

J2EE 轻量级框架

实验 P3

学号：SA16225221

姓名：欧勇

报告撰写时间：2016/12/27

1. 实验环境/器材

操作系统: Windows 10

IDE: Eclipse Kepler

SDK: JDK 1.8

Web Server: tomcat

数据库: MySQL 5.1.53

数据库可视化管理软件: Wamp Server

浏览器: Chrome 54.0.2840.87 m (64-bit)

2. 实验目的

搭建 SSH 开发环境, 理解 SSH 程序开发基本概念和调试方法。

3. 实验内容

1. Hibernate O/R Mapping. The following diagrams are helpful to express your idea:

A. UML class diagram

B. E-R model

2. Hibernate CRUD operation demo. The following diagrams are helpful to express your idea:

A. UML class diagram

B. UML sequence diagram

4. 实验过程

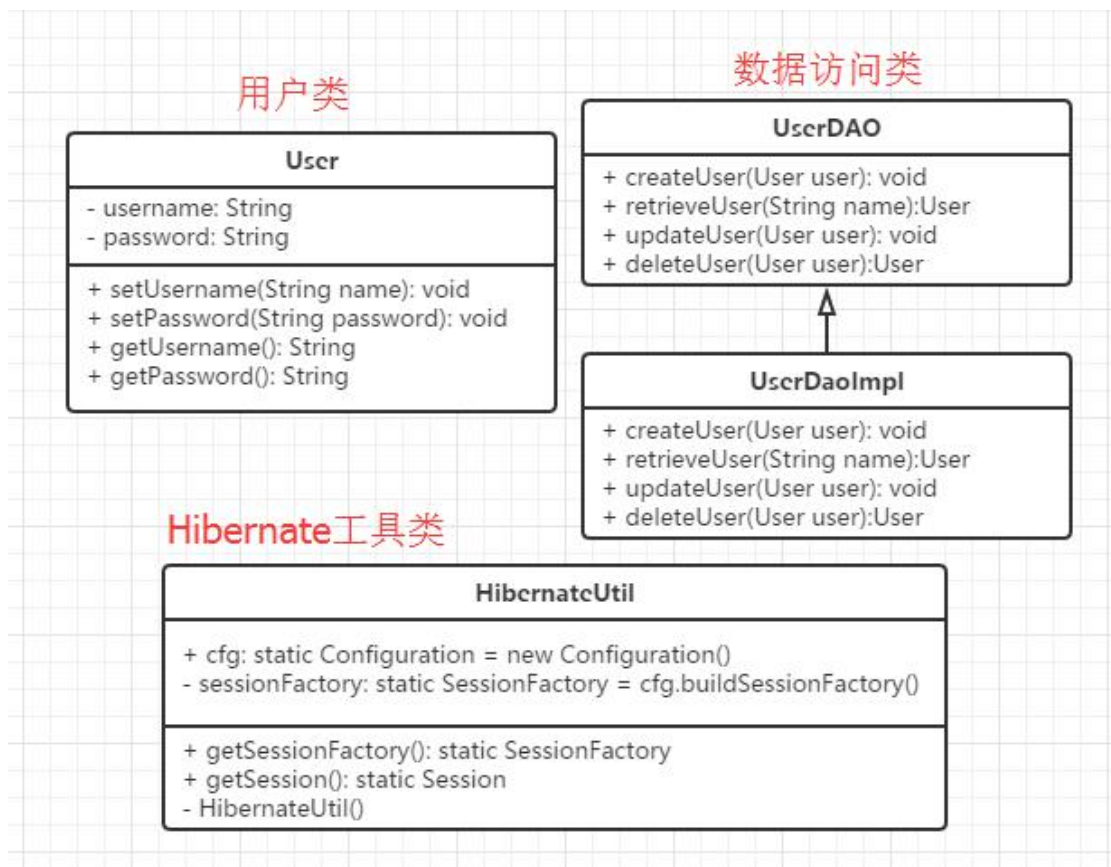


图 1. 类图

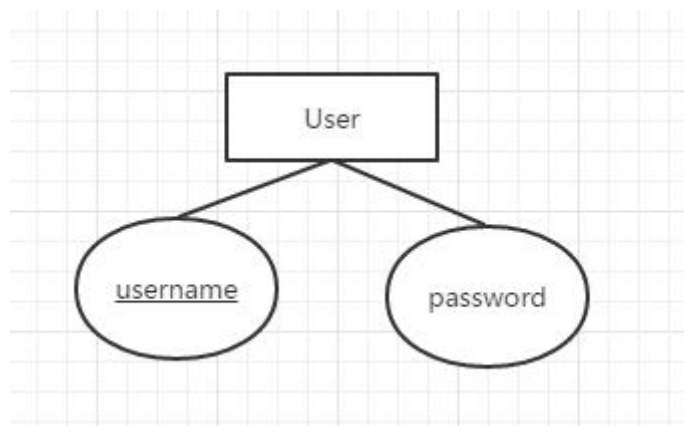


图 2. ER 模型（本次实验仅用到 User 类）

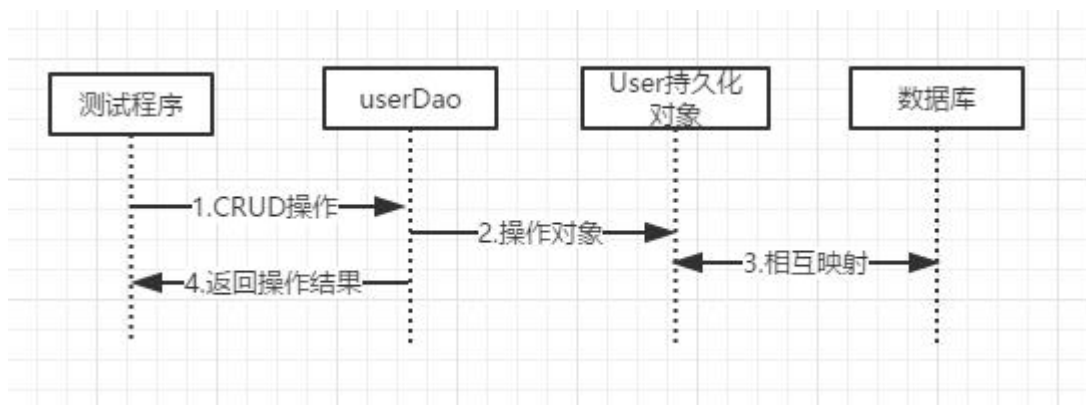


图 3. 顺序图

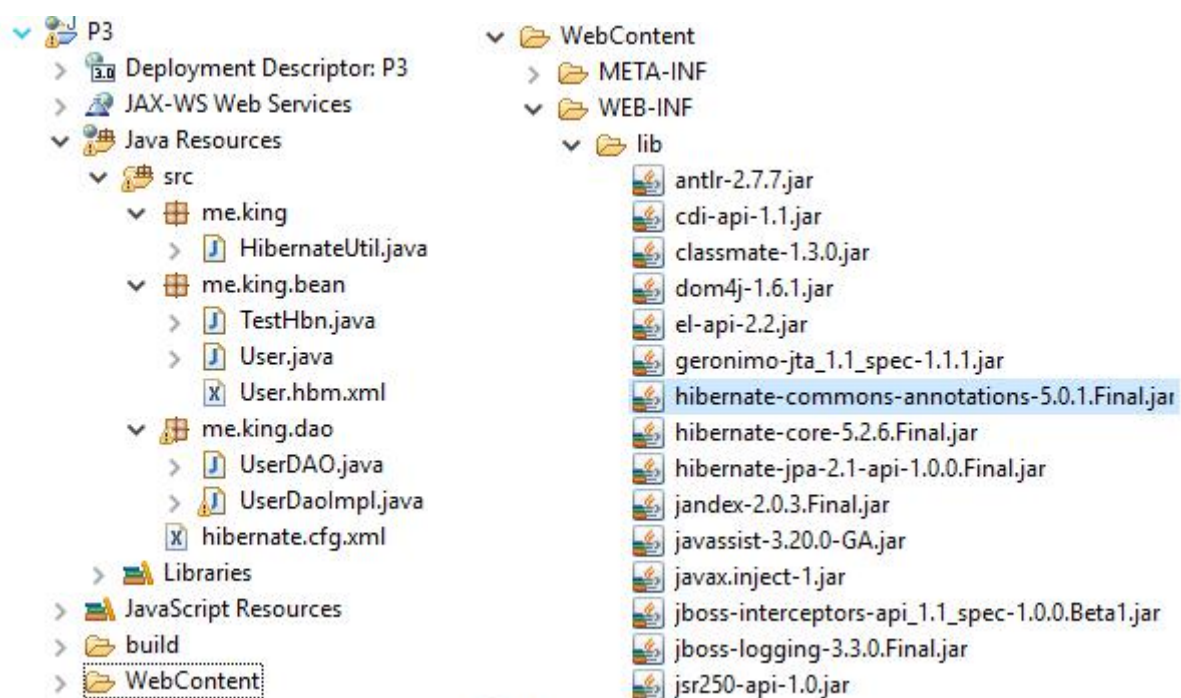
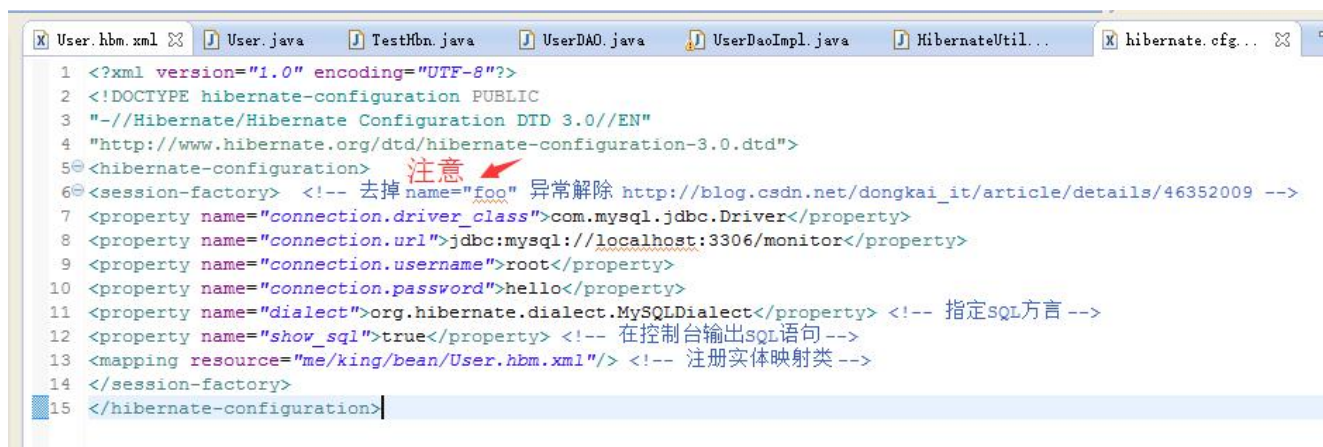


