基于SSH2框架的Web系统的设计与实现

何 信 杜 江 庞海艳

摘 要：在Web系统开发中，针对Spring、Struts2、Hibernate各自架构体系的特点和工作原理，提出了将三大框架整合成一个SSH2的多层架构的方案，阐述了Spring、Struts2和Hibernate整合在b/s结构开发中的优势，降低各层之间的耦合度。最后通过给出一个SSH2组合框架的应用实例，进而提高系统的开发效率、可移植性和可扩展性，降低系统的开发成本。

关键词：Spring Strut2 J2EE Hibernate Web 系统设计与分析

基金项目：本文系教育部人文社会科学研究规划基金项目资助“关联数据环境下安全机制与数据溯源的研究”，项目编号：12YJA870014

中图分类号：TP399 文献标识码：A

DOI：10.3969/j.issn.1674-537X.2015.01.017

0 引 言

随着Internet的日益普及，Web开发的复杂性不断增加。构建复杂灵活而又可靠的Web系统不仅提升企业的网络办公环境，还可以扩展营销渠道、增强与客户沟通联系。Spring、Struts2、Hibernate已成为企业开发Web系统的主流框架。Spring致力于J2EE各层的解决方案，包括表现层、业务层、数据持久层。Struts2、Hibernate是处理表现层、业务层、数据持久层比较成熟的架构体系。三种技术的结合使用，实现了层与层之间设计的松耦合，层内设计的高内聚，这有利于实现系统的维护性、拓展性和稳定性。

1 研究目的

J2EE(Java Platform Enterprise Edition)是标准的企业版Java应用平台，为开发企业应用系统提供了一整套的技术标准组件。但是J2EE结构庞大，不易上手，对于绝大多数

现在的思维模式，此问题可能是一个没有答案的问题。

（三）究竟通过什么方式可以让民众接受这类工程？

事实上，核电站项目、PX项目、垃圾处理项目等等，之所以受到项目周边居民的反对，甚至抗议除了这些项目本身可能对环境安全有一定的负面影响外，更多地与社会舆论对这些项目的污名化、人们对这些项目的真实性质不理解，以及项目的安全文化建设滞后等有关。比如网络上掀起的px真相大探讨。此外还有利益补偿问题，如果我们的政府和企业不能充分考虑周边民众的利益，要这些民众接受这些项目也是非常困难的。

（四）如何走出立项－抗议－停止的怪圈？

国际经验表明，要完全不开工建设邻避项目简直就是不可能的。但目前每当一个邻避项目开工时，总可能有人要闹，但一闹就停，也不是长久之计。为了长远发展，我们必须走出目前这样“立项—抗议—停止”的状况，不能让这种现实困境反复出现，以至于形成一种常态，甚至演变为一种模式，这就要认真研究如何防治此类邻避事件的问题。

三、邻避型事件防范与应对的政府治理路径

（一）将“稳评”嵌入政府决策过程，实现事前主动的风险预防和矛盾化解

重大事项社会稳定风险评估机制（简称“稳评”）是一项从源头上化解社会稳定风险和推动传统维稳模式转型的重要制度安排，强调维稳的关口前移和风险的源头治理。切实的稳评工作能够把握风险源头，化解社会矛盾，达成社会共识，从而推动建设事业的“积极机制”。因此，当务之急就是将一个由科学与民主进行双轮驱动的“稳评”机制嵌入到各级政府的决策过程中去，以更积极主动的姿态去化解矛盾进而创造社会稳定。

（二）规范政府治理的主体地位

地方政府与企业之间有着一定的共同利益，这种共同利益的存在导致地方政府对邻避设施的建立可能仅专注于其所带来的社会经济效益，而忽视了部分公众的利益，规避邻避事件，需要规范政府治理主体的中立地位，明确政府的第一要务不是发展经济，而是维护公平正义，不能把政府和企业捆绑在一起，站在群众的对立面上。

（三）建立完善的邻避补偿机制

对已经发生或可能发生的损失进行评估、赔偿，减轻设施周边居民的损失，以降低居民的抗争。通过金钱与非金钱二大类进行补偿，对于邻避型群体性事件中的损失易计算的部分较为有效。建立完善的补偿机制，可以很好的修复邻避设施的负外部性。

（四）进一步加强风险沟通与公众参与

有效的风险沟通有利于增强不同利益相关者对风险的可容忍性、弥合公众与政府间的风险感知差异。规避邻避事件，需要畅通群众诉求渠道，完善“稳评”机制，把有无良好的风险沟通机制作为重大工程项目稳评的重要内容，将吸收公众参与“稳评”的机制进一步制度化和程序化。

参考文献：

[1] 童星，张海波．群体性突发事件及其治理——社会风险与公共危机综合分析框架下的再考量．学术界，2008.2.

