# 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

<hibernate-configuration>

<session-factory>

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

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

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/softwarearchitecture?characterEncoding=UTF-8 </property>

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

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

<property name="hibernate.show\_sql">true</property>

<property name="hibernate.format\_sql">true</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<!-- 配置映射文件 -->

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

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

</session-factory>

</hibernate-configuration>

# Hibernate的ORM映射文件

hibernateApplicationContext.xml

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.1.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*>

<!-- 访问数据库的各种dao -->

<bean id=*"studentDAO"* class=*"cn.deu.scau.cmi.longting.hibernate.dao.StudentDao"*></bean>

<bean id=*"teacherDAO"* class=*"cn.deu.scau.cmi.longting.hibernate.dao.TeacherDAO"*></bean>

</beans>

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.longting.hibernate.domain.Student"* table=*"student"* catalog=*"softwarearchitecture"*>

<id name=*"id"* type=*"long"*>

<column name=*"id"*/>

<generator class=*"identity"*/>

</id>

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

<column name=*"tutor"*/>

</many-to-one>

<property name=*"name"* type=*"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">

<hibernate-mapping>

<class name=*"cn.edu.scau.cmi.longting.hibernate.domain.Teacher"* table=*"teacher"* catalog=*"softwarearchitecture"*>

<id name=*"id"* type=*"long"*>

<column name=*"id"*/>

<generator class=*"identity"*/>

</id>

<property name=*"name"* type=*"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.longting.hibernate.domain.Student"* />

</set>

</class>

</hibernate-mapping>

# 实体类

Student.java

**package** cn.edu.scau.cmi.longting.hibernate.domain;

**import** java.io.Serializable;

**public** **class** Student **implements** Serializable{

**private** Long id;

**private** String name;

**private** Teacher teacher;

**public** Student() {

**super**();

}

**public** Student(Long id, String name) {

**super**();

**this**.id = id;

**this**.name = name;

}

**public** Student(Long id, String name, Teacher teacher) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.teacher = teacher;

}

**public** Long getId() {

**return** id;

}

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

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** Teacher getTeacher() {

**return** teacher;

}

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

**this**.teacher = teacher;

}

}

Teacher.java

package cn.edu.scau.cmi.longting.hibernate.domain;

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

public class Teacher implements Serializable{

private Long id;

private String name;

private Set students = new HashSet(0);

public Teacher() {

super();

}

public Teacher(Long id, String name) {

super();

this.id = id;

this.name = name;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Set getStudents() {

return students;

}

public void setStudents(Set students) {

this.students = students;

}

}

# 客户端界面类

package cn.edu.scau.cmi.longting.hibernate.client.fxml;

import java.net.URL;

import java.util.Iterator;

import java.util.List;

import java.util.ResourceBundle;

import org.hibernate.Session;

import cn.edu.scau.cmi.longting.hibernate.dao.StudentDao;

import cn.edu.scau.cmi.longting.hibernate.dao.TeacherDao;

import cn.edu.scau.cmi.longting.hibernate.domain.Student;

import cn.edu.scau.cmi.longting.hibernate.domain.Teacher;

import cn.edu.scau.cmi.longting.hibernate.util.ScauCmiHibernateSessionFactoryUtil;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

public class HibernateClientController implements Initializable{

//学生名字

@FXML

private TextField insert\_stu\_name;

//要更新的学生ID

@FXML

private TextField update\_stu\_id;

//要更新的学生名字

@FXML

private TextField update\_stu\_name;

//要更新的学生指导老师

@FXML

private TextField update\_stu\_tutor;

//要删除的学生ID

@FXML

private TextField del\_stu\_id;

//要增加指导老师的学生id

@FXML

private TextField stu\_id;

//要增加、修改的指导老师id

@FXML

private TextField stu\_tutor;

@FXML

private TextArea allStudents;

@FXML

private TextArea allTeachers;

//老师名字

@FXML

private TextField insert\_teacher\_name;

