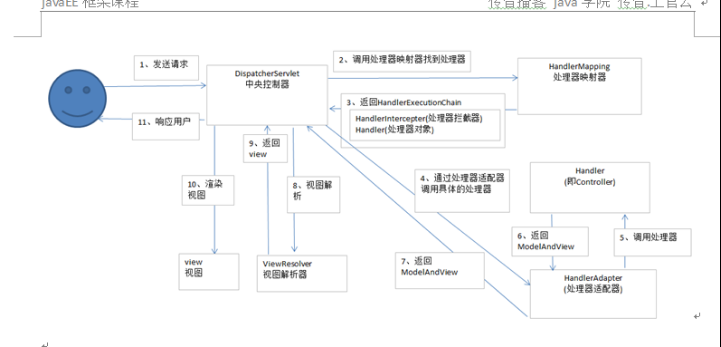
**springMvc简单介绍下你对springMVC的理解?**

Spring MVC Framework有这样一些特点:

1. 它是基于组件技术的.全部的应用对象,无论控制器和视图,还是业务对象之类的都是java组件.并且和Spring提供的其他基础结构紧密集成.
2. 不依赖于Servlet API(目标虽是如此,但是在实现的时候确实是依赖于Servlet的)
3. 可以任意使用各种视图技术,而不仅仅局限于JSP
4. 支持各种请求资源的映射策略
5. 它应是易于扩展的

**2) SpringMVC的工作流程?**  
  
1. 用户发送请求至前端控制器DispatcherServlet  
2. DispatcherServlet收到请求调用HandlerMapping处理器映射器。  
3. 处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。  
4. DispatcherServlet通过HandlerAdapter处理器适配器调用处理器  
5. 执行处理器(Controller，也叫后端控制器)。  
6. Controller执行完成返回ModelAndView  
7. HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet  
8. DispatcherServlet将ModelAndView传给ViewReslover视图解析器  
9. ViewReslover解析后返回具体View  
10. DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）。  
11. DispatcherServlet响应用户  
  
**3) 如果你也用过struts2.简单介绍下springMVC和struts2的区别有哪些?**  
  
  
1. springmvc的入口是一个servlet即前端控制器，而struts2入口是一个filter过虑器。  
2. springmvc是基于方法开发(一个url对应一个方法)，请求参数传递到方法的形参，可以设计为单例或多例(建议单例)，struts2是基于类开发，传递参数是通过类的属性，只能设计为多例。  
3. Struts采用值栈存储请求和响应的数据，通过OGNL存取数据， springmvc通过参数解析器是将request请求内容解析，并给方法形参赋值，将数据和视图封装成ModelAndView对象，最后又将ModelAndView中的模型数据通过reques域传输到页面。Jsp视图解析器默认使用jstl。  
  
**4) SpringMvc原理？  
  
  
  
5) SSM优缺点、使用场景？**  
  
  
1. Mybatis和hibernate不同，它不完全是一个ORM框架，因为MyBatis需要程序员自己编写Sql语句，不过mybatis可以通过XML或注解方式灵活配置要运行的sql语句，并将java对象和sql语句映射生成最终执行的sql，最后将sql执行的结果再映射生成java对象。  
  
2. Mybatis学习门槛低，简单易学，程序员直接编写原生态sql，可严格控制sql执行性能，灵活度高，非常适合对关系数据模型要求不高的软件开发，例如互联网软件、企业运营类软件等，因为这类软件需求变化频繁，一但需求变化要求成果输出迅速。但是灵活的前提是mybatis无法做到数据库无关性，如果需要实现支持多种数据库的软件则需要自定义多套sql映射文件，工作量大。  
  
3. Hibernate对象/关系映射能力强，数据库无关性好，对于关系模型要求高的软件（例如需求固定的定制化软件）如果用hibernate开发可以节省很多代码，提高效率。但是Hibernate的学习门槛高，要精通门槛更高，而且怎么设计O/R映射，在性能和对象模型之间如何权衡，以及怎样用好Hibernate需要具有很强的经验和能力才行。  
4. 总之，按照用户的需求在有限的资源环境下只要能做出维护性、扩展性良好的软件架构都是好架构，所以框架只有适合才是最好。