[2] 乌尔里希·贝克．风险社会．译林出版社，2003.

[3] 陶鹏，童星．邻避型群体性事件及其治理．南京社会科学，2010.8.

（作者单位：河北行政学院信息化教研部）

开发人员都是望而却步。在技术的不断演进的过程中，Web的应用形成了鲜明的层次，在各层次上基于不同的技术产生了成熟的架构体系，如：业务层的Spring框架体系、控制层的Struts框架体系以及数据持久层的Hibernate框架体系。如何整合利用这些成熟的框架，既能发挥各个架构的优势，又能实现系统的“低耦合、高内聚”，成为本文讨论的重点问题。

2 SS2H技术架构

2.1 JSP技术

JavaServer Page(JSP)是一种动态的网页技术标准，它允许脚本语言嵌入到HTML/XML文档中，动态生成HTML文档。JSP脚本语言可以是嵌入在JSP页面中的Java代码段或表达式，它在服务器端被解析，随后被转化为Servlet进行处理，生成动态的页面数据并传送到客户端显示。

在基于MVC(Model-View-Controller)[1-2]模式中，JSP位于View层，用来显示用户的界面。首先JSP将用户的请求通知控制器（Controller），由控制器对视图的请求进行处理，组织用户需要的模型数据。然后JSP从模型层获取并表现用户需要的数据，当模型层发生变化时，JSP负责维持数据表现的一致性。MVC设计模式如图1所示。

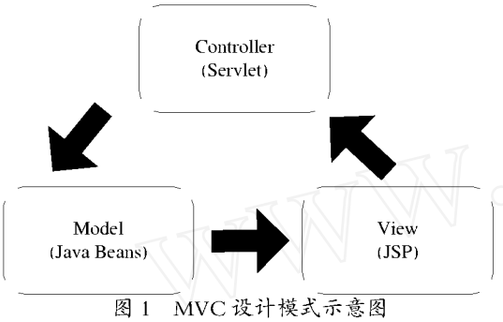

图1 MVC设计模式示意图

2.2 Spring核心技术

面向切面编程(AOP, Aspect Oriented Programming)[3-4]可是通过预编译的方式和运行期动态代理实现程序功能的统一维护的一种技术。降低业务逻辑各部分的耦合度。在实际的项目开发中的软件要考虑到日志记录、性能统计、安全控制、事务处理、异常处理等，这些功能不属于系统业务逻辑模块，却分布在系统业务逻辑的各个部分，这种行为被称为“横切关注点”。AOP提供一种Aspect方面，封装了“横切关注点”的实现代码，并为系统主要业务类的方法提供了切入服务。

控制反转(IOC, Inversion of Control)[5]是遵循了面向对象设计原则的一种设计模式。系统模块之间存在着各种各样的依赖关系，过强的耦合会导致软件的结构复杂，维护困难。而控制反转就是切断服务类对其依赖的接口具体实现类的直接联系，服务类不直接创建依赖的具体对象，而是由框架来注入，从而降低模块间的依赖关系，提高系统的可维护性。

2.3 Struts2核心技术

Struts2[6]是Apache软件基金会（Apache Software Foundation，简称ASF）下的一个开源项目架构，主要是在struts1.x原有的基础上，借助于WebWork的技术而发展形成的一个全新的框架。Struts2的总体结构图如图2所示。

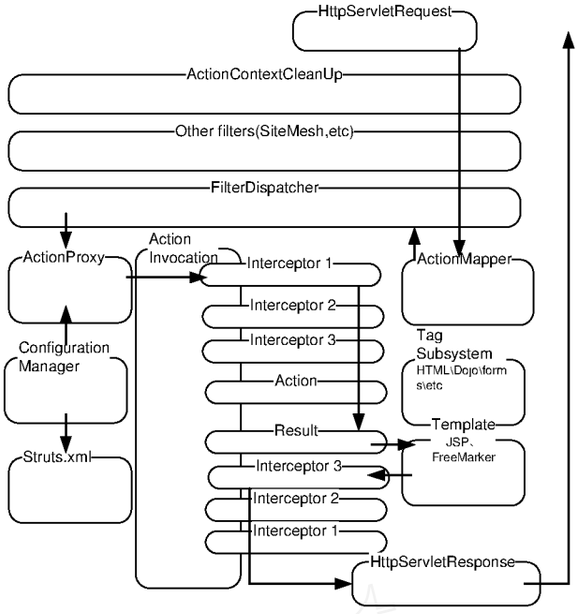

图2 Struts2工作原理

Struts2的工作原理：

客户端浏览器发出请求被封装成一个HttpServletResuest对象，通过过滤器ActionContextCleanUp、FilterDispatcher，对请求的信息进行过滤。

