# SSM框架

**作用**：

SSM框架是spring MVC ，spring和mybatis框架的整合，是标准的MVC模式，将整个系统划分为表现层，controller层，service层，DAO层四层

使用spring MVC负责请求的转发和视图管理

spring实现业务对象管理，mybatis作为数据对象的持久化引擎

原理：

SpringMVC：

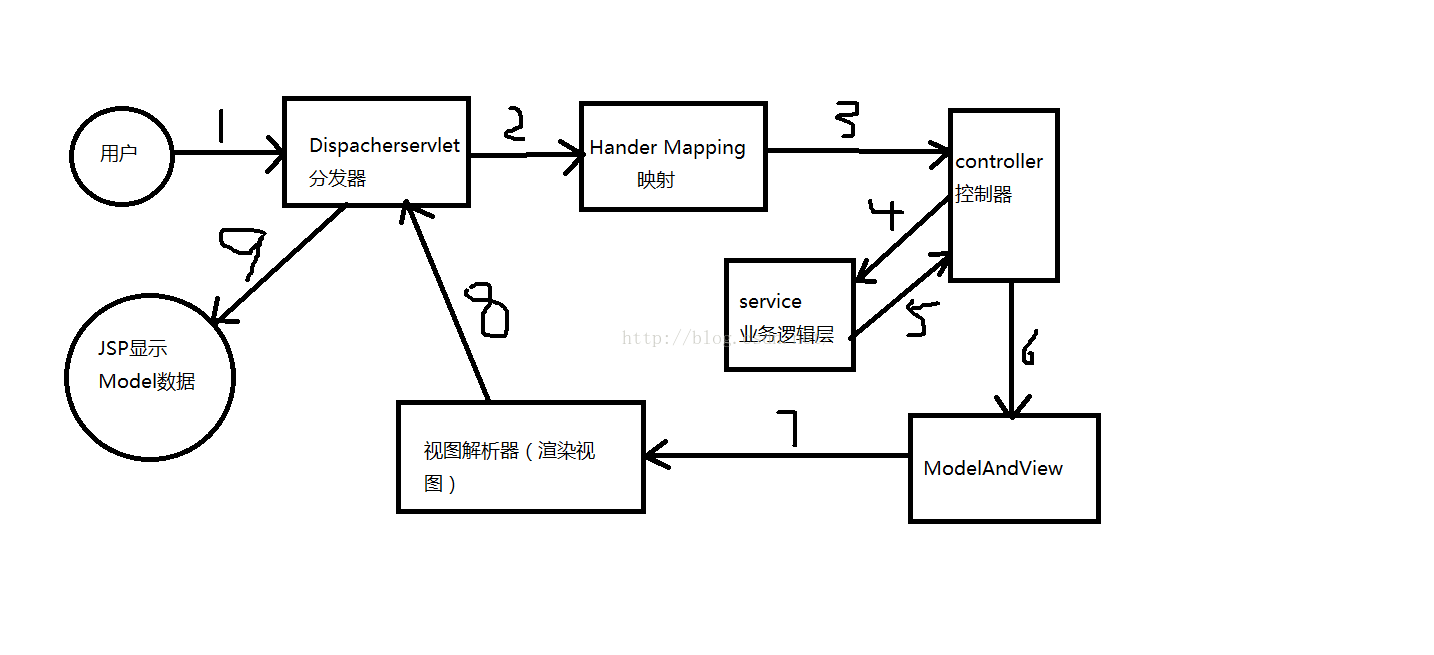
1.客户端发送请求到DispacherServlet（分发器）

2.由DispacherServlet控制器查询HanderMapping，找到处理请求的Controller

3.Controller调用业务逻辑处理后，返回ModelAndView

4.DispacherSerclet查询视图解析器，找到ModelAndView指定的视图

5.视图负责将结果显示到客户端



Spring：我们平时开发接触最多的估计就是IOC容器，它可以装载bean（也就是我们Java中的类，当然也包括service dao里面的），有了这个机制，我们就不用在每次使用这个类的时候为它初始化，很少看到关键字new。另外spring的aop，事务管理等等都是我们经常用到的。