**Mybatis**  
**1) 简单介绍下你对mybatis的理解？**  
  
  
1. mybatis配置  
2. SqlMapConfig.xml，此文件作为mybatis的全局配置文件，配置了mybatis的运行环境等信息。  
3. mapper.xml文件即sql映射文件，文件中配置了操作数据库的sql语句。此文件需要在SqlMapConfig.xml中加载。  
4. 通过mybatis环境等配置信息构造SqlSessionFactory即会话工厂  
5. 由会话工厂创建sqlSession即会话，操作数据库需要通过sqlSession进行。  
6. mybatis底层自定义了Executor执行器接口操作数据库，Executor接口有两个实现，一个是基本执行器、一个是缓存执行器。  
7. Mapped Statement也是mybatis一个底层封装对象，它包装了mybatis配置信息及sql映射信息等。mapper.xml文件中一个sql对应一个Mapped Statement对象，sql的id即是Mapped statement的id。  
8. Mapped Statement对sql执行输入参数进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql前将输入的java对象映射至sql中，输入参数映射就是jdbc编程中对preparedStatement设置参数。  
9. Mapped Statement对sql执行输出结果进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql后将输出结果映射至java对象中，输出结果映射过程相当于jdbc编程中对结果的解析处理过程。

Hibernate优点:  
  
  
(1) 对象/关系数据库映射(ORM)  
它使用时只需要操纵对象，使开发更对象化，抛弃了数据库中心的思想，完全的面向对象思想  
(2) 透明持久化(persistent)  
带有持久化状态的、具有业务功能的单线程对象，此对象生存期很短。这些对象可能是普通的JavaBeans/POJO，这个对象没有实现第三方框架或者接口，唯一特殊的是他们正与（仅仅一个）Session相关联。一旦这个Session被关闭，这些对象就会脱离持久化状态，这样就可被应用程序的任何层自由使用。（例如，用作跟表示层打交道的数据传输对象。）        
(3) 事务Transaction(org.hibernate.Transaction)  
应用程序用来指定原子操作单元范围的对象，它是单线程的，生命周期很短。它通过抽象将应用从底层具体的JDBC、JTA以及CORBA事务隔离开。某些情况下，一个Session之内可  
能包含多个Transaction对象。尽管是否使用该对象是可选的，但无论是使用底层的API还是使用Transaction对象，事务边界的开启与关闭是必不可少的。  
(4) 它没有侵入性，即所谓的轻量级框架  
(5) 移植性会很好  
(6) 缓存机制，提供一级缓存和二级缓存  
(7) 简洁的HQL编程  
  
  
  
  
Hibernate缺点:  
(1)  Hibernate在批量数据处理时有弱势  
(2) 针对单一对象简单的增删查改，适合于Hibernate,而对于批量的修改，删除，不适合用Hibernate,这也是OR框架的弱点；要使用数据库的特定优化机制的时候，不适合用  
  
(3) 优化策略应用不当会导致大量的资源消耗.  
  
  
更多问题总结，回帖可见  
  
  
本帖隐藏的内容  
  
  
Spring优缺点    
  
它是一个开源的项目，而且目前非常活跃；它基于IoC（Inversion of Control，反向控制）和AOP的构架多层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；它实现了很优雅的MVC，对不同的数据访问技术提供了统一的 接口，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Transcation Managment，等等  
  
spring优点:  
a. Spring能有效地组织你的中间层对象，不管你是否选择使用了EJB。如果你仅仅使用了Struts或其他为J2EE的 API特制的framework，Spring致力于解决剩下的问题。  
b. Spring能消除在许多工程中常见的对Singleton的过多使用。根据我的经验，这是一个很大的问题，它降低了系统的可测试性和面向对象的程度。  
c. 通过一种在不同应用程序和项目间一致的方法来处理配置文件，Spring能消除各种各样自定义格式的属性文件的需要。曾经对某个类要寻找的是哪个魔法般的属性项或系统属性  
  