过滤器调度类FilterDispatcher询问动作映射类ActionMapper是否进行action处理请求，并负责将请求处理提交给动作代理类ActionProxy。

ActionProxy通过Configuration Manager查找配置文件struts.xml，并且找到要调用的Action。

ActionProxy创建一个ActionInvocation对象对Action执行调度。ActionInvocation对Action的请求调度被包装在一系列的拦截器Interceptor内部。Action处理完成之后，ActionInvocation根据struts.xml的配置找到对应的返回结果。一般对应的是JSP页面。

2.4 Hibernate核心技术

Hibernate是一个开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程序员可以使用对象编程思维来操纵数据库。Hibernate的体系结构如图3所示。

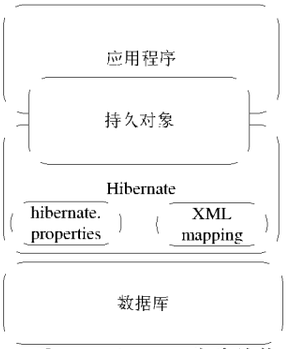

图3 Hibernate体系结构

Hibernate的工作原理：首先要初始化Hibernate，读取

Hibernate 的配置信息，配置信息包括：指定的数据源等，创建 SessionFactory 实例，然后调用 SessionFactory 实例创建 Session 对象。由 Session 负责执行被持久化对象的 CRUD（增加 Create、查询 Retrieve、更新 Update、删除 Delete）操作。

3 SS2H 技术在 Web 系统中的实现

将通过一个用户登录的实例来展现 SS2H 技术在 Web 系统中应用的具体设计和实现[7-9]。

3.1 Web.xml 文件的配置

Web.xml 作为程序启动的配置文件，对 SS2H 框架技术做了初始化配置工作。contextConfigLocation 参数定义了装载 Spring 的配置文件。Spring 的监听器类 ContextLoaderListener、负责读取 Xml 配置文件中的配置信息并产生 WebApplicationContext 对象，由这个对象访问 Spring 容器中 Bean。

```
<context-param>
  <param-name>contextConfigLocation</param-name>
<param-value>classpath:/spring-config/
applicationContext-*.xml</param-value>
</context-param>
<listener>
<listener-class>org.springframework.web.context.
ContextLoaderListener</listener-class>
 </listener>
```

Struts2 的过滤器 ActionContextCleanUp、FilterDispatcher 过滤器一起工作，对于用户提交的一些信息要进行过滤，例如过滤掉非法的 URL 请求，或者在传入 Servlet 或 Struts 的 action 前统一设置字符集。

```
<filter>
   <filter-name>struts-cleanup</filter-name>
   <filter-class>
  org.apache.struts2.dispatcher.ActionContextCleanUp
   </filter-class>
  </filter>
  <filter-mapping>
   <filter-name>struts-cleanup</filter-name>
 <url-pattern>/*</url-pattern>
</filter-mapping>
<filter>
     <filter-name>struts2</filter-name>
     <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
     </filter-class>
</filter>
<filter-mapping>
     <filter-name>struts2</filter-name>
     <url-pattern>/*</url-pattern>
</filter-mapping>
```

OpenSessionInViewFilter 是 Spring 提供的针对 Hibernate 的一个支持类，主要指在发起一个页面请求时打开 Hibernate 的 Session，而且一致保持这个 Session。以下是在 web.xml 中配置 OpenSessionInViewFilter 类的代码：

```
<filter>
   <filter-name>lazyLoadingFilter</filter-name>
     <filter-class>  org.springframework.orm.hibernate3.
support.OpenSessionInViewFilter
     </filter-class>
 </filter>
 <filter-mapping>
    <filter-name>lazyLoadingFilter</filter-name>
    <url-pattern>*.action</url-pattern>
</filter-mapping>
```

3.2 Struts.xml 文件的配置

设置作用于请求的字符集编码。

```
<constant name="struts.i18n.encoding" value="utf-8"/>
```

设置 struts 处理请求后缀，需要多个后缀时用英文逗号隔开。

```
<constant name="struts.action.extension"
value="action"/>
```

设置 struts 于 Spring 集成时，指定由 spring 负责 action 对象的创建。

```
<constant name="struts.objectFactory" value="spring"/>
```

