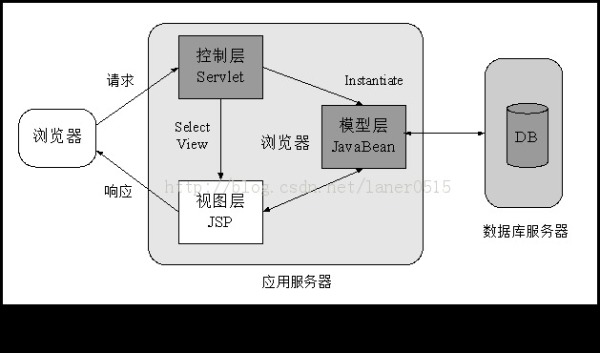
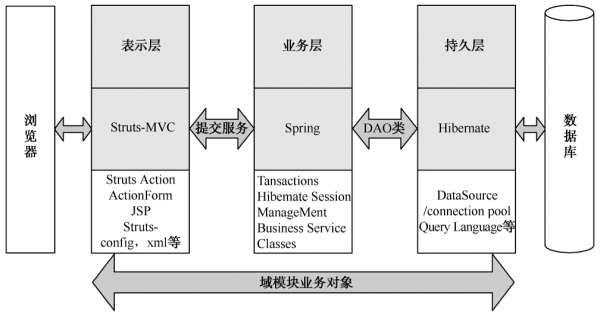
**JAVA Web的MVC设计模式**



传统的Java Web应用程序是采用JSP+Servlet+Javabean来实现的，这种模式实现了最基本的MVC分层，使的程序结构分为几层，有负责前台展示的 JSP、负责流程逻辑控制的Servlet以及负责数据封装的Javabean。但是这种结构仍然存在问题：如JSP页面中需要使用符号嵌入很多的 Java代码，造成页面结构混乱，Servlet和Javabean负责了大量的跳转和运算工作，耦合紧密，程序复用度低等等。

**Struts**

为了解决这些问题，出现了Struts框架，它是一个完美的MVC实现，它有一个中央控制类(一个 Servlet)，针对不同的业务，我们需要一个Action类负责页面跳转和后台逻辑运算，一个或几个JSP页面负责数据的输入和输出显示，还有一个 Form类负责传递Action和JSP中间的数据。JSP中可以使用Struts框架提供的一组标签，就像使用HTML标签一样简单，但是可以完成非常复杂的逻辑。从此JSP页面中不需要出现一行包围的Java代码了。

可是所有的运算逻辑都放在Struts的Action里将使得 Action类复用度低和逻辑混乱，所以通常人们会把整个Web应用程序分为三层，Struts负责显示层，它调用业务层完成运算逻辑，业务层再调用持久层完成数据库的读写。

使用JDBC连接来读写数据库，我们最常见的就是打开数据库连接、使用复杂的SQL语句进行读写、关闭连接，获得的数据又需要转换或封装后往外传，这是一个非常烦琐的过程。

**Hibernate**

这时出现了 Hibernate框架，它需要你创建一系列的持久化类，每个类的属性都可以简单的看做和一张数据库表的属性一一对应，当然也可以实现关系数据库的各种表件关联的对应。当我们需要相关操作是，不用再关注数据库表。我们不用再去一行行的查询数据库，只需要持久化类就可以完成增删改查的功能。使我们的软件开发真正面向对象，而不是面向混乱的代码。

现在我们有三个层了，可是每层之间的调用是怎样的呢?比如显示层的Struts需要调用一个业务类，就需要new一个业务类出来，然后使用;业务层需要调用持久层的类，也需要new一个持久层类出来用。通过这种new的方式互相调用就是软件开发中最糟糕设计的体现。简单的说，就是调用者依赖被调用者，它们之间形成了强耦合，如果我想在其他地方复用某个类，则这个类依赖的其他类也需要包含。程序就变得很混乱，每个类互相依赖互相调用，复用度极低。如果一个类做了修改，则依赖它的很多类都会受到牵连。为此，出现Spring框架。

**Spring**

Spring的作用就是完全解耦类之间的依赖关系，一个类如果要依赖什么，那就是一个接口。至于如何实现这个接口，这都不重要了。只要拿到一个实现了这个接口的类，就可以轻松的通过xml配置文件把实现类注射到调用接口的那个类里。所有类之间的这种依赖关系就完全通过配置文件的方式替代了。所以 Spring框架最核心的就是所谓的依赖注射和控制反转。

现在的结构是，Struts负责显示层，Hibernate负责持久层，Spring负责中间的业务层，另外，由于Spring使用的依赖注射以及AOP(面向方面编程)，所以它的这种内部模式非常优秀，以至于Spring自己也实现了一个使用依赖注射的MVC框架，叫做Spring MVC，同时为了很好的处理事物，Spring集成了Hibernate，使事物管理从Hibernate的持久层提升到了业务层，使用更加方便和强大。

**从知乎整理:https://www.zhihu.com/question/30525693/answer/219133341**