感到不解，为此不得不去读Javadoc甚至源编码？有了Spring，你仅仅需要看看类的JavaBean属性。Inversion of Control的使用（在下面讨论）帮助完成了这种简化。  
d. 通过把对接口编程而不是对类编程的代价几乎减少到没有，Spring能够促进养成好的编程习惯。  
e. Spring被设计为让使用它创建的应用尽可能少的依赖于他的APIs。在Spring应用中的大多数业务对象没有依赖于Spring。  
f. 使用Spring构建的应用程序易于单元测试。  
g. Spring能使EJB的使用成为一个实现选择,而不是应用架构的必然选择。你能选择用POJOs或local EJBs来实现业务接口，却不会影响调用代码。  
h. Spring帮助你解决许多问题而无需使用EJB。Spring能提供一种EJB的替换物，它们适用于许多web应用。例如，Spring能使用AOP提供声明性事务管理而不通过EJB容器，如果你仅仅需要与单个数据库打交道，甚至不需要一个JTA实现。   
i. Spring为数据存取提供了一个一致的框架,不论是使用的是JDBC还是O/R mapping产品（如Hibernate）。  
Spring确实使你能通过最简单可行的解决办法来解决你的问题。而这是有有很大价值的。  
  
Spring缺点:  
  
无明显缺点  
  
Struts2 的优点：  
Struts2比Struts1已经有了很大的进步，优点很多，其中主要两个是：对框架API和ServletAPI的依赖减少，可扩展性提高。  
  
Struts2的Action可以实现框架提供的Action接口也可以不实现这个接口。实际上框架Strut2的Action的要求很低，只要一个类，包含一个无参的、返回值类型为String的方法就行。其实Struts2的Action就是一个POJO。如果用户写一个类实现框架提供的Action接口或者继承框架提供的ActionSupport类， 则可以利用框架中的其他一些功能。比如在，Action接口中定义了一些常量，这些常量通常作为返回值共处理方法调用。  
  
由于Struts2的Action对框架API和Servlet API的依赖减少，因此可测程度大大提高。  
  
Struts2的可扩展性提高了。Struts2的核心jar包中由一个struts-default.xml文件，在该文件中设置了一些默认的bean,resultType类型，默认拦截器栈等，所有这些默认设置，用户都可以利用配置文件更改，可以更改为自己开发的bean，resulttype等。  
  
因此用户开发了插件的话只要很简单的配置就可以很容易的和Struts2框架融合，这实现了框架对插件的可插拔的特性。  
  
面向切面编程的思想在Strut2中也有了很好的体现。最重要的体现就是拦截器的使用。拦截器就是一个一个的小功能单位，用户可以将这些拦截器合并成一个大的拦截器，这个合成的拦截器就像单独的拦截器一样，只要将它配置到一个、Action中就可以。  
  
Struts2 的缺点：  
在并发量比较大的场景中,.每次请求都要创建一个Action,并维护很长的调用链(至少18个拦截器+OGNL解析+Action+Result),资源消耗比较大.  
使用场景  
SSH对于中小型项目提供了一套完整的解决方案.在表关系相对简单,数据量不大,并发量不高的项目中,能够极大的提高开发效率.  
表关系复杂或数据量比较大时,可以使用Mybatis替换Hibernate.  
并发量很高时可以使用SpringMVC替换struts  
  
struts2的原理？  
  
  
  
  
  
  
  
  
  
  
  
在Struts2框架中的处理大概分为以下几个步骤   
1 客户端初始化一个指向Servlet容器（例如Tomcat）的请求   
2 这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做ActionContextCleanUp的可选过滤器，这个过滤器对于Struts2和其他框架的集成很有帮助，例如：SiteMesh Plugin）   
3 接着StrutsPrepareAndExecuteFilter被调用，StrutsPrepareAndExecuteFilter询问ActionMapper来解析和判断该次请求是否需要由struts2框架来处理.  
4 如果ActionMapper判断需要struts2来处理请求，StrutsPrepareAndExecuteFilter会把请求的处理交给ActionProxy   
5 ActionProxy通过Configuration Manager加载框架的配置文件，找到需要调用的Action以及拦截器配置信息  
6 ActionProxy创建一个ActionInvocation的实例。   
7 ActionInvocation实例使用命名模式来调用，在调用Action的过程前后，涉及到相关拦截器（Intercepter）的调用。   
8 一旦Action执行完毕，ActionInvocation负责根据struts.xml中的配置找到对应的返回结果配置。根据配置找到对应的Result处理类来处理结果集.大多数情况输出会交由模版语言(JSP,FreeMarker)完成输出内容拼装  
  
