hibernate综合运用.

以一个内部留言本来讲解.本案例本身不复杂，通过该案例我们讲解 struts+hibernate+接口编程的方式, 主要听点是程序结构的设计.

开始:

* 1. 看需求
  2. 根据需求文档，我们画出程序框架图, 框架图的三个接口，暂时不写，后面我们再引入.
  3. 创建一个web项目.
  4. 创写web层.

4.1引入struts

42.把web层的jsp 表单 action 编写完毕，并且打通这个登录的模块

5. 引入hibernate 使用手动的方法来开发domain和对象关系文件

6. 把hibernate.cfg.xml写好

7. 写成工具类 dao HibernateUtil.java

8. 测试一下看看能否映射ok

测试成功!

9. 我们使用接口来解耦. web---接口--业务层

10. hibernate中，取消lazy 不要在 one-to-many 的 one的一方的<set /> ,可以在 many的一方配置 lazy=false;

11. 添加消息的功能.(假设用户是唯一的.,如果用户不唯一，我们可以通过id来发送信息)

\* 在jsp页面中获取当前项目名称可以

<%=this.getServletContext().getContextPath() %>

通过标签: ${pageContext.request.contextPath}

# Myself

1.新建一个web项目

2注意：让junit 来测试，使用接口，让业务层和 web层完全分开，使用接口

3.然后 引入Struts （除了web层，还有其他的框架，比如）

4.然后我们一步一步设置好一些简单的

5.开始引入hibernate

## 1.各个文件

## Users //这里用户 可以发动信息，也可以 接收信息 所以这个就是 one-to-many

**private** Integer userid;

**private** String username;

**private** String password;

**private** Set<Message> sendMessages;

**private** Set<Message> getMessages;

### Users.hbm.xml <!-- 写message 中的外键名 通过这个 外键 找自己 -->

<set name=*"sendMessages"*>

<!-- 写message 中的外键名 通过这个 外键 找自己 -->

<key column=*"sender\_id"* />

<one-to-many class=*"Message"*/>

</set>

<set name=*"getMessages"*>

<key column=*"getter\_id"* />

<one-to-many class=*"Message"*/>

</set>

## Message

**private** Integer mesId;

**private** java.util.Date mesTime;

**private** String content;

**private** Users sender;

**private** Users getter;

### Message.hbm.xml

<!-- 配置getter属性 -->

<many-to-one name=*"getter"* column=*"getter\_id"* />

<!-- 配置sendder -->

<many-to-one name=*"sender"* column=*"sender\_id"*/>

# 2.开始进行登录了

## 1.很简单，自己看

<a href=*"*${pageContext.request.contextPath}*/login.do?flag=logout"*>

## 2.实现注销

## 3.登录时候进行session保存用户名

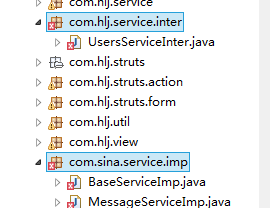
## 4.md5加密

## 4.1先加密然后放到数据库中去，下面这个是查找

String sql = "from Users where username='"+users.getUsername()+"'and password = '"+MyTools.Md5.*MD5Encrypt*(users.getPassword())+"'";

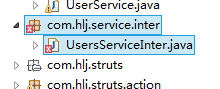
# 3.进行接口

## 3.1新建一个接口包Inter和实现的包Imp，



## 3.2然后我们将之前的service导入到 imp中去，删除之前的，imp->implments,并且将名字改为 UserServiceImp.java

## 3.3接口类中设置我们的接口，相当于UsersServiceInter中有一个接口，将来呢在Imp中进行实现



**public** **interface** UsersServiceInter {

**public** **boolean** checkUser2(Users users);

}

## 3.4然后Imp对接口进行实现

**public** **class** UserServiceImp **implements** UsersServiceInter{

## 3.5.回到loginAction中进行重写设置

//通过使用接口来进行调用

UsersServiceInter userService = **new** UserServiceImp();

# 4.接口详解基础接口，凡是继承的都要实现

## 1.将来里面有很多的方法，

### 比如：通过id号，返回一个user对象

## 2.为了满足普遍接口中就出现了基础接口，将一些常用的方法就直接定义到了BaseInter

**public** **interface** BaseInterface {

//将一些通用的方法 放到基础接口中

//通过id号 获取 用户对象，这里不用Integer 使用Serializable，将对象序列话

**public** Object findById(Class clazz,java.io.Serializable id);

}

# 5.接3.5然后我们到了UsersServiceInter接口中，让他继承这个基础接口，发现报错

**public** **interface** UsersServiceInter **extends** BaseInterface {

//设置接口方法

**public** **boolean** checkUser2(Users users);

}

## 5.2但是这样的话，我们凡是接口UsersServiceInter 都要实现继承接口BaseInterface，这样很不好，方法显得太多了，所以这样我们设置一个Imp抽象他进行接口他

**public** **abstract** **class** BaseServiceImp **implements** BaseInterface {

**public** Object findById(Class clazz, Serializable id) {

// **TODO** Auto-generated method stub

**return** **null**;

}

}

## 5.3然后让UserServiceImp 继承BaseServiceImp，这样就不会报错了，这样我们直接就可以使用基础接口中的东西

**public** **class** UserServiceImp **extends** BaseServiceImp **implements** UsersServiceInter{

## 5.4然后我们在工具类中添加上通过用户查找

**public** **abstract** **class** BaseServiceImp **implements** BaseInterface {

**public** Object findById(Class clazz, Serializable id) {

// **TODO** Auto-generated method stub

**return** HibernateUtil.*findById*(clazz, id);

}

}

# 6.开始取得信息，的一些东西，首先我们想到就是在MessageServiceInter接口中应该添加什么内容让messgeServiceImp来实现

## 6.2取得信息列表

**public** **interface** MessageServiceInter {

//现实用户接收到的所有的message

**public** List<Message> showMessages();

}

**public** **class** MessageServiceImp **extends** BaseServiceImp **implements** MessageServiceInter{

@Override

**public** List<Message> showMessages() {

// **TODO** Auto-generated method stub

String hql = "from Message";

String parameters = **null**;

List<Message> messages = HibernateUtil.*executeQuery*(hql, **null**);

**return** messages;

}

}

# 7.开始显示，但是要注意懒加载，我们需要关闭懒加载才可以的

<font size=*"6"*><b><a href=*"*${pageContext.request.contextPath}*/login.do?flag=logout"*>退出系统</a></b></font><br/>

欢迎${userinfo.username } 留言信息：

<table width=*"500px"*>

<tr><td>发送人</td><td>发送时间</td><td>接收人</td><td>信息内容</td></tr>

<c:forEach items=*"*${message}*"* var=*"message"*>

<tr>

<td>${message.sender.username }</td>

<td>${message.mesTime }</td>

<td>${message.getter.username }</td>

<td>${message.content }</td>

</tr>

</c:forEach>

# 8.取得的信息是所有的数据库中的信息，但是真实情况下是获得登录对象自己的信息

# 9.通过过滤器解决懒加载

# 二注意：

## 1/ 的有无

<form action=*"*${pageContext.request.contextPath}*/login.do?flag=login"* method=*"post"*>