JAVA三大框架和MVC架构的关系

JAVA三大框架的各自作用

hibernate是底层基于jdbc的orm(对象关系映射)持久化框架，即：表与类的映射，字段与属性的映射，记录与对象的映射 数据库模型 也就是Model;  
struts提供的丰富标签是用于显示层View;同时struts也充当了实现Control的功能(不管是Struts1，还是Struts2)，接收参数，视图分发。  
Spring是用于降低层与层间耦合度的，就是所有的类都可以由spring统一创建，用时只需注入即可(IOC)

从功能上来说，hibernate为M，struts2 为V，也可以为C，spring可以为C。  
但大多数项目中，hibernate为M，struts2 为V+C，spring用来帮助管理类以及实现一些业务无关功能，如日志记录等，大多数都是用struts来做C的。  
  
一、Spring  
  
Spring是一个解决了许多在J2EE开发中常见的问题的强大框架。 Spring提供了管理业务对象的一致方法并且鼓励了注入对接口编程而不是对类编程的良好习惯。Spring的架构基础是基于使用JavaBean属性的Inversion of Control容器。然而，这仅仅是完整图景中的一部分：Spring在使用IoC容器作为构建完关注所有架构层的完整解决方案方面是独一无二的。 Spring提供了唯一的数据访问抽象，包括简单和有效率的JDBC框架，极大的改进了效率并且减少了可能的错误。Spring的数据访问架构还集成了Hibernate和其他O/R mapping解决方案。Spring还提供了唯一的事务管理抽象，它能够在各种底层事务管理技术，例如JTA或者JDBC事务提供一个一致的编程模型。Spring提供了一个用标准Java语言编写的AOP框架，它给POJOs提供了声明式的事务管理和其他企业事务--如果你需要--还能实现你自己的aspects。这个框架足够强大，使得应用程序能够抛开EJB的复杂性，同时享受着和传统EJB相关的关键服务。Spring还提供了可以和IoC容器集成的强大而灵活的MVC Web框架。  
  
二、STRUCTS  
  
Struts是一个基于Sun J2EE平台的MVC框架，主要是采用Servlet和JSP技术来实现的。由于Struts能充分满足应用开发的需求，简单易用，敏捷迅速，在过去的一年中颇受关注。Struts把Servlet、JSP、自定义标签和信息资源(message resources)整合到一个统一的框架中，开发人员利用其进行开发时不用再自己编码实现全套MVC模式，极大的节省了时间，所以说Struts是一个非常不错的应用框架。  
  
三、Hibernate  
  
Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序实用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。映射很简单呀,就是页面的映射呀,多做点练习就会了。  
  
JAVA三大框架用来做WEB应用。 struts主要负责表示层的显示 spring利用它的IOC和AOP来处理控制业务（负责对数据库的操作） hibernate主要是数据的持久化到数据库。在用jsp的servlet做网页开发的时候有个web.xml的映射文件，里面有个mapping的标签就是用来做文件映射的。 当你在浏览器上输入URL地址的时候，文件就会根据你写的名称对应到一个java文件，根据java文件里编写的内容显示在浏览器上，就是一个网页，所以网页名字是随便写的，随便起的名字，不管你起的是.php .jsp .do还是其他的什么都对应这个java文件，这个java文件里的代码进行什么操作就干什么，显示一句话还是连接数据库还是跳转到其他页面等等，这个java文件把数据进行封装起到安全和便于管理的作用。其实这个java文件编译过来是.class的一个字节码文件，没有那么一个类似html嵌入标签和代码的网页文件。他与jsp文件区别就是jsp把代码嵌入到html标签中。 还有servlet或者struts中html表单的action中的提交名称对应一个java文件，struts一般是.do的，都和上面一样是映射。