简单介绍一下你对hibernate的理解?  
  
    答：hibernate框架是一个ORM的持久层框架，ORM的含义是对象关系映射，简单理解就是通过对象和关系型数据库之间建立映射信息，以后再操作对象就相当于操作数据库了。hibernate框架是对JDBC进行了轻量级的封装，可以更方便简单的对数据库完成增删改查的操作。同时也提供了查询的方式和查询的策略。例如HQL和QBC的查询方式等。还提供了缓存的策略，效率会更高。  
  
Hibernate和mybatis有什么区别?  
  
1）mybatis是把sql语句与java代码分离了，sql语句在xml文件配置的  
2）hibernate是ORM框架,它对jdbc进行了封装,在分层结构中处于持久化层，它能建立面向对象的域模型和关系数据模型之间的映射.它大大简化了dao层的编码工作  
      3）  mybatis是半自动的，hibernate是全自动的，就是说mybatis可以配置sql语句，对于sql调优来说是比较好的，hibernate会自动生成所有的sql语句，调优不方便，hibernate用起来难度要大于mybatis  
  
Hibernate中的update()和saveOrUpdate()的区别  
    答：update是修改的方法，saveOrUpdate是保存或者更新  
        saveorupdate()如果传入的对象在数据库中有就做update操作，如果没有就做save操作。  
        save()在数据库中生成一条记录，如果数据库中有，会报错说有重复的记录。  
        update()就是更新数据库中的记录  
  
谈谈Spring的ioc、aop？  
        IoC  
                Inversion of Control 控制反转。  
                我们以前开发，在一个类中使用其他类对象的时候都是采用new的方式直接获取，或者高级一点是通过反射的方式的得到需要的对象实例。这就造成了程序的耦合度非常高，一个类的运行，严重依赖于其他的类。并且还会出现程序中硬编码的情况。  
                而spring中的IoC很好的解决了该问题，我们在一个类中使用其他类对象时，只需要定义一个接口类型的类成员变量，由使用者在使用时为我们注入具体的实现类对象，从而降低了程序的耦合度。  
                实现IoC的思想就只有两种：依赖注入（Dependency Injection，简称DI）和依赖查找（Dependency Lookup）。  
                而依赖注入使用的更广泛一些。例如：构造函数注入,set方法注入等等。  
  
        AOP  
                Aspect Oriented Programming 面向切面编程。  
                我们之前学习的java语言，号称是面向对象编程，它有自己的优势。但是也存在着一些弊端。  
                举例说明，在实际开发中，我们都会有一根业务主线，即客户（甲方）的需求。而程序员要做的就是围绕业务主线上的需求，实现功能（实现功能的方法我们叫做业务核心方法）。但是不可避免的，会有一些功能与业务主线没有关系，却又不能不做，比如权限的控制，事务的控制，日志的记录等等，这些功能绝大多数时候和业务主线没有关系，但是却和很多业务核心方法交织在一起，使我们的开发变得麻烦，并且冗余代码增多。  
                而spring的提供了一种思想，把这些和业务主线没有关系的功能剥离出来，而在需要使用这些公共方法时，适时适地的把它加到我们的代码中去，使程序员在开发时，把更多的精力放在理解需求，实现业务核心功能上，并且让我们的代码变得简洁。这种思想就是面向切面编程。  
                Spring实现面向切面编程使用的是动态代理技术，并且会根据实际情况来选择使用基于子类的还是基于接口的动态代理。  
  