设置 action 事件，用匹配符 * 匹配前缀路径的请求，能准确地在 Action 类中定位匹配的方法，减少性能损耗。class="classBean" 是请求对应的类，与 Spring 中 bean 标签的 id 对应。<result> 中的 name="indexPage" 值对应的是 Action 类中的 return 值，<result>/index.jsp</result> 就是要跳转的页面。

```
<action name="loginUserAction"  calss="userAction"
method="{1}">
     <result name="indexPage">/index.jsp</result>
<result name="errorPage">/error.jsp</result>
</action>
```

3.3 Hibernate 文件的配置

Hibernate 的基本配置文件有两种：Hibernate.cfg.xml 文件与 .hbm.xml 文件。.hbm.xml 是 POJO 对象与数据库中的表进行连接的配置文件。使对象与数据库中的表进行映射，对象的成员属性与表中的字段进行映射。

数据库中存在 user 表和 log 表，创建表的语法为：

```
Create table user(
     userId integer auto_Increment primary key,
     userName varchar(20) comment '用户名称',
     userPassword varchar(20) comment '用户密码'
)
Create table log(
logId integer auto_Increment primary key,
     logName varchar(50) comment '日志名称',
     logDate Date comment '日志日期'
)
```

user 表和 log 表的对应的 JavaBeans 对象：

```
Public class User{
     Private Integer userId;
     Private String userName;
     Private String userPassword;
     …… // 省略 getter、setter
}
Public class log{
     Private Integer logId;
```

```
    Private String logName;
    Private Date logDate;
    …… // 省略 getter、setter
}
```

User.hbm.xml 是映射 user 表与 User 对象的配置文件，Log.hbm.xml 是映射 Log 表与 log 对象的配置文件，其中 class 标签中的 name 属性设置为 JavaBean 类，table 属性指定为数据库对应的表名称，catalog 是相应的数据库，标签 id 为 Hibernate 的映射文件中主键字段映射，property 为其它字段的映射。映射文件 User.hbm.xml 内容如下。

```
<hibernate-mapping>
<class name="User"table="user"
catalog="databasename">
<id name="userId"type="java.lang.integer"
<column name="userId"/>
<generator class="increment"></generator>
</id>
<property name="userName" type="java.lang.String">
<column name ="userName" length="20"/>
</property>
……
</class>
</hibernate-mapping>
```

映射文件 Log.hbm.xml 内容如下。

```
<hibernate-mapping>
<class name="Log"table="log"
catalog="databasename">
<id name="logId" type="java.lang.integer"
<column name="logId"/>
<generator class="increment"></generator>
</id>
<property name="logName" type="java.lang.String">
<column name ="logName" length="50"/>
</property>
…….
</class>
</hibernate-mapping>
```

Hibernate.cfg.xml 文件中包含了 Hibernate 与数据库的基本连接信息，包括数据库方言、提供 JDBC 连接数据库时的协议，子协议和数据源标识、加载数据库驱动、以及设置全部的 .hbm.xml 配置文件的路径。在 Hibernate 工作的初始阶段，这些信息被先加载到 Configuration 和 SessionFactory 的实例中。

```
<hibernate-configuration>
<session-factory>
    <property name="dialect">
      org.hibernate.dalect.MySQLDialect
    </property>
    <property name="connection.url">
            jdbc:mysql://localhost:3306/databasename
</property>
<property name="connection.username">root</property>
<property name="connection.password">123456</property>
<property name="connection.driver_class">
        com.mysql.jdbc.Driver
</property>
<mapping resource="User.hbm.xml"/>
<mapping resource="Log.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

3.4 applicationContext.xml 文件配置

配置文件 applicationContext.xml 是 Spring 容器的入口。该文件将控制层、服务层、数据持久层的接口实例 Bean 进行封装，并且配置 Spring 容器的事务处理。

```
<bean id="sessionFactory"
class="org.springframework.orm.hibernate4.
LocalSessionFactoryBean">
<property
name="configLocation"value="classpath:hibernate.cfg.
xml"/>
<propery
name="configurationClass"value="org.hibernate.c
fg.AnnotationConfiguration"/>
</bean>
```

Spring 配置文件中关于事务配置有三部分组成：分别是 DataSource、TransactionManager 和代理机制三部分。

```
<beanid="transactionManager"class="org.
springframework.orm.
Hibernate4.HibernateTransactionManager"/>
<bean id="transactionInterceptor"class="org.
springframeword.transaction.interceptor.TransactionIntercept
or"p:transactionManager-ref="transactionManager">
<property name="transactionAttributes">
<props>
<prop key="get*">PROPAGATION_REQUIRED,
readOnly</prop>
<prop
key="find*">PROPAGATION_REQUIRED,readOnly</
prop>
                    …….
</props>
</property>
</bean>

