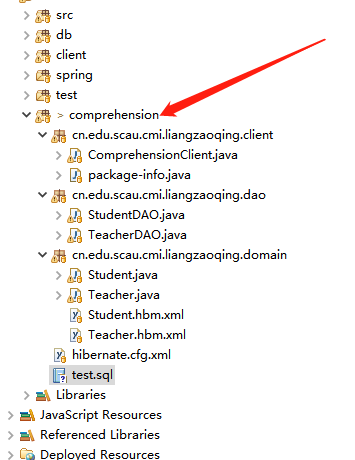
# 实验七：Hibernate框架实验参考

说明：

1. 实验参考的代码在comprehension源文件夹中。数据库有2张表，一个是student，一个是teacher，数据库的建表语句是test.sql，数据库名称是test。
2. 参考使用的是基于xml配置的情况，如果有同学使用注解方式完成的，也没问题。
3. 我只做了控制台客户端，并且只做了添加学生的功能，请同学们用JavaFX客户端完成，当然有同学说要用jsp完成，也可以。
4. 配置文件中需要注意的地方我用红色的字体，当然其他地方都需要注意。
5. **可能需要多次调试才能成功，祝大家顺利。**



# Hibernate的数据库配置文件

hibernate.cfg.xml

<?xml version=*'1.0'* encoding=*'UTF-8'*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"dialect"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/test</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>root</property>

<property name=*"connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<mapping resource=*"*cn/edu/scau/cmi/liangzaoqing/domain/Student.hbm.xml*"* />

<mapping resource=*"*cn/edu/scau/cmi/liangzaoqing/domain/Teacher.hbm.xml*"* />

</session-factory>

</hibernate-configuration>

# Hibernate的ORM映射文件

Student.hbm.xml

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

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

<hibernate-mapping>

<class name=*"*cn.edu.scau.cmi.liangzaoqing.domain.Student*"* table=*"*student*"* catalog=*"*sa*"*>

<id name=*"id"* type=*"java.lang.Long"*>

<column name=*"id"* />

<generator class=*"identity"* />

</id>

<many-to-one name=*"*teacher*"* class=*"cn.*edu*.*scau*.*cmi*.*liangzaoqing*.*domain*.*Teacher*"* fetch=*"select"*>

<column name=*"*tutor*"* />

</many-to-one>

<property name=*"name"* type=*"java.lang.String"*>

<column name=*"*name*"* not-null=*"true"* unique=*"*true*"* />

</property>

</class>

</hibernate-mapping>

Teacher.hbm.xml

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

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

<!--

Mapping file autogenerated by MyEclipse Persistence Tools

-->

<hibernate-mapping>

<class name=*"*cn.edu.scau.cmi.liangzaoqing.domain.Teacher*"* table=*"*teacher*"* catalog=*"*sa*"*>

<id name=*"id"* type=*"java.lang.Long"*>

<column name=*"id"* />

<generator class=*"identity"* />

</id>

<property name=*"name"* type=*"java.lang.String"*>

<column name=*"name"* not-null=*"true"* />

</property>

<set name=*"students"* inverse=*"true"*>

<key>

<column name=*"*tutor*"* />

</key>

<one-to-many class=*"*cn.edu.scau.cmi.liangzaoqing.domain.Student*"* />

</set>

</class>

</hibernate-mapping>

# 实体类

Teacher.java

package cn.edu.scau.cmi.liangzaoqing.domain;

import java.util.HashSet;

import java.util.Set;

public class Teacher implements java.io.Serializable {

public Teacher() {

}

private Long id;

private String name;

private Set students = new HashSet(0);

public Long getId() {

return this.id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return this.name;

}

public void setName(String name) {

this.name = name;

}

public Set getStudents() {

return this.students;

}

public void setStudents(Set students) {

this.students = students;

