# 实验三 基于Hibernate的CRUD

一、实验目的：

熟悉并掌握运用Servlet开发模式(HTML+JSP+JavaBean+Servlet)和Hibernate实现数据库CRUD基本编程。

二、实验内容：

在MyEclipse环境下运用HTML+JSP+JavaBean+Servlet以及Hibernate技术完成相应的数据库CRUD功能，调试运行程序。

三、实验要求：

1. 熟悉并掌握运用MVC技术和Hibernate技术开发功能模块的基本步骤；

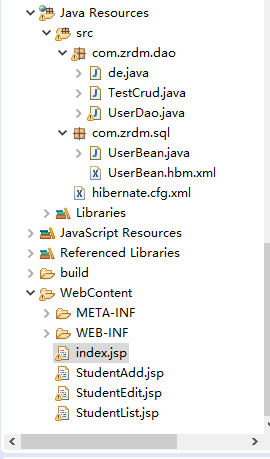
2. 运用相关技术（HTML+JSP+JavaBean+Servlet以及Hibernate）完成规定功能；

3. 写出实验报告。

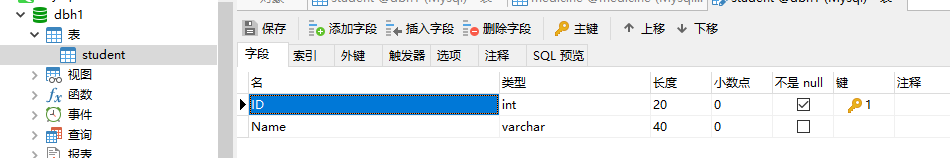
四、实验步骤：

1．进入MyEclipse环境，新建一个Web Project。

目录结构：



2. 设计数据库表结构。



3. 采用MVC技术和Hibernate完成数据库的CRUD。

核心源码：

<?xml version="1.0" encoding="UTF-8"?>

<!-- 在实体类所在的包下，创建一个xml文件。该文件建议名称为：实体类名称+.hbm+.xml

导入约束：dtd约束

-->

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.zrdm.sql">

<class name="UserBean" table="student">

<id name="id" column="ID" type="java.lang.Integer">

<generator class="native"></generator>

</id>

<property name="username" column="Name" type="java.lang.String"></property>

<property name="userid" column="stuid" type="java.lang.Integer"></property>

</class>

</hibernate-mapping>

**package** com.zrdm.dao;

**import** java.util.List;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**import** com.zrdm.sql.\*;;

**public** **class** UserDao

{

/\*\*

\*

\* 通过静态代码块加载配置文件

\*

\* \*\*\*\*/

**static** SessionFactory *sf* = **null**;

**static**

{

// 1 创建Configuration对象，用来加载hibernate的配置文件

Configuration cfg = **new** Configuration();

// 2加载配置文件

cfg.configure("hibernate.cfg.xml");

// 3通过cfg构造一个会话工厂对象

*sf* = cfg.buildSessionFactory();

}

**public** **void** saveStudent(UserBean stu)

{

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

// 获得数据库的会话对象

Session session = *sf*.openSession();

// 打开事务操作

Transaction tr = session.beginTransaction();

// 添加数据

session.save(stu);

// 提交事务

tr.commit();

// 关闭资源

session.close();

}

**public** **void** deleteStudent(UserBean stu)

{

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

// 1

Session session = *sf*.openSession();

// 2

Transaction tr = session.beginTransaction();

// 3

session.delete(stu);

// 4

tr.commit();

// 5

session.close();

}

**public** **void** updateStudent(UserBean stu)

{

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

// 1

Session session = *sf*.openSession();

// 2

Transaction tr = session.beginTransaction();

// 3

//session.update(stu);

/\*\*\*

\*

\* 如果传入的对象的唯一标识是null，那对数据库进行保存操作

\*

\* 如果传入的对象的唯一标识不是null，会进行更新操作，如果不存在

\*

\* \*\*/

session.saveOrUpdate(stu);

// 4

tr.commit();

// 5

session.close();

}

/\*\*\* 多条查询 \*\*\*/

**public** List<UserBean> selStudentAll()

{

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

// 1 session

Session session = *sf*.openSession();

// 2(通过hibernate提供的查询语句进行查询)

String hql = "from student";

// 3执行hql语句

**return** session.createQuery(hql).list();

}

}

**static** SessionFactory *sf* = **null**;

**static**

{

// 1 创建Configuration对象，用来加载hibernate的配置文件

Configuration cfg = **new** Configuration();

// 2加载配置文件

cfg.configure("hibernate.cfg.xml");

// 3通过cfg构造一个会话工厂对象

*sf* = cfg.buildSessionFactory();

}

**public** **void** saveStudent(UserBean stu)

{

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

// 获得数据库的会话对象

Session session = *sf*.openSession();

// 打开事务操作

Transaction tr = session.beginTransaction();

// 添加数据

session.save(stu);

// 提交事务

tr.commit();

// 关闭资源

session.close();

}

**public** **void** deleteStudent(UserBean stu)

{

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

// 1

Session session = *sf*.openSession();

// 2

Transaction tr = session.beginTransaction();

// 3

session.delete(stu);

