在Struts + Spring + Hibernate的组合框架模式中，三者各自的特点都是什么？
Struts 的MVC设计模式可以使我们的逻辑变得很清晰。
Spring 的IOC和AOP可以使我们的产品在最大限度上解耦。
hibernate的当然就是实体对象的持久化了
典型的J2EE三层结构，分为表现层、中间层（业务逻辑层）和数据服务层。三层体系将业务规则、数据访问及合法性校验等工作放在中间层处理。客户端不直接与数据库交互，而是通过组件与中间层建立连接，再由中间层与数据库交互。
表现层是传统的JSP技术，自 1999 年问世以来，经过多年的发展，tb其广泛的应用和稳定的表现，为其作为表现层技术打下了坚实的基础。
中间层采用的是流行的Spring+Hibernate，为了将控制层与业务逻辑层分离，又细分为以下几种。
Web层，就是MVC模式里面的“C”（controller），负责控制业务逻辑层与表现层的交互，调用业务逻辑层，并将业务数据返回给表现层作组织表现，该系统的MVC框架采用Struts。
Service层（就是业务逻辑层），负责实现业务逻辑。业务逻辑层以DAO层为基础，通过对DAO组件的正面模式包装，完成系统所要求的业务逻辑。
DAO层，负责与持久化对象交互。该层封装了数据的增、删、查、改的操作。
PO，持久化对象。通过实体关系映射工具将关系型数据库的数据映射成对象，很方便地实现以面向对象方式操作数据库，该系统采用Hibernate作为ORM框架。
Spring的作用贯穿了整个中间层，将Web层、Service层、DAO层及PO无缝整合，其数据服务层用来存放数据。
一个良好的框架可以让开发人员减轻重新建立解决复杂问题方案的负担和精力;它可以被扩展以进行内部的定制化;并且有强大的用户社区来支持它。框架通常能很好的解决一个问题。然而，你的应用是分层的，可能每一个层都需要各自的框架。仅仅解决UI问题并不意味着你能够很好的将业务逻辑和持久性逻辑和UI组件很好的耦合。

不可否认，对于简单的应用，采用ASP或者PHP的开发效率比采用J2EE框架的开发效率要高。甚至有人会觉得：这种分层的结构，比一般采用JSP + Servlet的系统开发效率还要低。
笔者从一下几个角度来阐述这个问题。
— 开发效率：软件工程是个特殊的行业，不同于传统的工业，例如电器、建筑及汽车等行业。这些行业的产品一旦开发出来，交付用户使用后将很少需要后续的维护。但软件行业不同，软件产品的后期运行维护是个巨大的工程，单纯从前期开发时间上考虑其开发效率是不理智的，也是不公平的。众所周知，对于传统的ASP和 PHP等脚本站点技术，将整个站点的业务逻辑和表现逻辑都混杂在ASP或PHP页面里，从而导致页面的可读性相当差，可维护性非常低。即使需要简单改变页面的按钮，也不得不打开页面文件，冒着破坏系统的风险。但采用严格分层J2EE架构，则可完全避免这个问题。对表现层的修改即使发生错误，也绝对不会将错误扩展到业务逻辑层，更不会影响持久层。因此，采用J2EE分层架构，即使前期的开发效率稍微低一点，但也是值得的。
— 需求的变更：以笔者多年的开发经验来看，很少有软件产品的需求从一开始就完全是固定的。客户对软件需求，是随着软件开发过程的深入，不断明晰起来的。因此，常常遇到软件开发到一定程度时，由于客户对软件需求发生了变化，使得软件的实现不得不随之改变。当软件实现需要改变时，是否可以尽可能多地保留软件的部分，尽可能少地改变软件的实现，从而满足客户需求的变更？答案是——采用优秀的解耦架构。这种架构就是J2EE的分层架构，在优秀的分层架构里，控制层依赖于业务逻辑层，但绝不与任何具体的业务逻辑组件耦合，只与接口耦合;同样，业务逻辑层依赖于DAO层，也不会与任何具体的DAO组件耦合，而是面向接口编程。采用这种方式的软件实现，即使软件的部分发生改变，其他部分也尽可能不要改变。
注意：即使在传统的硬件行业，也有大量的接口规范。例如PCI接口、显卡或者网卡，只要其遵守PCI的规

范，就可以插入主板，与主板通信。至于这块卡内部的实现，不是主板所关心的，这也正是面向接口编程的好处。假如需要提高电脑的性能，需要更新显卡，只要更换另一块PCI接口的显卡，而不是将整台电脑抛弃。如果一台电脑不是采用各种接口组合在一起，而是做成整块，那将意味着即使只需要更新网卡，也要放弃整台电脑。同样，对于软件中的一个个组件，当一个组件需要重构时，尽量不会影响到其他组件。实际上，这是最理想的情况，即使采用目前最优秀的架构，也会有或多或少的影响，这也是软件工程需要努力提高的地方。

技术的更新，系统重构：软件行业的技术更新很快，虽然软件行业的发展不快，但小范围的技术更新特别快。一旦由于客观环境的变化，不得不更换技术时，如何保证系统的改变最小呢？答案还是选择优秀的架构。

在传统的Model 1 的程序结构中，只要有一点小的需求发生改变，将意味着放弃整个页面。或者改写。虽然前期的开发速度快，除非可以保证以后永远不会改变应用的结构，否则不要采用Model 1 的结构。

采用Hibernate作为持久层技术的最大的好处在于：可以完全以面向对象的方式进行系统分析、系统设计。

DAO模式需要为每个DAO组件编写DAO接口，同时至少提供一个实现类，根据不同需要，可能有多个实现类。用Spring容器代替DAO工厂

通常情况下，引入接口就不可避免需要引入工厂来负责DAO组件的生成。Spring实现了两种基本模式：单态模式和工厂模式。而使用Spring可以完全避免使用工厂模式，因为Spring就是个功能非常强大的工厂。因此，完全可以让Spring充当DAO工厂。

由Spring充当DAO工厂时，无须程序员自己实现工厂模式，只需要将DAO组件配置在Spring容器中，由ApplicationContext负责管理DAO组件的创建即可。借助于Spring提供的依赖注入，其他组件甚至不用访问工厂，一样可以直接使用DAO实例。

优点：

Struts跟Tomcat、Turbine等诸多Apache项目一样，是开源软件，这是它的一大优点。使开发者能更深入的了解其内部实现机制。

除此之外，Struts的优点主要集中体现在两个方面：Taglib和页面导航。Taglib是Struts的标记库，灵活动用，能大大提高开发效率。另外，就目前国内的JSP开发者而言，除了使用JSP自带的常用标记外，很少开发自己的标记，或许Struts是一个很好的起点。

关于页面导航，我认为那将是今后的一个发展方向，事实上，这样做，使系统的脉络更加清晰。通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。

缺点：

Taglib是Struts的一大优势，但对于初学者而言，却需要一个持续学习的过程，甚至还会打乱你网页编写的习惯，但是，当你习惯了它时，你会觉得它真的很棒。

Struts将MVC的Controller一分为三，在获得结构更加清晰的同时，也增加了系统的复杂度。

Struts从产生到现在还不到半年，但已逐步越来越多运用于商业软件。虽然它现在还有不少缺点，但它是一种非常优秀的J2EE MVC实现方式，如果你的系统准备采用J2EE MVC架构，那么，不妨考虑一下Struts。