//要更新的老师ID

@FXML

private TextField update\_teacher\_id;

//要更新的学生名字

@FXML

private TextField update\_teacher\_name;

//要删除的学生ID

@FXML

private TextField del\_teacher\_id;

//要增加指导老师的学生id

@FXML

private TextField teacher\_id;

//要增加、修改的指导老师id

@FXML

private TextField tutor\_stu;

@FXML

private TextArea allStudents2;

@FXML

private TextArea allTeachers2;

// spring注入dao对象

private static StudentDao studentDao = new StudentDao();

private static TeacherDao teacherDao = new TeacherDao();

private static Session session;

@Override

public void initialize(URL arg0, ResourceBundle arg1) {

this.showAllStudents();

this.showAllTeachers();

}

/\*\*

\* 插入学生

\* @param e

\*/

@FXML

public void insert\_stu(ActionEvent e) {

if(this.insert\_stu\_name==null || this.insert\_stu\_name.getText().isEmpty()) {

this.showWarningStage("要插入的姓名不能为空。");

return;

}

Student student = new Student();

student.setName(this.insert\_stu\_name.getText());

studentDao.save(student);

this.showAllStudents();

}

/\*\*

\* 修改学生

\* @param e

\*/

@FXML

public void update\_stu(ActionEvent e) {

if(this.update\_stu\_id==null || update\_stu\_id.getText().isEmpty()) {

this.showWarningStage("要修改学生信息，学生ID 不能为空");

}else if(this.update\_stu\_name==null || update\_stu\_name.getText().isEmpty()) {

this.showWarningStage("要修改ID为 "+this.update\_stu\_id.getText()+" 学生信息，姓名不能为空，请输入要修改的姓名");

}else {

Student student = new Student();

student.setId(Long.valueOf(this.update\_stu\_id.getText()));

student.setName(this.update\_stu\_name.getText());

if(this.update\_stu\_tutor==null || !update\_stu\_tutor.getText().isEmpty()) {

Teacher teacher = new Teacher();

teacher.setId(Long.valueOf(update\_stu\_tutor.getText()));

student.setTeacher(teacher);

}

studentDao.attachDirty(student);

}

this.showAllStudents();

}

/\*\*

\* 删除学生

\* @param e

\*/

@FXML

public void delete\_stu(ActionEvent e) {

if(this.del\_stu\_id==null || del\_stu\_id.getText().isEmpty()) {

this.showWarningStage("请输入要删除的学生ID！");

return;

}

Student student = new Student();

student.setId(Long.valueOf(del\_stu\_id.getText()));

studentDao.delete(student);

this.showAllStudents();

}

/\*\*

\* 增加、修改学生的指导老师

\* @param e

\*/

@FXML

public void stu\_tutor(ActionEvent e) {

if(this.stu\_id==null || stu\_id.getText().isEmpty()) {

this.showWarningStage("要修改学生的指导老师，学生ID不能为空");

}else if(this.stu\_tutor==null || stu\_tutor.getText().isEmpty()) {

this.showWarningStage("要修改ID为 "+this.stu\_id.getText()+" 学生的指导老师，老师ID不能为空！");

}else {

Student student = studentDao.findById(Long.valueOf(stu\_id.getText()));

Teacher teacher = (student.getTeacher()==null ? new Teacher() : student.getTeacher());

teacher.setId(Long.valueOf(stu\_tutor.getText()));

student.setTeacher(teacher);

studentDao.attachDirty(student);

this.showAllStudents();

}

}

/\*\*

\* 插入老师

\* @param e

\*/

@FXML

public void insert\_teacher(ActionEvent e) {

if(this.insert\_teacher\_name==null || this.insert\_teacher\_name.getText().isEmpty()) {

this.showWarningStage("要插入的姓名不能为空。");

return;

}

Teacher teacher = new Teacher();

teacher.setName(insert\_teacher\_name.getText());

teacherDao.save(teacher);

this.showAllTeachers();