1. 谈谈你mvc的理解  
MVC是Model—View—Controler的简称。即模型—视图—控制器。MVC是一种设计模式，它强制性的把应用程序的输入、处理和输出分开。  
MVC中的模型、视图、控制器它们分别担负着不同的任务。  
视图: 视图是用户看到并与之交互的界面。视图向用户显示相关的数据，并接受用户的输入。视图不进行任何业务逻辑处理。  
模型: 模型表示业务数据和业务处理。相当于JavaBean。一个模型能为多个视图提供数据。这提高了应用程序的重用性  
控制器: 当用户单击Web页面中的提交按钮时,控制器接受请求并调用相应的模型去处理请求。  
然后根据处理的结果调用相应的视图来显示处理的结果。  
MVC的处理过程：首先控制器接受用户的请求，调用相应的模型来进行业务处理，并返回数据给控制器。控制器调用相应的视图来显示处理的结果。并通过视图呈现给用户。  
  
Struts  
1. struts1.2和struts2.0的区别？如何控制两种框架中的单例模式？  
struts1.2和struts2.0的对比  
a、Action类：  
struts1.2要求Action类继承一个基类。struts2.0 Action要求继承ActionSupport基类  
b、线程模式  
struts1.2 Action是单例模式的并且必须是线程安全的,因为仅有一个Action的实例来处理所有的请求。  
单例策略限制了Struts1.2 Action能做的事情,并且开发时特别小心。Action资源必须是线程安全的或同步的。  
struts2.0 Action为每一个请求产生一个实例,因此没有线程安全问题。  
c、Servlet依赖  
struts1.2 Action依赖于Servlet API,因为当一个Action被调用时HttpServletRequest和HttpServletResponse被传递给execut方法。  
struts2.0 Action不依赖于容器,允许Action脱离容器单独测试。如果需要,Struts2 Action仍然可以访问初始的Request和Response。  
但是,其他的元素减少或者消除了直接访问HttpServletRequest和HttpServletResponse的必要性。  
d、可测性  
测试struts1.2 Action的一个主要问题是execute方法暴露了Servlet API(这使得测试要依赖于容器)。一个第三方扩展：struts TestCase  
提供了一套struts1.2的模拟对象来进行测试。  
Struts2.0 Action可以通过初始化、设置属性、调用方法来测试,“依赖注入”也使得测试更容易。  
  
2. 项目中为什么使用SSH  
1. 使用Struts是因为struts是基于MVC模式的,很好的将应用程序进行了分层，使开发者更关注于业务逻辑的实现；第二，struts有着丰富的taglib,如能灵活运用，则能大大提高开发效率。  
2. 使用Hibernate：因为hibernate为Java应用提供了一个易用的、高效率的对象关系映射框架。hibernate是个轻量级的持久性框架，功能丰富。  
3. 使用Spring：因为spring基于IoC(Inversion of Control，反向控制)和AOP构架多层j2ee系统的框架，但它不强迫你必须在每一层中必须使用Spring，因为它模块化的很好，允许你根据自己的需要选择使用它的某一个模块；采用IoC使得可以很容易的实现bean的装配，提供了简洁的AOP并据此实现事务管理(Transcation Managment),等等   
  
3. struts与spring是如何继承的  
不懂题目之意：是否是说的struts的action都继承自org.apache.struts.action.Action而Spring中的继承就不清楚了。  
  
4. struts如何实现国际化  
以下以两国语言(中文，英文)为例：  
1. 在工程中加入Struts支持  
2. 编辑ApplicationResource.properties文件,在其中加入要使用国际化的信息, 例如：   
lable.welcome.china=Welcome!!!  
3. 创建英文资源文件ApplicationResource\_en.properites  
4. 创建临时中文资源文件ApplicationResource\_temp.properites 例如：lable.welcom.china=中国欢迎您!  
5. 对临时中文资源文件进行编码转换。可以使用myeclipse的插件,也可以在dos下执行:  
native2ascii -encoding gb2312 ApplicationResource\_temp.properties ApplicationResource\_zh\_CN.properties  
6. 在jsp中加入struts的bean标记库 <%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>  
显示信息：   
可以在struts-config.xml文件中定义多个资源包，而每个资源包可使用key属性指定包的名称。  
  
5. struts2.0的常用标签  
1. 往action里传值：  
2. 显示标签 property 用于输出指定值：  
3. 用于从页面往action中(user)的对象内传值：  
4. 判断 用于在页面中判断指定数据 …. ….   
5. 迭代用于将List、Map、ArrayList等集合进行循环遍历  
6. URL地址标签，用于生成一个URL地址，可以通过URL标签指定的子元素向URL地址发送请求参数  
7. 超链接 一般和标签一起使用，用于带多个参数。“>超链接  
8. set标签，用于将某个值放入指定的范围内。例如application,session等。  
  