Mybatis：mybatis是对jdbc的封装，它让数据库底层操作变的透明。mybatis的操作都是围绕一个sqlSessionFactory实例展开的。mybatis通过配置文件关联到各实体类的Mapper文件，Mapper文件中配置了每个类对数据库所需进行的sql语句映射。在每次与数据库交互时，通过sqlSessionFactory拿到一个sqlSession，再执行sql命令。

**使用方法：**

**要完成一个功能**：

先写实体类entity，定义对象的属性，（可以参照数据库中表的字段来设置，数据库的设计应该在所有编码开始之前）。

写Mapper.xml（Mybatis），其中定义你的功能，对应要对数据库进行的那些操作，比如 insert、selectAll、selectByKey、delete、update等。

写Mapper.java，将Mapper.xml中的操作按照id映射成Java函数。

写Service.java，为控制层提供服务，接受控制层的参数，完成相应的功能，并返回给控制层。

写Controller.java，连接页面请求和服务层，获取页面请求的参数，通过自动装配，映射不同的URL到相应的处理函数，并获取参数，对参数进行处理，之后传给服务层。

写JSP页面调用，请求哪些参数，需要获取什么数据。

DataBase ===> Entity ===> Mapper.xml ===> Mapper.Java ===> Service.java ===> Controller.java ===> Jsp.

Spring MVC  拥有控制器，作用跟Struts类似，接收外部请求，解析参数传给服务层

Spring 容器属于协调上下文，管理对象间的依赖，提供事务机制

mybatis 属于orm持久层框架，将业务实体 与数据表联合 起来

Spring MVC  控制层，想当与 Struts的作用

Spring 控制反转和依赖注入  创建对象交由容器管理，达到了解耦的作用

mybatis 主要用来操作数据库（数据库的增删改查）

**IOC**:控制反转，是一种降低对象之间耦合关系的设计思想，面试的时候最好能说出来个例子，加深理解。例子：租房子，以前租房子需要一个房子一个房子找，费时费力，然后现在加入一个房屋中介，把你需要的房型告诉中介，就可以直接选到需要的房子，中介就相当于spring容器。

AOP:面向切面编程，是面向对象开发的一种补充，它允许开发人员在不改变原来模型的基础上动态的修改模型以满足新的需求，如：动态的增加日志、安全或异常处理等。AOP使业务逻辑各部分间的耦合度降低，提高程序可重用性，提高开发效率。

**持久层：DAO层（mapper）**

DAO层：DAO层主要是做数据持久层的工作，负责与数据库进行联络的一些任务都封装在此，

DAO层的设计首先是设计DAO的接口，

然后在Spring的配置文件中定义此接口的实现类，

然后就可在模块中调用此接口来进行数据业务的处理，而不用关心此接口的具体实现类是哪个类，显得结构非常清晰，

DAO层的数据源配置，以及有关数据库连接的参数都在Spring的配置文件中进行配置。

**业务层：Service层**

Service层：Service层主要负责业务模块的逻辑应用设计。

首先设计接口，再设计其实现的类

接着再在Spring的配置文件中配置其实现的关联。这样我们就可以在应用中调用Service接口来进行业务处理。

Service层的业务实现，具体要调用到已定义的DAO层的接口，

封装Service层的业务逻辑有利于通用的业务逻辑的独立性和重复利用性，程序显得非常简洁。

**表现层：Controller层（Handler层）**

Controller层:Controller层负责具体的业务模块流程的控制，

在此层里面要调用Service层的接口来控制业务流程，

控制的配置也同样是在Spring的配置文件里面进行，针对具体的业务流程，会有不同的控制器，我们具体的设计过程中可以将流程进行抽象归纳，设计出可以重复利用的子单元流程模块，这样不仅使程序结构变得清晰，也大大减少了代码量。

**View层**

View层 此层与控制层结合比较紧密，需要二者结合起来协同工发。View层主要负责前台jsp页面的表示.