// 4

tr.commit();

// 5

session.close();

}

**public** **void** updateStudent(UserBean stu)

{

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

// 1

Session session = *sf*.openSession();

// 2

Transaction tr = session.beginTransaction();

// 3

//session.update(stu);

/\*\*\*

\*

\* 如果传入的对象的唯一标识是null，那对数据库进行保存操作

\*

\* 如果传入的对象的唯一标识不是null，会进行更新操作，如果不存在

\*

\* \*\*/

session.saveOrUpdate(stu);

// 4

tr.commit();

// 5

session.close();

}

/\*\*\* 多条查询 \*\*\*/

**public** List<UserBean> selStudentAll()

{

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

// 1 session

Session session = *sf*.openSession();

// 2(通过hibernate提供的查询语句进行查询)

String hql = "from student";

// 3执行hql语句

**return** session.createQuery(hql).list();

}

}

Transaction tr = session.beginTransaction();

// 3

//session.update(stu);

/\*\*\*

\*

\* 如果传入的对象的唯一标识是null，那对数据库进行保存操作

\*

\* 如果传入的对象的唯一标识不是null，会进行更新操作，如果不存在

\*

\* \*\*/

session.saveOrUpdate(stu);

// 4

tr.commit();

// 5

session.close();

}

/\*\*\* 多条查询 \*\*\*/

**public** List<UserBean> selStudentAll()

{

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

// 1 session

Session session = *sf*.openSession();

// 2(通过hibernate提供的查询语句进行查询)

String hql = "from student";

// 3执行hql语句

**return** session.createQuery(hql).list();

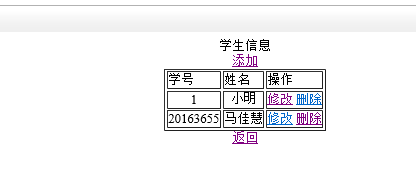
}

}

4. 程序运行截图。



**图1：首页图**

****

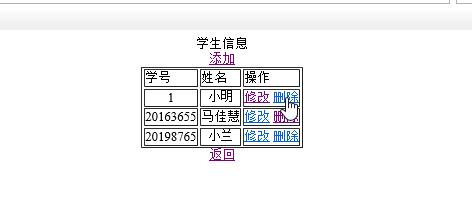
**图2：学生全部信息**

****

**图3：添加学生**

****

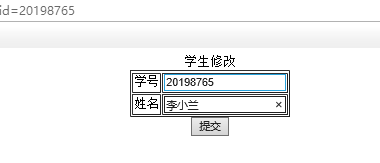
**图4：添加学生展示结果**

****

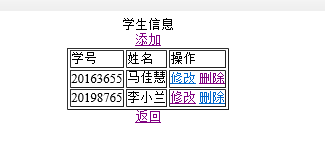
**图5：进行删除**

****

**图6：删除结果**

****

**图7：修改信息**

****

**图8：修改信息结果展示**

五、心得体会：

crud操作总结：

总结API:对象

Configuration ：加载hibernate的配置文件。

SessionFactory:会话工厂，在实际的程序也只存在一个对象。

Session:用来操作数据库的会话对象，产生事务对象

Transaction:事务控制对象

Session方法

BeginTransaction() 开始事务

Save()

saveOrUpdate()

delete()

update()

get()/load();

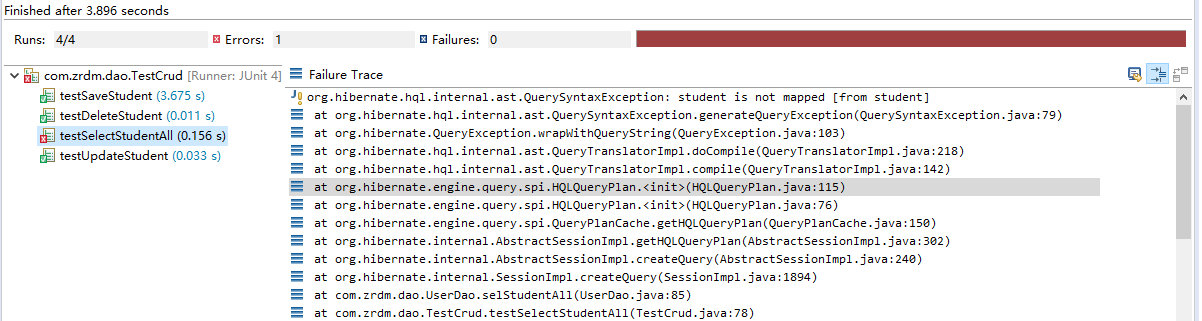
Query q = createQuery()

q.list()

//session.createSQLQuery(arg0)

Query:执行数据库的查询操作，负责执行HQL语句

这次的实验相对于实验二比较简单的还是，主要是这个Hibernate功能比较强大，封装了很多东西，一开始因为不了解，所以下手比较麻烦，参考了博客：<https://www.cnblogs.com/guanghe/p/6093078.html>，所以整体坐下来比较顺利，出现的问题使用unit单元测试也可以很容易知道是哪里的问题，比较方便：



如上图出现了bug,只需要改这一个方法的内容即可。学习到了很多，这个工具着实强大。继续学习！