6. struts中怎么配置form-bean、action、tiles  
此处配置的是struts1.2的  
form-bean配置：(在配置文件struts-config.xml中)：  
name: 指定form的名字; type指定form的类型：包名+类名；  
action配置：(在配置文件struts-config.xml中)  
attribute=""  
input=""  
name=""  
parameter=""  
scope="request"  
type=""  
>  
path：请求Action的名字; attribute：form的名字,与form-bean中的name对应; input：输入页的路径;  
name：如果配置了attribute,name不起作用,与attribute一样; parameter：使用分发Action时,指定调用分发  
Action中的方法名;

scope：Action的范围; type：Action的类型：包名+类名；

7.什么是IOC（DI），有什么好处(java project 的spring IOC模拟)

IOC（Inversion Of Control，控制反转）：一个类的某个属性需要使用时，需要有本类自己初始化。而在spring类的某些属性不用自己初始化而交给了spring来管理。这样的转换就是IOC思想。Spring的IOC思想是，spring的IOC容器在初始化的时候就根据配置对相应的对象进行初始化，并放在容器中。同时根据配置对相应类的属性进行初始化。也就是DI（Dependency Injection依赖注入），至此类的初始化不需要类本身负责初始化而交给了spring容器。  
IOC的好处是：降低耦合性：某个类的一个属性，以前是该类自己控制，而引人Spring后，这个类的属性由Spring控制。  
Spring IOC的模拟所做的事情是：在xml文件配置好相应的bean，设置注入关系。在解析xml时候对bean进行初始化同时对相应属性进行注入。

8.什么是AOP?

两种注入方式：

a) 使用Annotation  
b) 使用xml

面向切面编程（AOP）完善spring的依赖注入（DI），面向切面编程在spring中主要表现为两个方面  
1.面向切面编程提供声明式事务管理  
2.spring支持用户自定义的切面  
  
面向切面编程（aop）是对面向对象编程（oop）的补充，面向对象编程将程序分解成各个层次的对象，面向切面编程将程序运行过程分解成各个切面。  
AOP从程序运行角度考虑程序的结构，提取业务处理过程的切面，oop是静态的抽象，aop是动态的抽象，是对应用执行过程中的步骤进行抽象，从而获得步骤之间的逻辑划分。  
aop框架具有的两个特征：  
1.各个步骤之间的良好隔离性  
2.源代码无关性

9.DAO层采用JdbcTemplate操作数据，如何在Service层控制事务？  
DAO层采用JdbcTemplate进行数据库操作。   
Service层调用DAO层方法。   
由于一个Service方法可能需要调用多个DAO对象的方法，需要在Service层进行事务控制。   
由于一些原因，需要采用编程式事务(现使用TransactionTemplate)。   
可是存在一个问题：   
只有当创建TransactionTemplate对象的DataSourceTransactionManager使用的DataSource对象和DAO对象创建JdbcTemplate对象使用的DataSource是同一个对象时才能在transactionTemplate.execute()方法中控制事务。

软件的架构是系统的一个草图、阐述了各个组件之间的通讯、划分层次、一旦系统开始详细设计、架构蓝图就很难甚至无法更改、是由软件架构师从无到有设计出来的。三层架构：一种设计软件架构的思想 ，把软件上从逻辑上分为、表示层（UI）业务逻辑层（BLL）数据访问层（DAL），表现层（UI）：通俗讲就是展现给用户的界面，即用户在使用一个系统的时候他的所见所得。业务逻辑层（BLL）：针对具体问题的操作，也可以说是对数据层的操作，对数据业务逻辑处理。数据访问层（DAL）：该层所做事务直接操作数据库，针对数据的增添、删除、修改、更新、查找等。目的：低耦合、高内聚、各司其职、达到易更换、修改、可以分散部署、编码。软件框架是在一定领域内、别人已经对这个领域制作软件所需的基础架构功能、进行了总结、做出了有代码实体的软件框架结构、如果要制作这一领域的软件、可以在别人写好的框架上、继续设计、编写自己的软件、骨头架上填肉、框架有一定的局限性。MVC（框架），英文  Model View Controller、是针对Web开发、已经写好有代码的框架、分别为M 模型(model)－V视图(view)－C控制器(controller)三部分；目的：模型和视图分离开、使得一个模型可被多个视图使用、简单说就是同样的一个网站、用手机的视图（界面）和电脑的视图、可以共用一个模型。