**各层联系**

DAO层，Service层这两个层次都可以单独开发，互相的耦合度很低，完全可以独立进行，这样的一种模式在开发大项目的过程中尤其有优势

Controller，View层因为耦合度比较高，因而要结合在一起开发，但是也可以看作一个整体独立于前两个层进行开发。这样，在层与层之前我们只需要知道接口的定义，调用接口即可完成所需要的逻辑单元应用，一切显得非常清晰简单。

**Service逻辑层设计**

Service层是建立在DAO层之上的，建立了DAO层后才可以建立Service层，而Service层又是在Controller层之下的，因而Service层应该既调用DAO层的接口，又要提供接口给Controller层的类来进行调用，它刚好处于一个中间层的位置。每个模型都有一个Service接口，每个接口分别封装各自的业务处理方法。

SSH框架

SSH是 struts+spring+hibernate的一个集成[框架](https://baike.so.com/doc/1863840-1971314.html" \t "https://baike.so.com/doc/_blank)，是目前比较流行的一种Web应用程序开源框架。区别于 Secure Shell 。

集成SSH框架的系统从职责上分为四层:[表示层](https://baike.so.com/doc/5331330-5566567.html" \t "https://baike.so.com/doc/_blank)、[业务逻辑层](https://baike.so.com/doc/67632-71373.html" \t "https://baike.so.com/doc/_blank)、[数据持久层](https://baike.so.com/doc/1432728-1514442.html" \t "https://baike.so.com/doc/_blank)和域模块层，以帮助开发人员在短期内搭建结构清晰、可复用性好、[维护](https://baike.so.com/doc/257070-272132.html" \t "https://baike.so.com/doc/_blank)方便的[Web应用程序](https://baike.so.com/doc/5507210-5742956.html" \t "https://baike.so.com/doc/_blank)。其中使用Struts作为系统的整体基础架构，负责MVC的分离，在Struts框架的[模型](https://baike.so.com/doc/5408348-5646328.html" \t "https://baike.so.com/doc/_blank)部分，控制业务跳转，利用Hibernate框架对[持久层](https://baike.so.com/doc/7722506-7996601.html" \t "https://baike.so.com/doc/_blank)提供支持，Spring做管理，管理struts和hibernate。具体做法是:用[面向对象](https://baike.so.com/doc/6146835-6360018.html" \t "https://baike.so.com/doc/_blank)的分析方法根据需求提出一些模型，将这些模型实现为基本的Java对象，然后编写基本的DAO(Data Access Objects)接口，并给出Hibernate的DAO实现，采用Hibernate架构实现的DAO类来实现[Java](https://baike.so.com/doc/2886868-3046592.html" \t "https://baike.so.com/doc/_blank)类与数据库之间的转换和访问，最后由Spring做[管理](https://baike.so.com/doc/1462322-1546098.html" \t "https://baike.so.com/doc/_blank)，管理struts和hibernate。

业务流程

[系统](https://baike.so.com/doc/5402991-5640678.html" \t "https://baike.so.com/doc/_blank)的基本[业务](https://baike.so.com/doc/5411680-5649795.html" \t "https://baike.so.com/doc/_blank)流程是: 在[表示层](https://baike.so.com/doc/5331330-5566567.html" \t "https://baike.so.com/doc/_blank)中，首先通过JSP页面实现[交互界面](https://baike.so.com/doc/6959642-7182153.html" \t "https://baike.so.com/doc/_blank)，负责接收请求(Request)和传送响应(Response)，然后Struts根据[配置](https://baike.so.com/doc/6788938-7005547.html" \t "https://baike.so.com/doc/_blank)文件(struts-config.xml)将ActionServlet接收到的Request委派给相应的Action处理。在业务层中，管理服务组件的Spring IoC容器负责向Action提供业务模型(Model)组件和该组件的协作对象数据处理(DAO)组件完成业务逻辑，并提供事务处理、缓冲池等容器组件以提升系统性能和保证数据的完整性。而在[持久层](https://baike.so.com/doc/7722506-7996601.html" \t "https://baike.so.com/doc/_blank)中，则依赖于Hibernate的对象化映射和数据库交互，处理DAO组件请求的数据，并返回处理结果。