}

/\*\*

\* 修改老师

\* @param e

\*/

@FXML

public void update\_teacher(ActionEvent e) {

if(this.update\_teacher\_id==null || update\_teacher\_id.getText().isEmpty()) {

this.showWarningStage("要修改老师信息，老师ID 不能为空");

}else if(this.update\_teacher\_name==null || update\_teacher\_name.getText().isEmpty()) {

this.showWarningStage("要修改ID为 "+this.update\_stu\_id.getText()+" 老师信息，姓名不能为空，请输入要修改的姓名");

}else {

Teacher teacher = new Teacher();

teacher.setId(Long.valueOf(this.update\_teacher\_id.getText()));

teacher.setName(this.update\_teacher\_name.getText());

teacherDao.attachDirty(teacher);

}

this.showAllTeachers();

}

/\*\*

\* 删除老师

\* @param e

\*/

@FXML

public void delete\_teacher(ActionEvent e) {

if(this.del\_teacher\_id==null || del\_teacher\_id.getText().isEmpty()) {

this.showWarningStage("请输入要删除的老师ID！");

return;

}

Teacher teacher = new Teacher();

teacher.setId(Long.valueOf(del\_teacher\_id.getText()));

teacherDao.delete(teacher);

this.showAllTeachers();

}

/\*\*

\* 增加、修改老师指导的学生

\* @param e

\*/

@FXML

public void teacher\_stu(ActionEvent e) {

if(this.teacher\_id==null || teacher\_id.getText().isEmpty()) {

this.showWarningStage("要修改老师指导的学生，老师ID不能为空");

}else if(this.tutor\_stu==null || tutor\_stu.getText().isEmpty()) {

this.showWarningStage("要修改ID为 "+this.teacher\_id.getText()+" 老师所指导的学生，学生ID不能为空！");

}else {

Student student = studentDao.findById(Long.valueOf(tutor\_stu.getText()));

Teacher teacher = (student.getTeacher()==null ? new Teacher() : student.getTeacher());

teacher.setId(Long.valueOf(teacher\_id.getText()));

student.setTeacher(teacher);

studentDao.attachDirty(student);

this.showAllStudents();

}

}

public Session getSession() {

session = ScauCmiHibernateSessionFactoryUtil.getSession();

return session;

}

private void showWarningStage(String message) {

Alert warning = new Alert(Alert.AlertType.WARNING,message);

Button warnButton=new Button();

warnButton.setOnAction((ActionEvent)->{

warning.showAndWait();

});

warning.show();

}

private void showAllStudents() {

allStudents.clear();

List students = studentDao.findAll();

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

Student student;

while(studentIterator.hasNext()) {

student = (Student) studentIterator.next();

allStudents.setText(allStudents.getText()+"学生ID号："+student.getId()+", 学生名字："+student.getName()+", 指导老师ID："+(student.getTeacher() != null ? student.getTeacher().getId() : "无")+"\n");

}

allStudents2.setText(allStudents.getText());

}

private void showAllTeachers() {

allTeachers.clear();

List teachers = teacherDao.findAll();

Iterator<?> teacherIterator = teachers.iterator();

Teacher teacher;

while(teacherIterator.hasNext()) {

teacher = (Teacher) teacherIterator.next();

allTeachers.setText(allTeachers.getText()+"老师ID号："+teacher.getId()+", 老师名字："+teacher.getName()+"\n");

}

allTeachers2.setText(allTeachers.getText());

}

}

# 其他类

StudentDAO.java

package cn.edu.scau.cmi.longting.hibernate.dao;

import java.util.List;

import org.hibernate.LockOptions;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.Transaction;

import org.hibernate.criterion.Example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import cn.edu.scau.cmi.longting.hibernate.domain.Student;

import cn.edu.scau.cmi.longting.hibernate.util.ScauCmiHibernateSessionFactoryUtil;

public class StudentDao {

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

private static final String NAME = "name";

private Session session;

public void save(Student student) {

try {

this.getSession();