图 4. 项目文件结构图

虽然项目的文件结构为一个 J2EE 的动态 Web 项目，但是为了简单起见，本次实验测试的时候仅仅只用到了一个普通的 java 类 TestHbn 作为测试，而并没有使用 jsp 页面。



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5 <hibernate-configuration>
6 <!-- 注意 去掉 name="foo" 异常解除 http://blog.csdn.net/dongkai_it/article/details/46352009 -->
7 <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
8 <property name="connection.url">jdbc:mysql://localhost:3306/monitor</property>
9 <property name="connection.username">root</property>
10 <property name="connection.password">hello</property>
11 <property name="dialect">org.hibernate.dialect.MySQLDialect</property> <!-- 指定SQL方言 -->
12 <property name="show_sql">>true</property> <!-- 在控制台输出SQL语句 -->
13 <mapping resource="me/king/bean/User.hbm.xml"/> <!-- 注册实体映射类 -->
14 </session-factory>
15 </hibernate-configuration>

```

图 5. hibernate.cfg.xml 配置文件

从第 7 行~第 10 行配置了数据库 Driver 类，数据库 url 地址（MySQL 为 url 格式），数据库连接用户名，数据库连接密码，

第 11 行指定了数据库方言（每种数据库的方言不太一样，最好指定使用的数据库的方言），

第 12 行配置在控制台输出 hibernate 自动生成的 sql 语句（方便调试），

在 13 行配置文件中注册与数据库表的实体映射类的配置文件位置，此处指定为 [me/king/bean/User.hbm.xml](#)

注意此处若在 session-factory 标签内指定 name 属性的话会报“错误解析 JNDI 名字”的错误。如下图 5.1，解决方法是不指定 name 属性



```

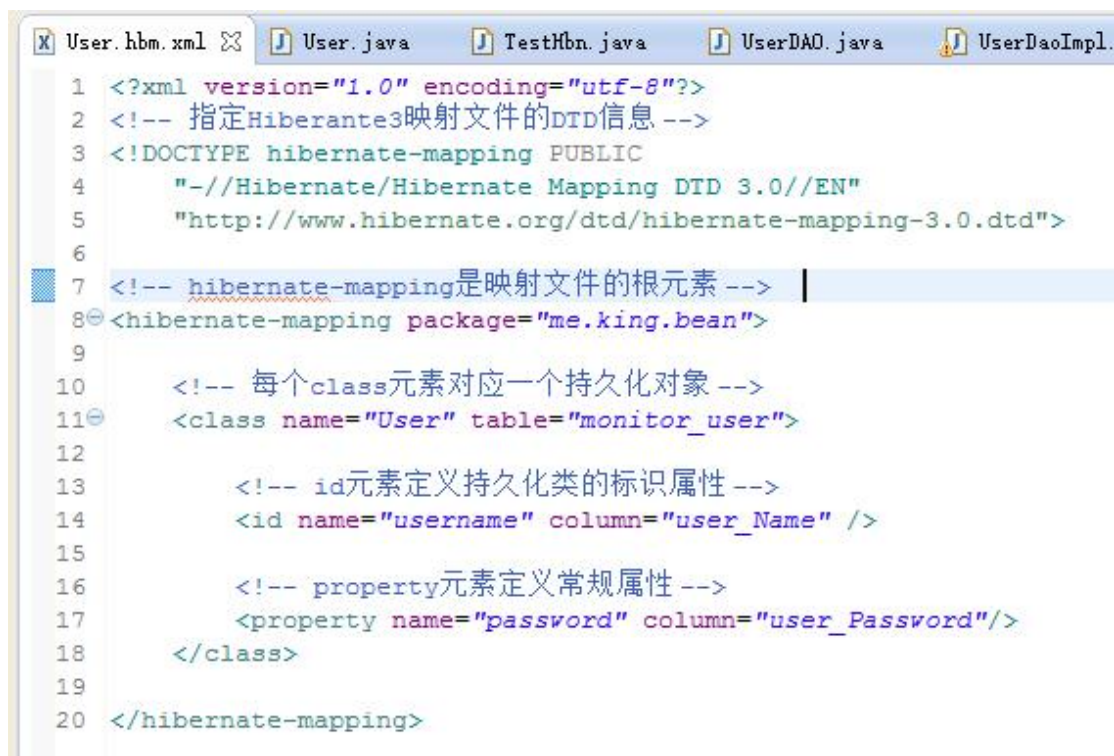
org.hibernate.engine.jndi.JndiException: Error parsing JNDI name [foo]
    at org.hibernate.engine.jndi.internal.JndiServiceImpl.parseName(JndiServiceImpl.java:124)
    at org.hibernate.engine.jndi.internal.JndiServiceImpl.bind(JndiServiceImpl.java:140)
    at org.hibernate.internal.SessionFactoryRegistry.addSessionFactory(SessionFactoryRegistry.java:88)
    at org.hibernate.internal.SessionFactoryImpl.<init>(SessionFactoryImpl.java:368)
    at org.hibernate.boot.internal.SessionFactoryBuilderImpl.build(SessionFactoryBuilderImpl.java:445)
    at org.hibernate.cfg.Configuration.buildSessionFactory(Configuration.java:710)
    at org.hibernate.cfg.Configuration.buildSessionFactory(Configuration.java:726)
    at me.king.HibernateUtil.<clinit>(HibernateUtil.java:15)
    at me.king.dao.UserDaoImpl.retrieveUser(UserDaoImpl.java:36)
    at me.king.bean.TestHbn.main(TestHbn.java:18)

```

图 5.1 JNDI 名字错误截图

由于本次实验中并没有使用到 JNDI 相关的东西，但在 session-factory 中配置 name 属性之后，Hibernate 试图将这个 sessionFacoty 注册到 JNDI 中，但是却无法解析配置的 name，所以报错。

解决方法参考此博客：http://blog.csdn.net/dongkai_it/article/details/46352009