<bean id="ProxyCreator"class="org.springframework.aop.
framework.autoproxy.BeanNameAutoProxyCreator"p:beanNames="
*Service,*ServiceImpl"p:interceptorNames="transactionInIntercep
tor"/>
                <!-- user DAO 层 -->
        <bean id="userDAO"class="dao.impl.
    UserDAOImpl"scope="singleton">
            <property name="sessionFactory">
                    <ref bean="sessionFactory" />
                </property>
    </bean>
    <!-- user Service 层 -->
    <bean id="userService" class="service.impl.
```

```
UserServiceImpl" scope="singleton">
                <property name="userDAO">
                        <ref bean="userDAO" />
                </property>
 </bean>
<!-- user Action 层 -->
        <bean id="userAction" class="action.UserAction"
           scope="prototype" >
          <property name="userService">
                  <ref bean="userService" />
           </property>
</bean>

          <!--log DAO 层 -->
          <bean id=" log DAO" class="dao.impl.LogDAOImpl"
scope="singleton">
            <property name="sessionFactory">
                        <ref bean="sessionFactory" />
                </property>
</bean>
<!-- log Service 层 -->
<bean id="logService" class="service.impl.LogServiceImpl"
scope="singleton">
                <property name="logDAO">
                        <ref bean="logDAO" />
                </property>
 </bean>
<!-- log Action 层 -->
        <bean id="logAction" class="action.LogAction"
           scope="prototype" >
          <property name="logService">
                  <ref bean="logService" />
           </property>
</bean>
```

3.5 applicationContext.xml 文件配置

用户身份验证控制层 Action 类代码：

```
public class UserAction extends ActionSupport{
      private UserServiceInterface userService;
      ……// 省略服务层 setter
      public String login(){
        String userName = request.getParameter("userName");
        String userPassword = request.
getParameter("userPassward");
        UserForm userForm=this.userService.
checkLogin(userName);
        if(userForm.userPassword.equals(userPassword))
                return "indexPage";
        else
 return "errorPage";
}
}
```

用户身份验证服务层 Service 类代码：

```
public class UserServiceImpl implements
UserServiceInterface{
      private UserDAOInterface userDAO;
      ……// 省略 Dao 层 setter
      public UserForm checkLogin(String userName){
 return this. userDAO.checkLogin(userName);
}
}
```

用户身份验证持久层 Dao 类代码：

```
public class UserDAOImpl implements UserDAOInterface{
      private SessionFactory sessionFactory;
      …..// 省略 sessionFactory setter
      public User checkLogin(String userName){
      String sql ="from User u where
u.userName="+username;
      ……// 省略 Hibernate 查询用户语句
      return user;
}
}
```

4 结束语

目前 SS2H 开源框架已成为企业应用开发中使用最多的开源框架。它的松耦合的设计理念；基于注射式的依赖模式；非分布式和协同性等特点使得软件开发变得简单而便捷，这使得本文提出的基于 SS2H 框架的 Web 系统设计与实现方法在软件开发中具有一定的指导意义。

参考文献：

[1] 李海峰 .MVC 模式架构的应用研究 [J]. 自动化与仪表仪器 ,2013(2):4-7.

[2] 冯向阳、冯飞飞、苏厚勤 .MVC 软件构架在城市安全生产监管系统中的设计和应用 [J]. 计算机应用与软件 ,2013,30(4):192-207.

[3] 袁绪峰 . 基于 Spring 框架的 AOP 编程 [J]. 计算机与现代化 ,2006,(1):118-120.

[4] 梁琳、许向众、洪超 Spring 框架与 AOP 思想的研究与应用 [J]. 计算机信息技术 ,2006(4):24-26.

[5] 薄奇、许林英 .Spring 框架中的 IoC 的实现 [J]. 微处理机 ,2008(1):147-153.

[6] 刘艳春、洪晓慧 .Struts2 框架核心配置文件的研究与应用 [J]. 计算机技术与发展 ,2013,23(2):78-81.

[7] 邓子云、罗涛、黄友森、潘果、钟静 . 基于 Struts2+Hibernate3+Spring2 的物流数据交换平台 [J]. 计算机应用与软件 ,2009,26(10):88-100.

[8] 于东超 . 基于 Struts2_Spring_Hibernate 三种框架的通用 Web 框架的研究及应用 [D]. 大连交通大学 ,2008.

[9] 王雪 . 基于 Struts-Spring-Hibernate 框架的企业培训管理系统的设计与实现 [D]. 吉林大学软件学院 ,2012.

（作者单位：北京石油化工学院经济管理学院
冀东油田能源公司）