采用上述开发模型，不仅实现了视图、控制器与模型的彻底分离，而且还实现了[业务逻辑层](https://baike.so.com/doc/67632-71373.html" \t "https://baike.so.com/doc/_blank)与持久层的分离。这样无论前端如何变化，模型层只需很少的改动，并且数据库的变化也不会对前端有所影响，大大提高了系统的可复用性。而且由于不同层之间[耦合度](https://baike.so.com/doc/1051405-1112171.html" \t "https://baike.so.com/doc/_blank)小，有利于团队成员并行工作，大大提高了开发[效率](https://baike.so.com/doc/5173394-5404182.html" \t "https://baike.so.com/doc/_blank)。

框架结构

**Struts**

Struts对Model，View和Controller都提供了对应的组件。

ActionServlet，这个类是Struts1的核心控制器，负责拦截来自用户的请求。

Action，这个类通常由用户提供，该控制器负责接收来自ActionServlet的请求，并根据该请求调用模型的业务逻辑方法处理请求，并将处理结果返回给JSP页面显示。

**Model部分:**

由ActionForm和JavaBean[组成](https://baike.so.com/doc/1534347-1622069.html" \t "https://baike.so.com/doc/_blank)，其中ActionForm用于将用户的请求参数封装成ActionForm对象，该对象被ActionServlet转发给Action，Action根据ActionForm里面的请求参数处理用户的请求。

JavaBean则封装了底层的业务逻辑，包括数据库访问等。

**View部分:**

该部分采用JSP(或HTML、PHP……)实现。

Struts提供了丰富的[标签库](https://baike.so.com/doc/5997574-6210547.html" \t "https://baike.so.com/doc/_blank)，通过标签库可以减少脚本的使用，自定义的标签库可以实现与Model的有效交互，并增加了现实功能。对应上图的JSP部分。

**Controller组件:**

Controller组件有两个部分组成--系统核心控制器，业务逻辑控制器。

系统核心控制器，对应上边的ActionServlet。该控制器继承了HttpServlet类，因此可以配置成标注的Servlet。该控制器负责拦截所有的[HTTP请求](https://baike.so.com/doc/1535748-1623556.html" \t "https://baike.so.com/doc/_blank)，然后根据用户请求决定是否要转给业务逻辑控制器。

业务逻辑控制器，负责处理用户请求，本身不具备处理能力，而是调用Model来完成处理。对应Action部分。

**Spring**

Spring是一个开源框架，它由Rod Johnson创建。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、[可测试性](https://baike.so.com/doc/1917107-2028288.html" \t "https://baike.so.com/doc/_blank)和[松耦合](https://baike.so.com/doc/626939-663595.html" \t "https://baike.so.com/doc/_blank)的角度而言，任何Java应用都可以从Spring中受益。

目的:解决企业应用开发的复杂性

功能:使用基本的JavaBean代替EJB，并提供了更多的企业应用功能

范围:任何Java应用

简单来说，Spring是一个轻量级的[控制反转](https://baike.so.com/doc/868703-918468.html" \t "https://baike.so.com/doc/_blank)(IoC)和面向切面(AOP)的容器框架。

轻量--从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的:典型地，Spring应用中的对象不依赖于Spring的特定类。

[控制反转](https://baike.so.com/doc/868703-918468.html" \t "https://baike.so.com/doc/_blank)--Spring通过一种称作控制反转(IoC)的技术促进了[松耦合](https://baike.so.com/doc/626939-663595.html" \t "https://baike.so.com/doc/_blank)。当应用了IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为IoC与JNDI相反--不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

面向切面--Spring提供了[面向切面编程](https://baike.so.com/doc/7533542-7807635.html" \t "https://baike.so.com/doc/_blank)的丰富支持，允许通过分离应用的业务逻辑与系统级服务(例如审计(auditing)和事务(transaction)管理)进行[内聚性](https://baike.so.com/doc/6480168-6693871.html" \t "https://baike.so.com/doc/_blank)的开发。[应用对象](https://baike.so.com/doc/312399-330772.html" \t "https://baike.so.com/doc/_blank)只实现它们应该做的--完成业务逻辑--仅此而已。它们并不负责(甚至是意识)其它的系统级关注点，例如日志或事务支持。