```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!-- 指定Hibernate3映射文件的DTD信息 -->
3 <!DOCTYPE hibernate-mapping PUBLIC
4     "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
5     "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
6
7 <!-- hibernate-mapping是映射文件的根元素 -->
8 <hibernate-mapping package="me.king.bean">
9
10     <!-- 每个class元素对应一个持久化对象 -->
11     <class name="User" table="monitor_user">
12
13         <!-- id元素定义持久化类的标识属性 -->
14         <id name="username" column="user_Name" />
15
16         <!-- property元素定义常规属性 -->
17         <property name="password" column="user_Password"/>
18     </class>
19
20 </hibernate-mapping>
```

图 6. User.hbm.xml 映射配置文件

如图 6 所示，在 User.hbm.xml 文件中配置了 User 类与数据库中表的映射关系，此文件的命名必须遵循 Xxx.hbm.xml 的格式，这样 hibernate 框架才会自动去帮助映射。

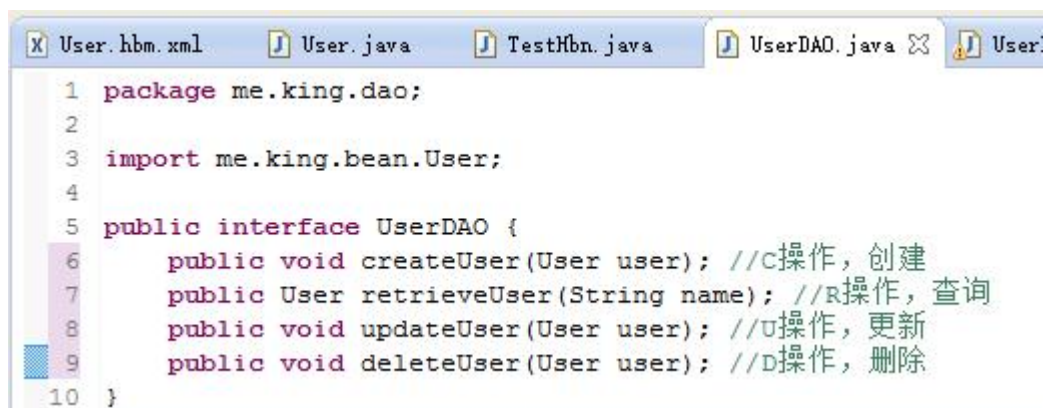
将 monitor_user 表映射到 User 类上，id 标签定义持久化类的标识属性，即表的主键，property 定义其他的常规的属性，此处的定义标签中，若表的字段与持久化类属性名称不一致则需显式使用 column 属性配置对应的表字段，否则可以省略 column 属性。



```
1 package me.king.bean;
2
3 public class User {
4     private String username;
5     private String password;
6
7     public String getUsername() {
8         return username;
9     }
10
11    public void setUsername(String username) {
12        this.username = username;
13    }
14
15    public String getPassword() {
16        return password;
17    }
18
19    public void setPassword(String password) {
20        this.password = password;
21    }
22 }
```

图 7. 持久化类 User

由于 hibernate 的配置，此持久化类就是一个普通的 java 类，即 pojo，同时由于采用配置文件的方式，所以并没有使用注解。