Transaction transaction = session.beginTransaction();

session.save(student);

transaction.commit();

} catch (RuntimeException re) {

throw re;

}

}

public void delete(Student student) {

log.debug("deleting Student instance");

try {

this.getSession();

Transaction transaction = session.beginTransaction();

session.delete(student);

transaction.commit();

log.debug("delete successful");

} catch (Exception e) {

log.error("delete faild", e);

throw e;

}

}

public Student findById(Long id) {

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

try {

Student instance = (Student) session.get("cn.edu.scau.cmi.longting.hibernate.domain.Student", id);

return instance;

} catch (RuntimeException re) {

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

throw re;

}

}

public List findByExample(Student student) {

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

try {

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

.add(Example.create(student)).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 student) {

log.debug("merging Student instance");

try {

this.getSession();

Transaction transaction = session.beginTransaction();

Student result = (Student) session.merge(student);

transaction.commit();

log.debug("merge successful");

return result;

} catch (RuntimeException re) {

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

throw re;

}

}

public void attachDirty(Student student) {

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

try {

this.getSession();

Transaction transaction = session.beginTransaction();

session.saveOrUpdate(student);

transaction.commit();

log.debug("attach successful");

} catch (RuntimeException re) {

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

throw re;

}

}

public void attachClean(Student student) {

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

try {

this.getSession();

Transaction transaction = session.beginTransaction();

session.buildLockRequest(LockOptions.NONE).lock(student);

transaction.commit();

log.debug("attach successful");

} catch (RuntimeException re) {

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

throw re;

}

}

public Session getSession() {

session = ScauCmiHibernateSessionFactoryUtil.getSession();

return session;

}

}

TeacherDAO.java

package cn.edu.scau.cmi.longting.hibernate.dao;

import java.util.List;

import org.hibernate.LockOptions;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.Transaction;

import org.hibernate.criterion.Example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import cn.edu.scau.cmi.longting.hibernate.domain.Student;

import cn.edu.scau.cmi.longting.hibernate.domain.Teacher;

import cn.edu.scau.cmi.longting.hibernate.util.ScauCmiHibernateSessionFactoryUtil;

public class TeacherDao {

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

private static final String NAME = "name";

private Session session;

public void save(Teacher teacher) {

this.getSession();

Transaction Transaction = session.beginTransaction();

session.save(teacher);

Transaction.commit();

}

public void delete(Teacher teacher) {

log.debug("deleting Student instance");

try {

this.getSession();

Transaction transaction = session.beginTransaction();

session.delete(teacher);

transaction.commit();

log.debug("delete successful");

} catch (Exception e) {

log.error("delete faild", e);

throw e;

}

}

public Teacher findById(Long id) {

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

try {

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

return instance;

} catch (RuntimeException re) {

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

throw re;

}

}

public List findByExample(Teacher teacher) {

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

try {

List results = getSession().createCriteria("cn.edu.scau.cmi.longting.hibernate.domain.Teacher")

.add(Example.create(teacher)).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 teacher) {

log.debug("merging Teacher instance");

try {

this.getSession();

Transaction transaction = session.beginTransaction();

Teacher result = (Teacher) session.merge(teacher);

transaction.commit();

log.debug("merge successful");

return result;

} catch (RuntimeException re) {

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

throw re;

}

}

public void attachDirty(Teacher teacher) {

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

try {

this.getSession();

Transaction transaction = session.beginTransaction();

session.saveOrUpdate(teacher);

transaction.commit();

log.debug("attach successful");

} catch (RuntimeException re) {

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

throw re;

}

}

public void attachClean(Teacher teacher) {

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

try {

this.getSession();

Transaction transaction = session.beginTransaction();

session.buildLockRequest(LockOptions.NONE).lock(teacher);

transaction.commit();

log.debug("attach successful");

