# SSH框架的体验实践

13302010076 王思杰

## 1 概述

本次实验通过在SSH框架上实现一个留言板web项目，学习SSH的框架结构和业务流程。并将项目部署到了VPS上（http://45.78.59.29:8080/Demo/）

## 2 ssh框架

SSH 是 struts+spring+hibernate集成框架的总称。下面对各个框架进行简要介绍。

Struts

过去我们常用JSP、Servlet、JavaBean编写web应用，这些技术的构建比较复杂烦乱，Struts框架提供了把这些技术组织起来的规则。
Struts2基本流程：1、客户端浏览器发出HTTP请求。2、根据web.xml配置，该请求被FilterDispatcher接收。3、根据struts.xml配置，找到需要调用的Action类和方法， 并通过IoC方式，将值注入给Aciton。4、Action调用业务逻辑组件处理业务逻辑，这一步包含表单验证。5、Action执行完毕，根据struts.xml中的配置找到对应的返回结果result。6、返回HTTP响应到客户端浏览器。

Spring

Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。在不使用Spring框架之前，我们的service层中要使用dao层的对象，不得不在service层中new一个对象。使用Spring后，service层要用dao层对象只需要在xml文件中配置。

Hibernate

Hibernate是JDBC进行封装的持久化框架。使用Hibernate可以大大简化数据访问层繁琐的重复性代码。Hibernate还支持延迟加载，即当Hibernate在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，节省了服务器的内存开销，从而提高服务器性能。

总的来说SSH框架可以分为四层：表示层、业务逻辑层、数据持久层和域模块层。

表示层主要由struts负责，通过页面实现交互，传送请求和接收响应，Struts根据配置文件将接收到的Request委派给相应的Action处理。

业务层主要由spring负责，管理服务组件的Spring IoC容器负责向Action提供业务模型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。

持久层主要由Hibernate，通过Hibernate的对象化映射和数据库交互，处理DAO组件请求的数据，返回处理结果。

## 3 项目介绍

下面介绍留言板项目在SSH上的实现。

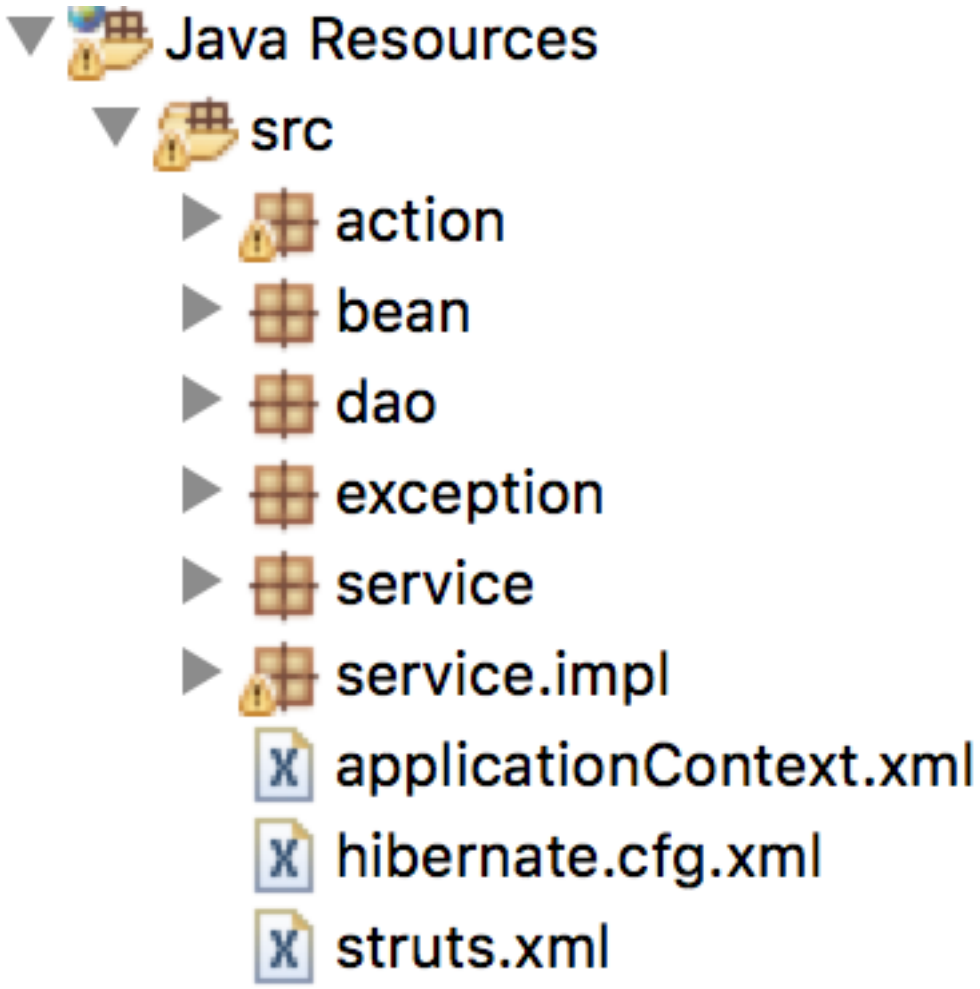
### 3.1 项目需求

实现一个web页面，用户可以在页面上发送留言。页面显示所有用户提交的留言信息，包括用户名，留言内容，留言时间。

### 3.2 开发环境

在eclipse jee下采用 struts2.3.20+spring4.1.6+hibernate4.3.8 开发。相关软件包位于WebContent\WEB-INF\lib