```
1 package me.king.dao;
2
3 import me.king.bean.User;
4
5 public interface UserDAO {
6     public void createUser(User user); //C操作，创建
7     public User retrieveUser(String name); //R操作，查询
8     public void updateUser(User user); //U操作，更新
9     public void deleteUser(User user); //D操作，删除
10 }
```

图 8. UserDAO 接口

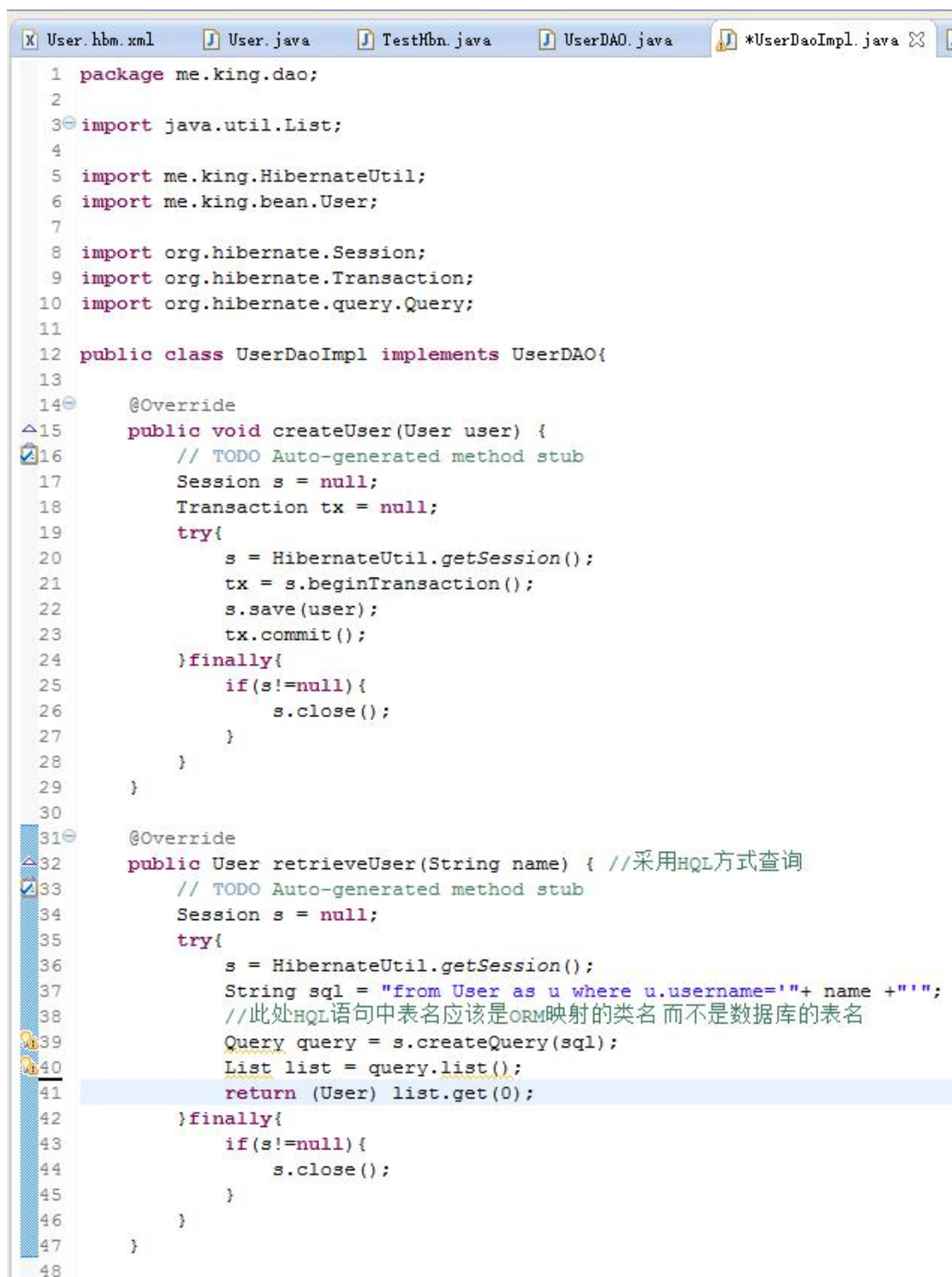
定义 CRUD 四种操作，分别是：

根据一个 User 对象在数据库中创建一个新的记录，

根据用户名查询数据库中的记录，并返回为 User 对象，

将当前传入的 User 对象的信息更新到数据库中，

将传入的 User 对象的数据库记录删除。



```
1 package me.king.dao;
2
3 import java.util.List;
4
5 import me.king.HibernateUtil;
6 import me.king.bean.User;
7
8 import org.hibernate.Session;
9 import org.hibernate.Transaction;
10 import org.hibernate.query.Query;
11
12 public class UserDaoImpl implements UserDao{
13
14     @Override
15     public void createUser(User user) {
16         // TODO Auto-generated method stub
17         Session s = null;
18         Transaction tx = null;
19         try{
20             s = HibernateUtil.getSession();
21             tx = s.beginTransaction();
22             s.save(user);
23             tx.commit();
24         }finally{
25             if(s!=null){
26                 s.close();
27             }
28         }
29     }
30
31     @Override
32     public User retrieveUser(String name) { //采用HQL方式查询
33         // TODO Auto-generated method stub
34         Session s = null;
35         try{
36             s = HibernateUtil.getSession();
37             String sql = "from User as u where u.username='"+ name +"'";
38             //此处HQL语句中表名应该是ORM映射的类名 而不是数据库的表名
39             Query query = s.createQuery(sql);
40             List list = query.list();
41             return (User) list.get(0);
42         }finally{
43             if(s!=null){
44                 s.close();
45             }
46         }
47     }
48 }
```

图 9.1 UserDaoImpl 实现类 createUser 和 retrieveUser 方法

图 9.1 中，采用 HQL 的方式查询，同时第 37 行需要注意 sql 语句的写法，表名需要使用 ORM 映射的类名而不是数据库的表名。否则会报 Xxxx is not mapped 的错误。如图 9.2