} catch (RuntimeException re) {

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

throw re;

}

}

public Session getSession() {

session = ScauCmiHibernateSessionFactoryUtil.getSession();

return session;

}

}

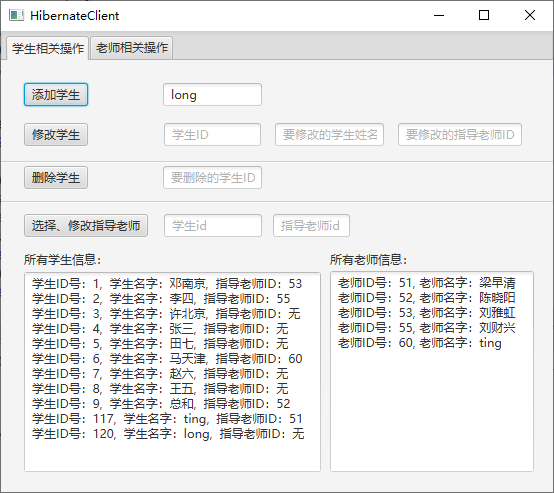
# 运行效果

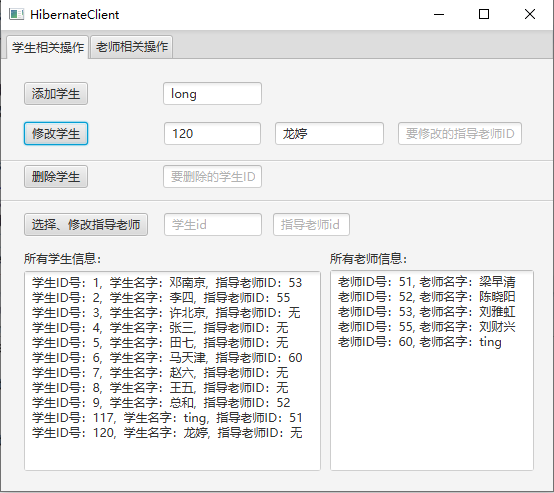
截屏展示运行的效果，注意适当的说明和图的编号

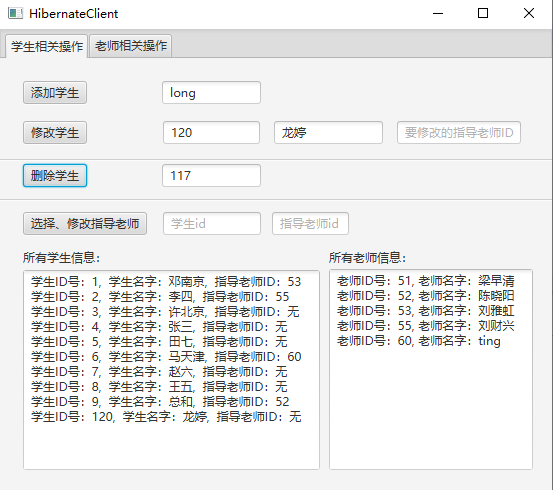