容器--Spring包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，你可以配置你的每个bean如何被创建--基于一个可配置原型(prototype)，你的bean可以创建一个单独的实例或者每次需要时都生成一个新的实例--以及它们是如何相互关联的。然而，Spring不应该被混同于传统的重量级的EJB容器，它们经常是庞大与笨重的，难以使用。

框架--Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，[应用对象](https://baike.so.com/doc/312399-330772.html" \t "https://baike.so.com/doc/_blank)被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能(事务管理、持久化框架集成等等)，将应用逻辑的开发留给了你。

所有Spring的这些特征使你能够编写更干净、更可管理、并且更易于测试的代码。它们也为Spring中的各种模块提供了基础支持。

**Hibernate**

Hibernate是一个开放源代码的[对象关系映射](https://baike.so.com/doc/1334221-1410580.html" \t "https://baike.so.com/doc/_blank)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可以应用在任何场合使用JDBC，既可以在Java的客户端程序使用，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](https://baike.so.com/doc/5030500-5256785.html" \t "https://baike.so.com/doc/_blank)的重任。

Hibernate的核心接口一共有5个，分别为:Session、SessionFactory、Transaction、Query和Configuration。这5个核心接口在任何开发中都会用到。通过这些接口，不仅可以对持久化对象进行存取，还能够进行事务控制。下面对这五个核心接口分别加以介绍。

·Session接口:Session接口负责执行被持久化对象的CRUD操作(CRUD的任务是完成与数据库的交流，包含了很多常见的SQL语句。)。但需要注意的是[Session对象](https://baike.so.com/doc/2245462-2375809.html" \t "https://baike.so.com/doc/_blank)是非[线程安全](https://baike.so.com/doc/6809842-7026796.html" \t "https://baike.so.com/doc/_blank)的。同时，Hibernate的session不同于JSP应用中的HttpSession。这里当使用session这个术语时，其实指的是Hibernate中的session，而以后会将HttpSession对象称为用户session。

·SessionFactory接口:SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建Session对象。这里用到了工厂模式。需要注意的是SessionFactory并不是轻量级的，因为一般情况下，一个项目通常只需要一个SessionFactory就够，当需要操作多个数据库时，可以为每个数据库指定一个SessionFactory。

·Configuration接口:Configuration接口负责配置并启动Hibernate，创建SessionFactory对象。在Hibernate的启动的过程中，Configuration类的实例首先定位映射文档位置、读取配置，然后创建SessionFactory对象。

·Transaction接口:Transaction接口负责事务相关的操作。它是可选的，开发人员也可以设计编写自己的底层事务处理代码。

·Query和Criteria接口:Query和Criteria接口负责执行各种数据库查询。它可以使用HQL语言或SQL语句两种表达方式。

SSH框架和SSM框架的区别

SSH： 通常指的是 Struts2 做控制器(controller)，spring 管理各层的组件，hibernate 负责持久化层。

SSM： 则指的是 SpringMVC 做控制器(controller)，Spring 管理各层的组件，MyBatis 负责持久化层。

**共同点：**1.Spring依赖注入DI来管理各层的组件。

2.使用面向切面编程AOP管理事物、日志、权限等。

**不同点：**1.Struts2 和 SpringMVC 控制器(controller)控制视图和模型的交互机制的不同，

Struts2是Action类级别，SpringMVC是方法级别，更容易实现RESTful风格。

总结：SSM和SSH不同主要在MVC实现方式，以及ORM持久化方面不同（Hiibernate与Mybatis）。SSM越来越轻量级配置，将注解开发发挥到极致，且ORM实现更加灵活，SQL优化更简便；而SSH较注重配置开发，其中的Hiibernate对JDBC的完整封装更面向对象，对增删改查的数据维护更自动化，但SQL优化方面较弱，且入门门槛稍高。