```

Caused by: org.hibernate.hql.internal.ast.QuerySyntaxException: monitor_User is not mapped [from monitor_User as u where u.username='test']
    at org.hibernate.hql.internal.ast.QuerySyntaxException.generateQueryException(QuerySyntaxException.java:79)
    at org.hibernate.QueryException.wrapWithQueryString(QueryException.java:103)
    at org.hibernate.hql.internal.ast.QueryTranslatorImpl.doCompile(QueryTranslatorImpl.java:217)
    at org.hibernate.hql.internal.ast.QueryTranslatorImpl.compile(QueryTranslatorImpl.java:141)
    at org.hibernate.engine.query.spi.HQLQueryPlan.<init>(HQLQueryPlan.java:115)
    at org.hibernate.engine.query.spi.HQLQueryPlan.<init>(HQLQueryPlan.java:77)
    at org.hibernate.engine.query.spi.QueryPlanCache.getHQLQueryPlan(QueryPlanCache.java:153)
    at org.hibernate.internal.AbstractSharedSessionContract.getQueryPlan(AbstractSharedSessionContract.java:545)
    at org.hibernate.internal.AbstractSharedSessionContract.createQuery(AbstractSharedSessionContract.java:654)
    ... 3 more
Caused by: org.hibernate.hql.internal.ast.QuerySyntaxException: monitor_User is not mapped
    at org.hibernate.hql.internal.ast.util.SessionFactoryHelper.requireClassPersister(SessionFactoryHelper.java:171)
    at org.hibernate.hql.internal.ast.tree.FromElementFactory.addFromElement(FromElementFactory.java:91)
    at org.hibernate.hql.internal.ast.tree.FromClause.addFromElement(FromClause.java:79)
    
```

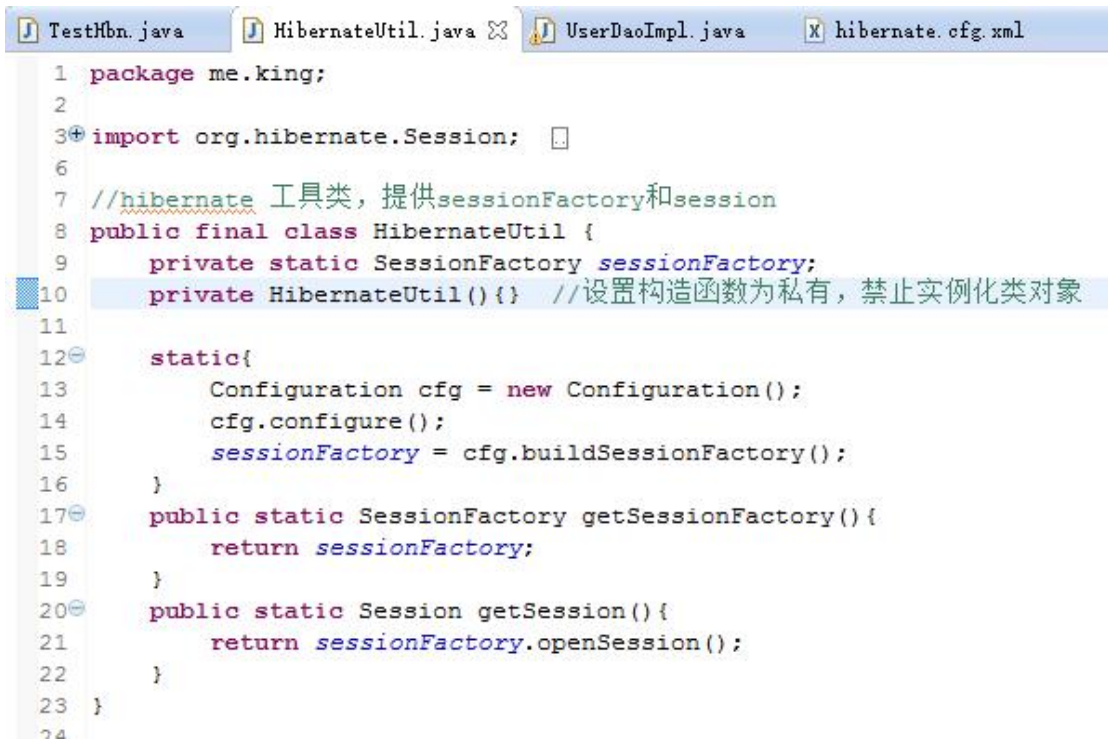
图 9.2 Xxxx is not mapped 错误截图

解决方法参考博客：http://blog.csdn.net/jsj_126abc/article/details/6582074