MVC主要用于表现层，3层主要用于体系架构，3层一般是表现层、中间层、数据层，其中表现层又可以分成M、V、C，(Model View Controller)模型－视图－控制器，MVC模式是GUI界面开发的指导模式，基于表现层分离的思想把程序分为三大部分：Model-View-Controller，呈三角形结构。Model是指数据以及应用程序逻辑，View是指 Model的视图，也就是用户界面。这两者都很好理解，关键点在于Controller的角色以及三者之间的关系。在MVC模式中，Controller和View同属于表现层，通常成对出现。Controller被设计为处理用户交互的逻辑。一个通常的误解是认为Controller负责处理View和Model的交互，而实际上View和Model之间是可以直接通信的。由于用户的交互通常会涉及到Model的改变和View的更新，所以这些可以认为是Controller的副作用。MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开，它们各自处理自己的任务。而在N层架构中，一般还会有一个Model层，用来与数据库的表相对应，也就是所谓O/RM中的O。这个Model可能是POCO，也可能是包含一些验证逻辑的实体类，一般也不包含数据读取。进行数据读取的是数据访问层。而作为UI层的MVC一般不直接操作数据访问层，中间会有一个业务逻辑层封装业务逻辑、调用数据访问层。UI层（Controller）通过业务逻辑层来得到数据（Model），并进行封装（ViewModel），然后选择相应的View。MVC本来是存在于Desktop程序中的，M是指数据模型，V是指用户界面，C则是控制器。使用MVC的目的是将M和V的实现代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据你可以分别用柱状图、饼图来表示。C存在的目的则是确保M和V的同步，一旦M改变，V应该同步更新。MVC如何工作MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。视图V视图是用户看到并与之交互的界面。对老式的Web应用程序来说，视图就是由HTML元素组成的界面，在新式的Web应用程序中，HTML依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括Macromedia Flash和象XHTML，XML/XSL，WML等一些标识语言和Web services.如何处理应用程序的界面变得越来越有挑战性。MVC一个大的好处是它能为你的应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。模型M模型表示企业数据和业务规则。在MVC的三个部件中，模型拥有最多的处理任务。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。控制器C控制器接受用户的输入并调用模型和视图去完成用户的需求。所以当单击Web页面中的超链接和发送HTML表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。模型Model 模型是应用程序的主体部分。模型表示业务数据，或者业务逻辑。 实现具体的业务逻辑、状态管理的功能。视图View 视图是应用程序中用户界面相关的部分，是用户看到并与之交互的界面。 就是与用户实现交互的页面，通常实现数据的输入和输出功能。控制器controller 控制器工作就是根据用户的输入，控制用户界面数据显示和更新model对象状态。起到控制整个业务流程的作用，实现View层跟Model层的协同工作。

MVC和三层架构的相同点是分层，解耦。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。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。三层架构中，DAL、BLL、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。

一件事，要知其然往往很简单，要知其所以然通常不是那么容易，就如最近重新巩固spring的过程中，就觉得还有许多问题其实并不是十分明了。   
屈指一算，手头上做过的正式项目也有了四五六七个了，不管用的数据库和其他一些细节上的技术如何，总的来说大的框架结构都是差不多的。   
说白了，也就是mvc和三层结构。   
而mvc和三层结构究竟是什么关系，我曾在面试的过程中被人问过几次，也曾仔细的想过、查过这个问题，但是直到此时，我也还是不能完全确定。