简单介绍一下你对spring的理解?  
        任何框架在植入项目之中都不能带来效率的提升，反而是会带来效率的下降。因为java核心机制的问题，内存中多创建一个对象，就会造成性能降低。  
        但是，spring带来的好处还是显而易见的：  
                1、它的核心之一IoC，降低了我们程序的耦合度，使我们可以把项目设计成为一个可插拔的组件式工程。  
                2、它的另一大核心AOP，使我们在开发过程中，精力得到释放，可以更专注的去理解客户的需求。并且在后期维护时，可以只维护很少的一部分。  
                3、它提供的事务管理机制，采用声明的方式来配置事务，从而在维护时无需改动源码，解决了程序硬编码的弊端。  
                4、它提供的DAO模板使我们的持久层开发又多了一种途径。  
                5、它可以整合其他时下流行的框架，使我们在管理项目时，更加清晰，明确。  
        并且以现在计算机的水平，使用spring框架造成的这点下降对程序的影响是微乎其微的。  
        所以，总体来说spring框架的使用还是利大于弊的。  
  
Spring是干嘛的?  
        它是一个full-stack框架，提供了从表现层到业务层再到持久层的一套完整的解决方案。我们在项目中可以只使用spring一个框架，它就可以提供表现层的mvc框架，持久层的Dao框架。  
        它的两大核心IoC和AOP更是为我们程序解耦和代码简洁易维护提供了支持。  
  
你知道依赖注入么?简单介绍一下  
  
        首先，明确依赖注入是实现控制反转的一种思想（另一种是依赖查找）。  
        其次，在开发过程中，我们需要某个类的实例时，是由使用者为我们提供该类的实例。而不是自己去获取。  
        最后，实现依赖注入的方式可以是使用构造方法注入或者set方法注入两种方式。  
        在spring中注入的方式就有很多了，比如constructor-arg元素，property元素，p名称空间等等。  
  
你知道控制反转么?简单介绍一下  
        它是spring的核心之一。或者说是spring的基础核心，spring的其余核心功能都必须有IoC的支持。  
        控制反转指的是，我们在获取对象的时候，由之前的主动出击，变成了被动接收。也就是说，在编写某个类时，只需要提供一个接口类型的类成员，并不需要关系具体的实现类，而是由使用者在使用时提供。这就降低了类和类之间的耦合度。  
  
spring中自动装配的方式有哪些？  
        1、 No：  
                即不启用自动装配。  
        2、 byName：  
                通过属性的名字的方式查找JavaBean依赖的对象并为其注入。比如说类Computer有个属性printer，指定其autowire属性为byName后，Spring IoC容器会在配置文件中查找id/name属性为printer的bean，然后使用Seter方法为其注入。  
        3、 byType：  
                通过属性的类型查找JavaBean依赖的对象并为其注入。比如类Computer有个属性printer，类型为Printer，那么，指定其autowire属性为byType后，Spring IoC容器会查找Class属性为Printer的bean，使用Seter方法为其注入。  
        4、 constructor：  
                通byType一样，也是通过类型查找依赖对象。与byType的区别在于它不是使用Seter方法注入，而是使用构造子注入。  
        5、 autodetect：  
                在byType和constructor之间自动的选择注入方式。  
        6、 default：  
                由上级标签<beans>的default-autowire属性确定。  
  
spring和hibernate管理事务有啥区别  
        1、从编码上说，hibernate的事务管理是硬编码，是写在程序之中的。这就造成了，如果需要调整，就要修改源码，重新编译。  
        2、从事务控制的位置来说：        hibernate是持久层框架，事务是控制在持久层的，这样就造成了越权操作。事务应放在业务层，而非持久层  
        3、从代码维护上来说：hibernate控制事务，需要在每个需要事务支持的地方编写代码，后期维护不便。  
  
spring管理事务有几种方式  
        有两种方式：  
                1、编程式事务，在代码中硬编码。(不推荐使用)  
                 2、声明式事务，在配置文件中配置（推荐使用）  
                        声明式事务又分为两种：  
                                a、基于XML的声明式事务  
                                b、基于注解的声明式事务