# 实验七：Hibernate框架实验

# 项目要求

1. 功能达到实验六的要求；
2. 利用JavaFx完成实验六的界面；
3. 利用Hibernate完成数据的CRUD。

# 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="hbm2ddl.auto">update</property>

<property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>

<property name="connection.url">jdbc:mysql://localhost:3306/ex06</property>

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

<property name="connection.password">25800852</property>

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

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

<mapping resource="cn/edu/scau/cmi/zhengdinghao/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.zhengdinghao.domain.Student" table="student">

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

<column name="id" />

<generator class="identity" />

</id>

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

<column name="tutor" />

</many-to-one>

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

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

</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.zhengdinghao.domain.Teacher" table="teacher">

<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.zhengdinghao.domain.Student" />

</set>

</class>

</hibernate-mapping>

# 实体类

Teacher.java

package cn.edu.scau.cmi.zhengdinghao.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 Teacher(Long id,String name) {

this.id = id;

this.name = name;

}

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.zhengdinghao.domain;

public class Student implements java.io.Serializable {

public Student() {

}

private Long id;

private Teacher teacher;

private String name;

public Student(Long id,String name) {

this.id = id;

this.name = 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

HibernateClient.java

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

import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

import org.hibernate.Transaction;

import cn.edu.scau.cmi.zhengdinghao.domain.\*;

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

import cn.edu.scau.cmi.zhengdinghao.dao.TeacherDAO;

public class HibernateClient {

public static void main(String args[]) {

System.out.println("请选择功能：\n功能1：添加学生\r\n" +

"功能2：修改学生\r\n" +

"功能3：删除学生\r\n" +

"功能4：添加教师\r\n" +

"功能5：修改教师\r\n" +

"功能6：删除教师\r\n" +

"功能7：学生选择指导教师\r\n" +

"功能8：学生修改指导教师\r\n" +

"功能9：教师选择被指导的学生\r\n" +

"功能10：教师修改被指导的学生\r\n" +

"");

Scanner reader = new Scanner(System.in);

Student student;

Teacher teacher;

Set<Student> students;

switch (reader.nextInt()) {

case 1:

System.out.println("请输入要添加的学生id与姓名");

student = new Student(reader.nextLong(),reader.nextLine());

newStudent(student);

break;

case 2:

System.out.println("请输入要修改的学生id");

student = findStudent(reader.nextLong());

System.out.println("请输入修改后的学生的姓名");

student.setName(reader.nextLine());

updateStudent(student);

break;

case 3:

System.out.println("请输入要删除的学生id");

student = new Student(reader.nextLong(),null);

deleteStudent(student);

break;

case 4:

System.out.println("请输入要添加的老师id与姓名");

teacher = new Teacher(reader.nextLong(),reader.nextLine());

newTeacher(teacher);

break;

case 5:

System.out.println("请输入要修改的老师id");

teacher = new Teacher(reader.nextLong(),null);

System.out.println("请输入修改后的老师的姓名");

teacher.setName(reader.nextLine());

updateTeacher(teacher);

break;

case 6:

System.out.println("请输入要删除的老师id");

teacher = new Teacher(reader.nextLong(),null);

deleteTeacher(teacher);

break;

case 7:

System.out.println("请输入要操作的学生id");

student = findStudent(reader.nextLong());

System.out.println("请输入指导老师的id");

teacher = findTeacher(reader.nextLong());

student.setTeacher(teacher);

updateStudent(student);

break;

case 8:

System.out.println("请输入要操作的学生id");

student = findStudent(reader.nextLong());

System.out.println("请输入指导老师的id");

teacher = findTeacher(reader.nextLong());

student.setTeacher(teacher);

updateStudent(student);

break;

case 9:

System.out.println("请输入要操作的老师id");

teacher = findTeacher(reader.nextLong());

System.out.println("请输入学生们的id -1为结束");

students = new HashSet<>();

while((i = reader.nextLong())!=-1) {

student = findStudent(i);

students.add(student);

}

teacher.setStudents(students);

updateTeacher(teacher);

break;

case 10:

System.out.println("请输入要操作的老师id");

teacher = findTeacher(reader.nextLong());

System.out.println("请输入学生们的id -1为结束");

students = new HashSet<>();

Long i;

while((i = reader.nextLong())!=-1) {

student = findStudent(i);

students.add(student);

}

teacher.setStudents(students);

updateTeacher(teacher);

break;

default:

break;

}

}

public static void newStudent(Student student) {

StudentDAO studentDAO = new StudentDAO();

Transaction transaction = studentDAO.getSession().beginTransaction();

studentDAO.save(student);

transaction.commit();

}

public static void updateStudent(Student student) {

StudentDAO studentDAO = new StudentDAO();

Transaction transaction = studentDAO.getSession().beginTransaction();

studentDAO.attachDirty(student);

transaction.commit();

}

public static void deleteStudent(Student student) {

StudentDAO studentDAO = new StudentDAO();

Transaction transaction = studentDAO.getSession().beginTransaction();

studentDAO.delete(student);

transaction.commit();

}

public static void newTeacher(Teacher teacher) {

TeacherDAO teacherDAO = new TeacherDAO();

Transaction transaction = teacherDAO.getSession().beginTransaction();

teacherDAO.save(teacher);

transaction.commit();

}

public static void updateTeacher(Teacher teacher) {

TeacherDAO teacherDAO = new TeacherDAO();

Transaction transaction = teacherDAO.getSession().beginTransaction();

teacherDAO.attachDirty(teacher);

transaction.commit();

}

public static void deleteTeacher(Teacher teacher) {

TeacherDAO teacherDAO = new TeacherDAO();

Transaction transaction = teacherDAO.getSession().beginTransaction();

teacherDAO.delete(teacher);

transaction.commit();

}

public static Student findStudent(Long id) {

StudentDAO studentDAO = new StudentDAO();

return studentDAO.findById(id);

}

public static Teacher findTeacher(Long id) {

TeacherDAO teacherDAO = new TeacherDAO();

return teacherDAO.findById(id);

}

}

# 其他类

StudentDAO.java

package cn.edu.scau.cmi.zhengdinghao.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 org.hibernate.Session;

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

import cn.edu.scau.cmi.zhengdinghao.hibernate.HibernateSessionFactory;

public class StudentDAO {

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

// property constants

public static final String NAME = "name";

public Session getSession() {

return HibernateSessionFactory.getSession();

}

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.zhengdinghao.domain.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.zhengdinghao.domain.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;

}

}

}

TeacherDAO.java

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

import java.util.List;

import java.util.Set;

import org.hibernate.LockOptions;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.criterion.Example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import cn.edu.scau.cmi.zhengdinghao.domain.Teacher;

import cn.edu.scau.cmi.zhengdinghao.hibernate.HibernateSessionFactory;

public class TeacherDAO {

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

// property constants

public static final String NAME = "name";

public Session getSession() {

return HibernateSessionFactory.getSession();

}

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.zhengdinghao.domain.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.zhengdinghao.domain.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;

}

}

}

# 运行效果

截屏展示运行的效果，注意适当的说明和图的编号

无图形界面