只不过随着时间的积累，随着技术的沉淀，随着视野的拓宽，我大体上认同了两种说法，不管别人怎么看，我个人是觉得两种说法都有道理，欢迎对这个问题有不同看法的朋友一起讨论。   
三层结构是什么，是展现层、应用层、数据访问层，这个基本上是没有太大的异议的，两种看法的来源基本上都是来自对于mvc的理解。   
对于java web应用来说，不管是B/S还是C/S，大体上都可以分成服务端和客户端两部分，只不过B/S的客户端就是公用的浏览器。

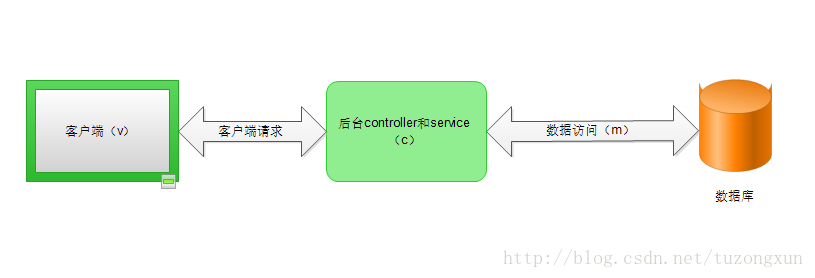
基于这种大的架构，有一种对于mvc的说法就是：

m是model，也就是和数据库相关的那些，比如实体类和dao、mapper.xml等，对应着三层结构的数据访问层；

v是view，也就是前台的页面或者说是客户端展示给用户看的东西，也就是展现层；

而c就是controller以及service等具体的业务逻辑，对应着三层结构的应用层。

* 1
* 2
* 3

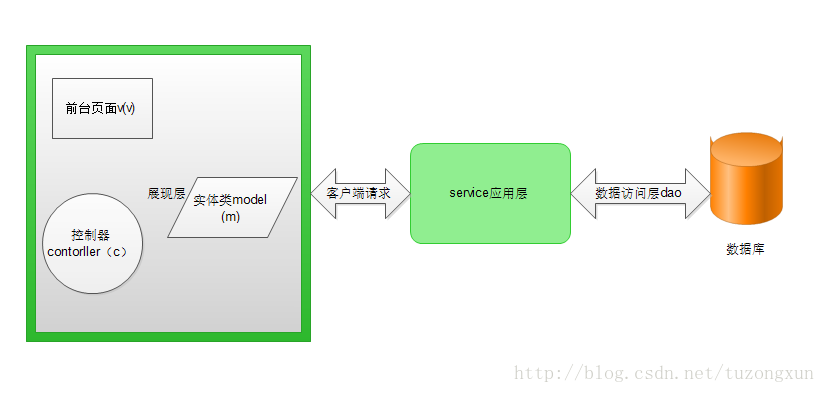
   
这种说法我觉得应该是为了对应而对应，就是要把mvc和三层结构的关系一一对应起来，因此也差不多就是一个对应一个。   
或许是经验还不够多的缘故，或许本来就是这样子，反正我是觉得这种说法正确，或者更确切的说，这是大的mvc。

那么还有一种说法可能就不是像上边这样一一对应了，大体上的说法是这样的：

我们通常所说的mvc实际上只是对应了三层结构中的展现层，也就是view而已，然后v是客户端，m是实体，c是controller。

至于三层结构中的应用层和数据访问层则是分别对应了后台代码中的service和dao。

* 1
* 2

   
而对于这种说法，为什么我觉得有道理呢？是因为按照这种描述，就是和前台展示直接相关的东西都放在展现层。   
\*比如controller要直接和url打交道，而很多时候返回给客户端的数据也会封装成对象的形式，经常就是model；也就是说不管是controller还是model，都实打实和用户看得到的部分相关，就划为了展现层。   
只不过在某些时候，就比如我们现在的项目中，为了进一步的实现松耦合，我们会创建一个command类，类似于实体model，然后用model操作数据库，用command和前台打交道，道理是一样的。   
而在另一方面，我们现在项目前端使用的技术是angular js，这项技术现在也说实现了前台的mvc，有controller、service，还有数据层。   
因此在这种情况下，我个人就觉得，mvc本就是一个概念，重要的是一种理解，它本身的作用只是为了实现松耦合，而不是为了mvc而mvc，未必一定要有一个唯一的答案！