}

}

Student.java

**package** cn.edu.scau.cmi.liangzaoqing.domain;

**public** **class** Student **implements** java.io.Serializable {

**public** Student() {

}

**private** Long id;

**private** Teacher teacher;

**private** String name;

**public** Long getId() {

**return** **this**.id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Teacher getTeacher() {

**return** **this**.teacher;

}

**public** **void** setTeacher(Teacher teacher) {

**this**.teacher = teacher;

}

**public** String getName() {

**return** **this**.name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

# 客户端界面类

package cn.edu.scau.cmi.liangzaoqing.client;

import java.util.Iterator;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.Transaction;

import cn.edu.scau.cmi.liangzaoqing.dao.StudentDAO;

import cn.edu.scau.cmi.liangzaoqing.dao.customize.ScauCmiHibernateSessionFactoryUtil;

import cn.edu.scau.cmi.liangzaoqing.domain.Student;

import cn.edu.scau.cmi.liangzaoqing.hibernate.dao.BookDAOByHibernate;

import cn.edu.scau.cmi.liangzaoqing.hibernate.dao.HibernateSessionFactoryUtil;

import cn.edu.scau.cmi.liangzaoqing.hibernate.domain.Book;

public class ComprehensionClient {

public static void main(String[] args) {

newStudent();

listStudents();

}

public static void newStudent() {

Session session = ScauCmiHibernateSessionFactoryUtil.getSession();

Transaction transaction = session.beginTransaction();

Student student=new Student();

student.setName("综合性实验的学生姓名");

StudentDAO studentDAO=new StudentDAO();

studentDAO.save(student);

transaction.commit();

session.close();

}

private static void listStudents() {

StudentDAO studentDAO=new StudentDAO();

List students = studentDAO.findAll();

Iterator<?> studentIterator = students.iterator();

while(studentIterator.hasNext()){

Student student = (Student) studentIterator.next();

System.out.println("学生的ID号是："+student.getId()+"学生的名字是："+student.getName());

}

}

}

# 其他类

StudentDAO.java

package cn.edu.scau.cmi.liangzaoqing.dao;

import java.util.List;

import org.hibernate.LockOptions;

import org.hibernate.Query;

import org.hibernate.criterion.Example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import cn.edu.scau.cmi.liangzaoqing.domain.Student;

import cn.edu.scau.cmi.liangzaoqing.hibernate.dao.BaseHibernateDAO;

public class StudentDAO extends BaseHibernateDAO {

private static final Logger log = LoggerFactory.getLogger(StudentDAO.class);

// property constants

public static final String NAME = "name";

public void save(Student transientInstance) {

log.debug("saving Student instance");

try {

getSession().save(transientInstance);

log.debug("save successful");

} catch (RuntimeException re) {

log.error("save failed", re);

throw re;

}

}

public void delete(Student persistentInstance) {

log.debug("deleting Student instance");

try {

getSession().delete(persistentInstance);

log.debug("delete successful");

} catch (RuntimeException re) {

log.error("delete failed", re);

throw re;

}

}

public Student findById(java.lang.Long id) {

log.debug("getting Student instance with id: " + id);

try {

Student instance = (Student) getSession().get("cn.edu.scau.cmi.liangzaoqing.hibernate.Student", id);

return instance;

} catch (RuntimeException re) {

log.error("get failed", re);

throw re;

}

}

public List findByExample(Student instance) {

log.debug("finding Student instance by example");

try {

List results = getSession().createCriteria("cn.edu.scau.cmi.liangzaoqing.hibernate.Student")

.add(Example.create(instance)).list();

log.debug("find by example successful, result size: " + results.size());

return results;

} catch (RuntimeException re) {

log.error("find by example failed", re);

throw re;

}

}

public List findByProperty(String propertyName, Object value) {

log.debug("finding Student instance with property: " + propertyName + ", value: " + value);

try {

String queryString = "from Student as model where model." + propertyName + "= ?";

Query queryObject = getSession().createQuery(queryString);

queryObject.setParameter(0, value);

return queryObject.list();

} catch (RuntimeException re) {

log.error("find by property name failed", re);

throw re;

}

}

public List findByName(Object name) {

return findByProperty(NAME, name);

}

public List findAll() {

log.debug("finding all Student instances");

try {

String queryString = "from Student";

Query queryObject = getSession().createQuery(queryString);

return queryObject.list();

} catch (RuntimeException re) {

log.error("find all failed", re);

throw re;

}

}

public Student merge(Student detachedInstance) {

log.debug("merging Student instance");

try {

Student result = (Student) getSession().merge(detachedInstance);

log.debug("merge successful");

return result;

} catch (RuntimeException re) {

log.error("merge failed", re);

throw re;

}

}

public void attachDirty(Student instance) {

log.debug("attaching dirty Student instance");

try {

getSession().saveOrUpdate(instance);

log.debug("attach successful");

