然而，它所带来的负面效应是：

由于控件封装了很多东西，开发者很难了解这背后的HTML是如何运作的

容易得到一个包含大量ViewState的页面，使得页面尺寸远远超过所需的内容，使得页面的打开速度较慢

不容易被[**测试**](http://lib.csdn.net/base/softwaretest)

2.**什么是MVC**?

MVC（Model-View-Controller，模型-视图-控制器模式）是软件工程中的一种软件架构模式。它把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。

3.什么是**ASP.NET MVC**?

ASP.NET MVC是微软的开发团队基于MVC开发的一个易于被测试的框架。它具有以下特性：

没有数据回传

没有在页面中保存视图状态

开发者可以完全掌控页面的呈现过程

易于单元测试

易于测试驱动开发

可扩展、可替换

支持WebForm中的有关特性，如：用户控件、母版页、数据绑定、本地化等

不在基于文件目录结构，而是将URL映射到控制器中

4. **为什么使用ASP.NET MVC**?

易于进行单元测试

易于进行测试驱动开发

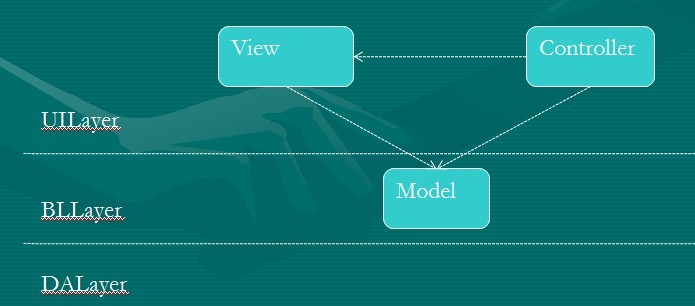
避免了WebForm中大量的ViewState导致页面文件变得臃肿

5.**MVC与三层架构**?

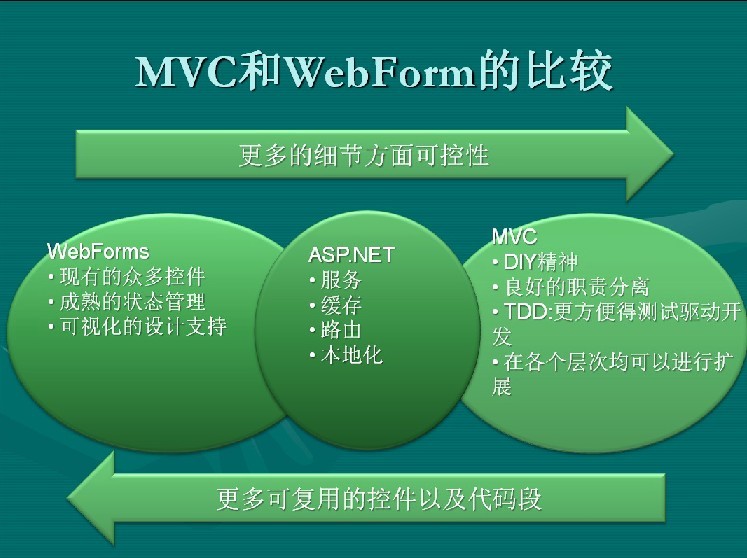
**MVC是一种模式  
ASP.NET MVC是一个基于MVC模式的开发框架  
三层架构是一种架构**

至于区别，可以严格的从概念上区分开来。

下图是MVC与三层架构的对应关系



6. MVC与Webform如何抉择?



7. 两种技术并存

ASP.NET MVC框架只是给开发者提供欧诺个了开发web应用程序的一种选择，并不是要取代Webform

这两种技术各有优缺点，开发者需要根据实际情况，选择对应的技术

有时候，可以在同一个项目中混合使用这两种技术

8.ASP.NET MVC与Webform技术的架构图



总结：

看完本文，相信ASP.NET WebForm与ASP.NET MVC的选择相信大家应该可以做到心中有数了，我始终觉得，很多时候并不是什么技术好不好的问题，而是适合不适合不适合的问题或者能否把它用好的问题。

打个比方：如果让千里马犁地，恐怕未必能达到理想的效果，最终可能还会抱怨，什么破马，一点劲都没有。

**========================================================================**

<http://www.blogjava.net/phyeas/archive/2008/07/01/211995.html>

同样是架构级别的，它们有什么相同点和不同点呢？这篇文章讨论一下它们的异同点。希望能帮助读者理解其中的玄机。 :)