```

52 @Override
53 public void updateUser(User user) {
54     // TODO Auto-generated method stub
55     Session s = null;
56     Transaction tx = null;
57     try{
58         s = HibernateUtil.getSession();
59         tx = s.beginTransaction();
60         s.update(user);
61         tx.commit();
62     }finally{
63         if(s!=null){
64             s.close();
65         }
66     }
67 }
68
69 @Override
70 public void deleteUser(User user) {
71     // TODO Auto-generated method stub
72     Session s = null;
73     Transaction tx = null;
74     try{
75         s = HibernateUtil.getSession();
76         tx = s.beginTransaction();
77         s.delete(user);
78         tx.commit();
79     }finally{
80         if(s!=null){
81             s.close();
82         }
83     }
84 }
85 }
    
```

图 9.2. UserDaoImpl 实现类 updateUser 和 deleteUser 方法



```
1 package me.king;
2
3 import org.hibernate.Session;
4
5
6 //hibernate 工具类，提供sessionFactory和session
7 public final class HibernateUtil {
8     private static SessionFactory sessionFactory;
9     private HibernateUtil(){} //设置构造函数为私有，禁止实例化类对象
10
11
12     static{
13         Configuration cfg = new Configuration();
14         cfg.configure();
15         sessionFactory = cfg.buildSessionFactory();
16     }
17     public static SessionFactory getSessionFactory(){
18         return sessionFactory;
19     }
20     public static Session getSession(){
21         return sessionFactory.openSession();
22     }
23 }
24
```

图 10. HibernateUtil 工具类

将类构造函数定为私有，禁止实例化类的对象，因为此类所有的属性和方法都是静态的。对外暴露的只有 `getSessionFactory` 和 `getSession` 两个静态方法。

```

1 package me.king.bean;
2
3 import me.king.dao.UserDAO;
4
5
6 public class TestHbn
7 {
8     public static void main(String[] args)
9         throws Exception
10    {
11        UserDAO dao = new UserDaoImpl();
12        User user = new User();
13        user.setUsername("bbbb");
14        user.setPassword("bbbbbb");
15        dao.createUser(user); //创建新用户bbbb
16
17        User user1 = dao.retrieveUser("test"); //获取test用户
18        dao.deleteUser(user1); //删除test用户
19
20        User user2 = dao.retrieveUser("name"); //获取name用户
21        user2.setPassword("name"); //更新密码为name
22        dao.updateUser(user2); //更新
23
24    }
25 }

```

图 11. TestHbn 测试类

图 11.TestHbn 测试类，在类主函数 main 中，直接对本次实验需要的 hibernate CRUD 操作进行测试，测试方式如下：

- 新建一个用户名为 bbbb，密码为 bbbbbb 的 User 类，并将其保存到数据库中；
- 获取 test 用户（已经预先在数据库中的待删除类），并将其删除；
- 获取 name 用户（原始密码为 newpwd），将其密码改为 name，并更新到数据库中。

测试前数据库截图如下图 13，测试时控制台输出的 SQL 语句及辅助信息如图 12，测试完成之后数据库截图为图 14。