} catch (RuntimeException re) {

log.error("attach failed", re);

throw re;

}

}

public void attachClean(Student instance) {

log.debug("attaching clean Student instance");

try {

getSession().buildLockRequest(LockOptions.NONE).lock(instance);

log.debug("attach successful");

} catch (RuntimeException re) {

log.error("attach failed", re);

throw re;

}

}

}

**package** cn.edu.scau.cmi.liangzaoqing.dao;

**import** java.util.List;

**import** java.util.Set;

**import** org.hibernate.LockOptions;

**import** org.hibernate.~~Query~~;

**import** org.hibernate.criterion.Example;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** cn.edu.scau.cmi.liangzaoqing.domain.Teacher;

**import** cn.edu.scau.cmi.liangzaoqing.hibernate.dao.BaseHibernateDAO;

**public** **class** TeacherDAO **extends** BaseHibernateDAO {

**private** **static** **final** Logger ***log*** = LoggerFactory.*getLogger*(TeacherDAO.**class**);

// property constants

**public** **static** **final** String ***NAME*** = "name";

**public** **void** save(Teacher transientInstance) {

***log***.debug("saving Teacher instance");

**try** {

getSession().save(transientInstance);

***log***.debug("save successful");

} **catch** (RuntimeException re) {

***log***.error("save failed", re);

**throw** re;

}

}

**public** **void** delete(Teacher persistentInstance) {

***log***.debug("deleting Teacher instance");

**try** {

getSession().delete(persistentInstance);

***log***.debug("delete successful");

} **catch** (RuntimeException re) {

***log***.error("delete failed", re);

**throw** re;

}

}

**public** Teacher findById(java.lang.Long id) {

***log***.debug("getting Teacher instance with id: " + id);

**try** {

Teacher instance = (Teacher) getSession().get("cn.edu.scau.cmi.liangzaoqing.hibernate.Teacher", id);

**return** instance;

} **catch** (RuntimeException re) {

***log***.error("get failed", re);

**throw** re;

}

}

**public** List findByExample(Teacher instance) {

***log***.debug("finding Teacher instance by example");

**try** {

List results = getSession().~~createCriteria~~("cn.edu.scau.cmi.liangzaoqing.hibernate.Teacher")

.add(Example.*create*(instance)).list();

***log***.debug("find by example successful, result size: " + results.size());

**return** results;

} **catch** (RuntimeException re) {

***log***.error("find by example failed", re);

**throw** re;

}

}

**public** List findByProperty(String propertyName, Object value) {

***log***.debug("finding Teacher instance with property: " + propertyName + ", value: " + value);

**try** {

String queryString = "from Teacher as model where model." + propertyName + "= ?";

~~Query~~ queryObject = getSession().createQuery(queryString);

queryObject.~~setParameter~~(0, value);

**return** queryObject.~~list~~();

} **catch** (RuntimeException re) {

***log***.error("find by property name failed", re);

**throw** re;

}

}

**public** List findByName(Object name) {

**return** findByProperty(***NAME***, name);

}

**public** List findAll() {

***log***.debug("finding all Teacher instances");

**try** {

String queryString = "from Teacher";

~~Query~~ queryObject = getSession().createQuery(queryString);

**return** queryObject.~~list~~();

} **catch** (RuntimeException re) {

***log***.error("find all failed", re);

**throw** re;

}

}

**public** Teacher merge(Teacher detachedInstance) {

***log***.debug("merging Teacher instance");

**try** {

Teacher result = (Teacher) getSession().merge(detachedInstance);

***log***.debug("merge successful");

**return** result;

} **catch** (RuntimeException re) {

***log***.error("merge failed", re);

**throw** re;

}

}

**public** **void** attachDirty(Teacher instance) {

***log***.debug("attaching dirty Teacher instance");

**try** {

getSession().saveOrUpdate(instance);

***log***.debug("attach successful");

} **catch** (RuntimeException re) {

***log***.error("attach failed", re);

**throw** re;

}

}

**public** **void** attachClean(Teacher instance) {

***log***.debug("attaching clean Teacher instance");

**try** {

getSession().buildLockRequest(LockOptions.***NONE***).lock(instance);

***log***.debug("attach successful");

} **catch** (RuntimeException re) {

***log***.error("attach failed", re);

**throw** re;

}

}

}