1. 初始界面，显示数据库中所有的学生、老师信息

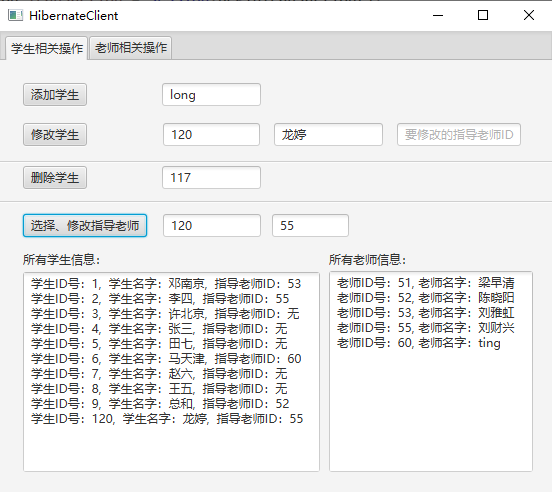


1. 下面依次展示增、删、改学生信息，以及修改学生对应的指导老师

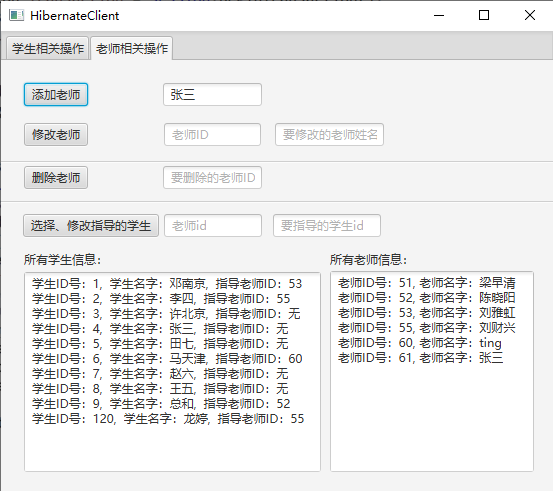


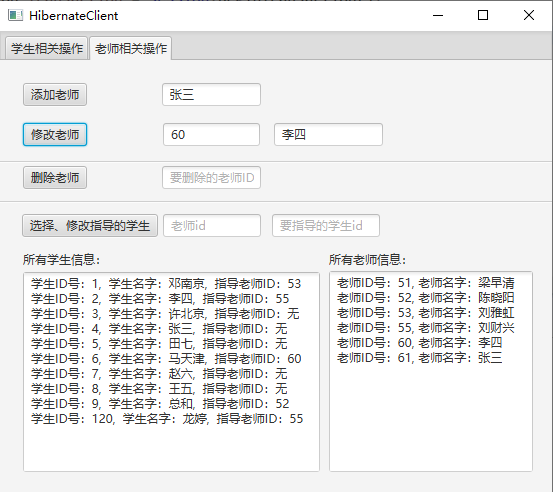


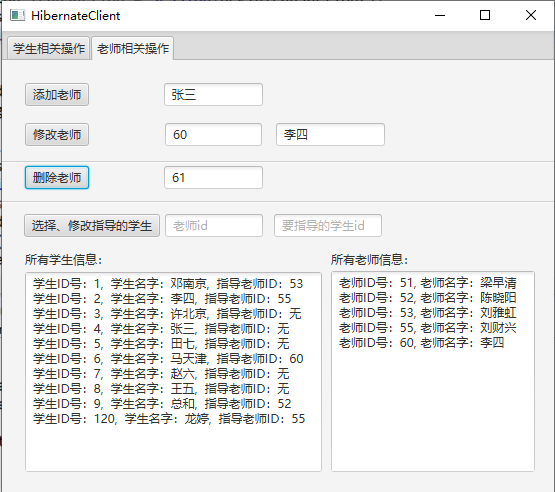


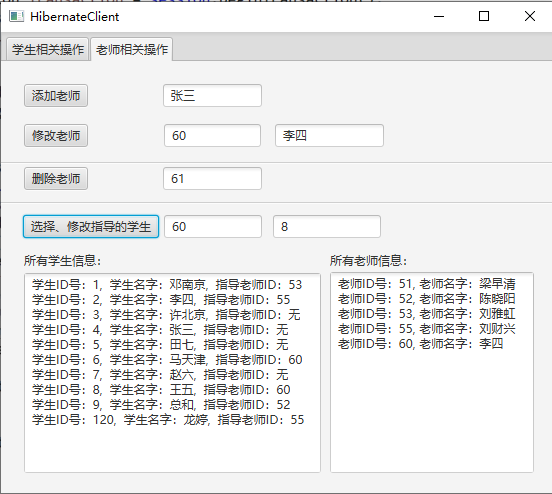


1. 下面依次展示增、删、改老师信息，以及修改老师对应指导的学生









1. 具体操作时，会适当有一些出错提示：