其实它们相同的地方在于他们都有一个表现层。

但是他们不同的地方在于其他的两个层。

首先先解释一下MVC。V即View.是视图的意思。C即Controler.是控制器的意思。而M即Model，是模型的意思。这三个里.最不容易理解的应该是Model.就是什么是Model，而为什么叫Model。我先不说为什么叫Model，先解释Controler。

Controller是控制器的意思，所谓控制器，就是将用户请求转发给模型层，经过处理后把结果返回到界面展现的一个中间层，那么Controler到底管什么工作呢？先不说.先来看下在**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)**Web中这三个层一般的定义，一般在**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "_blank)** Web里，JSP充当V，Servlet充当C,JavaBean充当M，这里的Servlet管什么工作呢？接受输入，转到Model层去处理，处理结果保存后转发到JSP，然后展现数据。所以它的功能就是控制器的基本功能，它就管转发，在V和M之间转来转去。

再来说说M，即Model，在Java Web里说的是JavaBean，我认识的很多人都把JavaBean误认为是实体类，其实JavaBean有比实体类更丰富的定义，在JavaBean中除了其属性和字段，还可以有行为及其事件，JavaBean可以理解为普通Java对象。Java普通对象，就是符合Java规范的所有对象，这和实体类完全是两回事。所以，我认为在MVC中。业务逻辑和数据访问应该放在Model层，也就是V负责展示数据，Controler除了转发不做业务逻辑。真正的逻辑事务，数据访问，甚至**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "_blank)**都放到Model去。

再说三层架构。三层其实很好理解，界面，业务，数据访问，就这三个，从字面都可以理解出它们的意思。我要说的是它和MVC的区别。在三层架构中没有定义Controler的概念。这是我认为最不同的地方。而MVC也没有把业务的逻辑访问看成两个层，这是采用三层架构或MVC搭建程序最主要的区别。

当然了。在三层中也提到了Model，但是三层架构中Model的概念与MVC中Model的概念是不一样的，“三层”中典型的Model层是已实体类构成的，而MVC里，则是由业务逻辑与访问数据组成的。不一样的概念。虽然名字一样。