3.2 项目结构




结构如图所示。其中

applicationContext.xml为spring框架相关配置

structs.xml为structs框架相关配置

hibernate.cfg.xml为hibernate框架相关配置

3.3 实现流程

3.3.1 数据库以及bean设置

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
    destroy-method="close">
    <property name="driverClass" value="com.mysql.jdbc.Driver" />
    <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/test?characterEncoding=utf8"></property>
    <property name="user" value="root" />
    <property name="password" value="123456" />
    <property name="maxPoolSize" value="12" />
    <property name="minPoolSize" value="0" />
    <property name="maxStatements" value="200" />
    <property name="initialPoolSize" value="3" />
    <property name="maxIdleTime" value="10" />
    <property name="idleConnectionTestPeriod" value="10" />
</bean>
```

创建Message的javabean并与数据库进行关联。
如图在applicationContext.xml配置Mysql相关参数。

在bean包创建javabean Message.java以及hibernate的映射配置文件Message.hbm.xml。在配置文件中设置Message属性与数据库条目的对应关系。

```
<class name="bean.Message" table="message">
    <id name="messageId" type="java.lang.Integer">
        <column name="message_id" />
        <generator class="increment" />
    </id>
    <property name="messageWords" type="java.lang.String">
        <column name="message_words"  />
    </property>
    <property name="messageUsername" type="java.lang.String">
        <column name="message_username"   />
    </property>
    <property name="messageTime" type="java.lang.String">
        <column name="message_time"   />
    </property>
</class>
```

上图中将Message类映射到数据库中的message表。将messageWords属性对应到数据库中的message_项目。

最后在配置文件hibernate.cfg.xml中设置映射源。即完成了javabean的创建以及与数据库的映射。

```
<mapping resource="bean/Message.hbm.xml" />
```

### 3.3.2添加服务

下面介绍添加发送留言服务。

首先在service包中添加IMessageService.java接口，表示是对于Message进行操作的服务，其中包含void addMessage(Message m);方法。然后在service.impl包中添加其实现MessageServiceImpl.java。实现addMessage方法如下图。

```
private EntityDAO entityDAO;

public void setEntityDAO(EntityDAO entityDAO) {
    this.entityDAO = entityDAO;
}

@Override
public void addMessage(Message m) {
    // TODO Auto-generated method stub
    entityDAO.save(m);
}
```

entityDAO是经过hibernate封装的DAO，可以看到在数据库中添加message繁琐的语句被简化为了一句entityDAO.save(m)。

接下来我们还需要添加一个action处理页面发送的请求。在action包中添加SendMessage.java。如下图

```
private String messageWords;

private String messageUsername;

private Message message = new Message();

private IMessageService messageService;

public String execute() {

    Date date = new Date(System.currentTimeMillis());
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String messageTime = dateFormat.format(date);

    message.setMessageUsername(messageUsername);
    message.setMessageWords(messageWords);
    message.setMessageTime(messageTime);
    messageService.addMessage(message);

    return "success";
}
```

messageWords，messageUsername作为请求的参数。excute()方法对请求进行处理，根据参数给message赋值后通过messageService的addMessage方法添加到数据库。随后还要添加messageWords，messageUsername的setter方法以便参数的传入。

```
<bean id="sendMessageAction" class="action.SendMessage"
    scope="prototype">
    <property name="messageService" ref="MessageService"></property>
</bean>

<bean id="MessageService" class="service.impl.MessageServiceImpl">
    <property name="entityDAO" ref="EntityDAO"></property>
</bean>
```

之后需要在applicationContext.xml中配置action和service如下图

这样我们就获得了添加留言的服务，页面可以通过提交sendMessage请求以及messageWords，messageUsername参数调用该服务。

最后我们需要在structs.xml中配置该action的返回值，如下。可见sendMessage的返回值是

```
<action name="sendMessage" class="sendMessageAction">
    <result name="success">/messageBoard.html</result>
</action>
```

一个html页面。

我们可以用同样的方法添加getMessage请求以及相关服务，这里不再赘述。

## 4 部署

完成整个项目后，我尝试将项目部署到一台VPS上。VPS的系统为centos6。部署的主要流程如下。

1、配置java环境，通过yum安装jdk。
2、安装tomcat，通过wget在官网下载tomcat7进行安装。
3、安装myspl，通过yum安装phpMyAdmin
4、创建数据库，通过phpMyAdmin在网页创建项目需要的页表
5、项目打包，在eclipse中将项目打包为.war文件
6、将war文件上传至服务器tomcat的webapp目录下，重启tomcat服务。

可以通过http://45.78.59.29:8080/Demo/访问部署好的项目。

## 5 总结

通过这次项目实践，我初步体验了web项目在SSH框架上的开发。这也是我第一次在web框架上开发项目。给我最大的感受是，相较于传统的，javabean，Servlet开发，SSH省去了许多不必要的重复代码，如sql语句。整体的结果更为清晰，可以让我专注到业务逻辑的实现，提高了开发的效率。当然SSH框架还有许多特性在项目中我没有体现，我也需要近一步的学习才能更好的掌握SSH框架的使用。