```

INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: insert into monitor_user (user_Password, user_Name) values (?, ?)
Dec 27, 2016 6:27:36 PM org.hibernate.hql.internal.QueryTranslatorFactoryInitiator initiateService
INFO: HHH000397: Using ASTQueryTranslatorFactory
Hibernate: select user0_.user_Name as user_Nam1_0_, user0_.user_Password as user_Pas2_0_ from monitor_user user0_ where user0_.user_Name='test'
0 : test
Hibernate: select user_.user_Name, user_.user_Password as user_Pas2_0_ from monitor_user user_ where user_.user_Name=?
Hibernate: delete from monitor_user where user_Name=?
Hibernate: select user0_.user_Name as user_Nam1_0_, user0_.user_Password as user_Pas2_0_ from monitor_user user0_ where user0_.user_Name='name'
0 : newpwd
Hibernate: update monitor_user set user_Password=? where user_Name=?

```

图 12. 测试时控制台输出的 SQL 语句及辅助信息截图

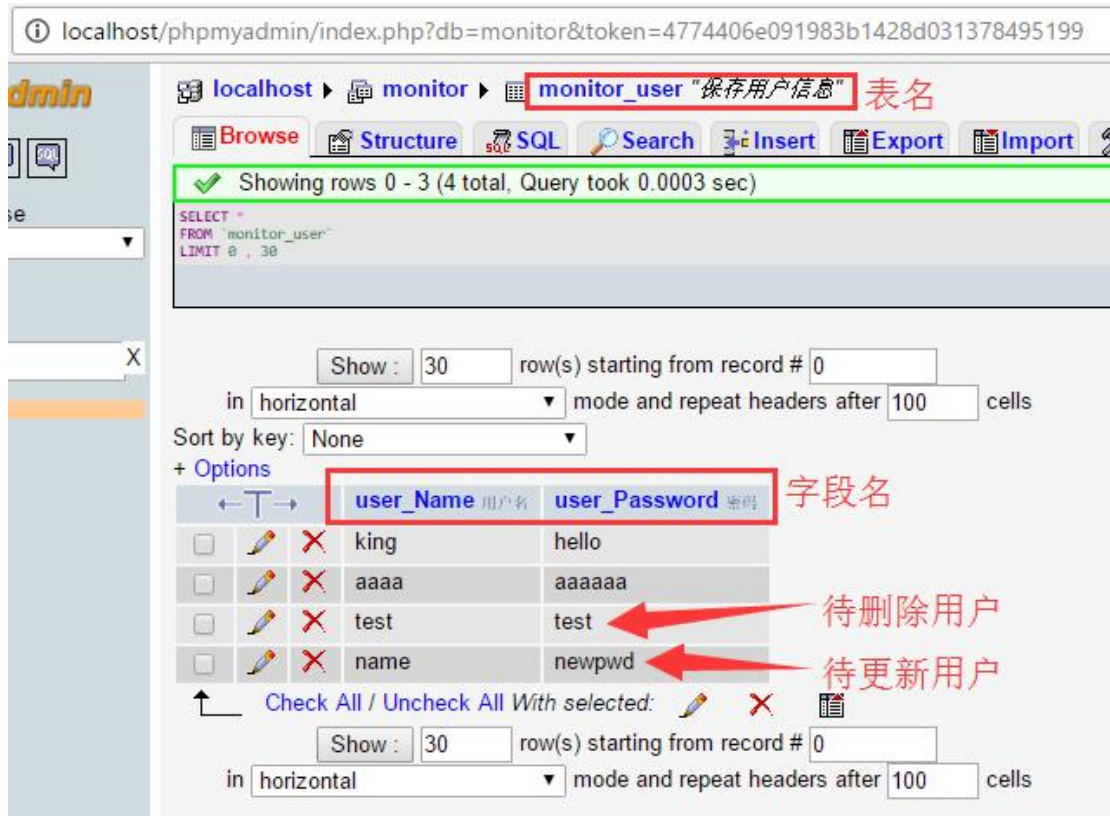


图 13. 测试前数据库截图

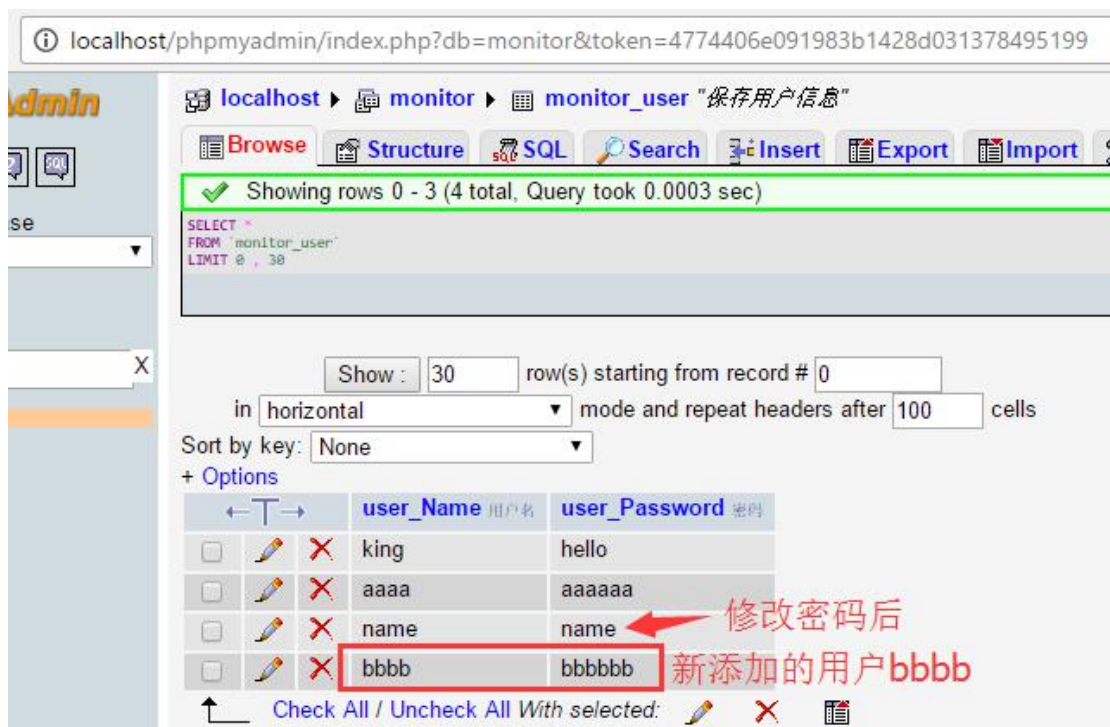


图 14. 测试后数据库截图

5. 实验总结

对概念/方法的理解与总结，实验碰到的问题及解决方法.....

遇到的一些问题及解决方法已经在上文提及。

本次实验仅仅只做了最简单的 hibernate 的映射测试，用到的实体类只有一个简单的 user 类，而且也只有 username 和 password 两个属性，测试的方式也很简单，直接使用 java 执行程序，而没有使用 web，还有许多 hibernate 的功能没有用到。

但即使是这样，也遇到了很多问题，而这些问题都比较奇怪，比如我用的 hibernate 版本为 5.2.6，在实现 UserDao 的查询方法的时候，在网上查找到很多种实现查询的方法，我测试了其中的三种，分别为 HQL 查询方法，对象化查询 Criteria 方法，native sql 查询方法，结果 Eclipse 都提示已经过时，这让我很是无语，由于下次实验也需要使用，所以最后还是采用了 HQL 方式。

框架帮我们做了很多，但是却也隐藏了很多，必须遵循它的方式，否则会报很多莫名其妙的错误。