<http://blog.sina.com.cn/s/blog_677b66170100mv16.html>

**asp.net mvc和三层架构联系与区别**

首先,*MVC*和三层架构，是不一样的。  
　　三层架构中，DAL(数据访问层)、BLL(业务逻辑层)、WEB层各司其职，意在职责分离。  
　　*MVC*是 Model-View-Controller，严格说这三个加起来以后才是三层架构中的WEB层，也就是说，*MVC*把三层架构中的WEB层再度进行了分化，分成了控制器、视图、实体三个部分，控制器完成页面逻辑，通过实体来与界面层完成通话；而C层直接与三层中的BLL进行对话。  
　　所以， *.net*的三层结构中，并没有action这个概念。  
　　*asp.net mvc* 是微软新发布的一种网站开发架构。为了解决传统*asp.net*开发中不能分离Model,View和Controller而设计的。  
　　普通的网站为了解决可移植，可维护，可扩展等问题，会把网站设计成三个独立的模块，Model负责数据库部分，View负责网页的界面，而Controller负责界面与数据的交互及业务逻辑，这样设计的网站如果想设计或者重新开发某一个模块对其他的模块是没有影响的。但是*asp.net*的页面后台代码与每个页面代码都是一一对应的，业务逻辑在某些情况下不可避免的被写到了与View关联的后台代码中。这样就不能保证View与Controller的分离，也就很难实现网站的重写和升级。  
　　而在*MVC*中页面代码并不是与后台代码一一对应，而是分别被存放成Controller和View两个部分，彻底的解决了，View和Controller不能独立的问题。从而改善网站的重写和升级过程。  
　　但是*MVC*也有其缺点，由于在页面代码中不再可以使用服务器控件，因此给某些*asp.net*服务器端控件的使用带来了麻烦，而且*MVC*也页面的设计工作带来了很多障碍。  
　　*ASP.NET MVC* 是微软在2009年4月份发布的一种新的网站开发架构，http://msdn.microsoft.com/en-us/library/dd394709.a spx，它是把传统意义上的*MVC*开发思想融合到了*ASP.NET*的开发当中。  
　　那么我也来讲讲我对这两者的理解吧。  
　　首先对这个题目，本身是存在问题的，"XX结构"与"XX模式"的区别？请问中国社会制度与美国人生活方式有什么区别？  
　　这两者本身讲的是不同方向与角度的问题，在实际应用中他们的确存在一些相似的特点，在很多*书籍*中也没有深入讲解，以致于造成困惑，为了更好的理解他们，姑且来说说区别吧。  
　　首先N层结构是一种软件抽象的层次结构，是对复杂软件的一种纵向切分，每一层次中完成同一类型的操作，以便将各种代码以其完成的使命作为依据来分割，以将低软件的复杂度，提高其可维护性。一般来说，层次之间是向下依赖的，下层代码未确定其接口（契约）前，上层代码是无法开发的，下层代码接口（契约）的变化将使上层的代码一起变化。三层结构是N层结构的一种，是人产在长时间使用中得出来的一种应用场合广泛的N层结构，被当作一种典型的软件层次结构而广为流传甚至写入教科书。  
　　*MVC*模式是一种复合设计模式，一种在特定场合用于解决某种实际问题来得出的可以反复实践的解决方案。巧合的是他也有三个事物组成，于是乎人们就有了一种想当然的对应关系：展示层-View；业务逻辑层-Control；持久层-Model。首先*MVC*中的三个事物之间并不存在明显的层次结构，没有明显的向下依赖关系，相反的，View和Model往往是比较独立的，而Control是连接两者的桥梁，他们更像是横向的切分。这样一来就出现一个结果，*MVC*中每个块都是可以独立测试的，而三层结构中，上层模块的运行测试势必要提供下层代码或者提供相同接口的桩。相对来说，*MVC*复杂得多，但是结构更清晰，耦合性更低。  
　　另外，*MVC*中每一块内部特别是Model内部经常被设计为多层的。在我认为的一个良好的*MVC*模式构建的结构中，Control是核心，小且较为稳定的，可以作为一个核心框架来提供，有扩展点，但基本上可以简单配置不需要任何代码就可以运行。而View则可能是一套或多种可选择的视图引擎，决定了软件展示给用于的界面，使用时的主要工作量在于扩展点以及根据需要而数量不同的视图模板。Model则是业务提供者，决定了软件提供的功能，其内部可能是一些普通的类或者是实现了某些接口的类，在这一块当中可能根据业务的不同而色彩缤纷，对于复杂的软件可能会分成很多层，如业务逻辑层、业务提供层、系统提供层、数据提供层、数据访问层等。  
　　我经常用于比喻*MVC*的例子是小时候玩的那种卡带式游戏机，Control是主机，一般来说我买一个主机就行了，只要他不坏，他就能一直让我玩这一类的游戏。View则是电视机和游戏手柄，电视机可以独立工作，他不管输入的是电视信号、影碟机信号还是游戏机信号，他只管显示，而且他决定了我们看到的效果是怎么样的，如果我想要个尺寸更大的或者彩色的显示效果，我只需要买个相应的电视机就行了，手柄也是可以换的，要遥杆还是带震动的。Model则是游戏卡带，他绝定了我玩的是什么游戏，是魂斗罗还是超级玛莉，而且游戏机主机和电视机生产厂家永远也不知道在上面有可能会运行什么样的游戏。卡带中可能会有游戏代码和存储单元，都根据游戏的需要而设计。  
　　有朋友提到游戏主机提供的卡带插槽的接口，在设计中，有时也由Control提供一组接口，以用于Model或View的实现，这样就形成了依赖。一般来说这样设计也没有太大的问题，只是会提高模块间的耦合度，也会带来一些侵入性。为了更完美，可以不用接口来提供契约，可以用配置信息（或称元数据信息）+反射来提供契约，那么这个类接口就可以退化到只要符合CLS就可以了，也就是普通的类，就像现在的计算机接口广泛采用USB，无论是U盘、打印机、扫描仪或者是加密狗，他们都是普通的USB设备而已。  
　　提到USB有一个题外话，模块的可插拔性设计甚至是热插拔设计，系统可以在不停止运行的情况下动态的挂载或移除模块，动态挂载模块需要系统能够自动发现新模块并根据自描述的信息进行自动配置，移除可能情况更复杂一点，需要"安全删除硬件"类似的功能。  
　　在设计广泛重用的框架时会考虑多种情况以达到更大的适应性，一般项目中应用*MVC*模式可以较为随意。