# 实验二：ORM设计模式和DAO模式

# 项目建立

1. 全部代码在cn.edu.scau.cmi.zhangjiayi包下
2. 工具类代码在cn.edu.scau.cmi.zhangjiayi.util 包下，其中有在数据库操作中抽出来的数据库连接这一公共部分类DBconnect
3. DAO类放置在cn.edu.scau.cmi.zhangjiayi.dao 包下，其中有数据库StudentDAO,CourseDAO
4. 客户端类放置在cn.edu.scau.cmi.zhangjiayi.client包下，是程序的入口

# 实体类、DAO类、工具类、客户端类的设计

客户类、实体类、DAO类、数据库访问类的UML设计



图1 系统相关类图

# 实体类设计

根据实验一建立的需求和数据库，设计相应的实体类。建立的实体类放置到cn.edu.scau.cmi.zhangjiayi.domain.

将设计的实体类代码复制到下面的代码框中。

1. Student

public class Student {

private String id="";

private String name="";

private String className="";

private List<Course> courseList; //某个学生选了的课的列表

public Student(){

}

public Student(String id,String name,String className)

{

this.id=id;

this.name=name;

this.className=className;

}

public String getID()

{

return this.id;

}

public void setID(String id)

{

this.id = id;

}

public String getName()

{

return this.name;

}

public void setName(String name)

{

this.name=name;

}

public String getSchoolClassName()

{

return this.className;

}

public void setClassName(String className)

{

this.className=className;

}

public List<Course> getStdCourse()

{

return this.courseList;

}

public void setStdCourse(List<Course> courseList)

{

this.courseList = courseList;

}

}

1. Course

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

import java.util.List;

public class Course {

private String id;

private String name;

private Teacher teacher;

private String credits;

private List<Student> stdList; //选了某门课的全部学生

public Course(){

}

public Course(String id,String name,Teacher teacher,String credits)

{

this.id=id;

this.credits=credits;

this.teacher=teacher;

this.name=name;

}

public void setID(String id)

{

this.id=id;

}

public String getID()

{

return this.id;

}

public void setCredits(String credits)

{

this.credits=credits;

}

public String getCredits()

{

return this.credits;

}

public Teacher getTeacher()

{

return this.teacher;

}

public void setTeacher(Teacher teacher)

{

this.teacher=teacher;

}

public void setName(String name)

{

this.name=name;

}

public String getName()

{

return this.name;

}

public void setStdList(List<Student> stdList)

{

this.stdList=stdList;

}

public List<Student> getStdList()

{

return this.stdList;

}

}

1. Teacher

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

public class Teacher {

private String id;

private String name;

public String getID()

{

return this.id;

}

public void setID(String id)

{

this.id=id;

}

public String getName()

{

return this.name;

}

public void setName(String name)

{

this.name=name;

}

}

1. Score

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

public class Score {

private String id;

private Student student;

private Course course;

public String getID()

{

return this.id;

}

public void setID(String id)

{

this.id=id;

}

public Student getStudent()

{

return this.student;

}

public void setStudent(Student student)

{

this.student=student;

}

public Course getCourse()

{

return this.course;

}

public void setCourse(Course course)

{

this.course=course;

}

}

# DAO类设计

根据实验一建立的需求和数据库，设计相应的DAO类。建立DAO类放置到cn.edu.scau.cmi.zhangshan.dao。将设计的DAO类代码复制到下面的代码框中。

1. StudentDAO

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

import cn.edu.scau.cmi.zhangjiayi.domain.Course;

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

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

import cn.edu.scau.cmi.zhangjiayi.util.DBconnect;

import java.sql.\*;

import java.util.ArrayList;

public class StudentDAO {

/\*

\* 1.学生可以查询到自己的成绩 1对n

\* 2.学生可以查询到自己选的课程 n对n 现在只写这个

\* 3.增删查改学生

\*/

//增加学生

public void newStudent(Student student)

{

Connection conn = null;

PreparedStatement pstmt = null;

DBconnect dbconn = new DBconnect();

try{

//1.连接数据库

conn = dbconn.connOpen();

//conn.setAutoCommit(false);

//2.创建statement 编写sql语句

String sql;

sql="insert into student(s\_id,name,className) values(?,?,?)";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, student.getID());

pstmt.setString(2, student.getName());

pstmt.setString(3, student.getSchoolClassName());

System.out.println(pstmt.toString());

//3.执行sql语句

int count=pstmt.executeUpdate();

if(count>0)

{

//conn.commit();

System.out.println("添加学生成功！");

}

}

catch(SQLException e){

e.printStackTrace();

System.out.println("添加学生失败！");

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

}

}

//通过学号，获得该学生的所有信息

public Student findStudent(Student student)

{

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs=null;

DBconnect dbconn = new DBconnect();

//Student student = new Student();

try{

//1.连接数据库

conn = dbconn.connOpen();

//conn.setAutoCommit(false);

//2.创建statement 编写sql语句

String sql;

sql="select \* from student where s\_id= ?";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, student.getID());

System.out.println(pstmt.toString());

//3.执行sql语句

rs=pstmt.executeQuery();

while( rs.next() )

{

student.setID(student.getID());

student.setName(rs.getString("sname"));

student.setClassName(rs.getString("className"));

}

}

catch(SQLException e){

e.printStackTrace();

System.out.println("学生信息获取失败");

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

dbconn.closeResult(rs);

}

return student;

}

public void deleteStudent(Student student)

{

Connection conn = null;

PreparedStatement pstmt = null;

DBconnect dbconn = new DBconnect();

try{

//1.连接数据库

conn = dbconn.connOpen();

//conn.setAutoCommit(false);

String sql;

sql="delete from student where s\_id=?";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, student.getID());

//System.out.println(pstmt.toString());

int count=pstmt.executeUpdate();

if(count==1)

{

//conn.commit();

System.out.println("删除学生成功！");

}

}

catch(SQLException e){

e.printStackTrace();

System.out.println("删除学生失败！");

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

}

}

//查询学生选的所有课程后 把结果填充到student.courseList中

public Student queryCourse(Student student)

{

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

ArrayList<Course> stdls=new ArrayList<Course>();

DBconnect dbconn = new DBconnect();

try{

//1.连接数据库

conn = dbconn.connOpen();

conn.setAutoCommit(false);

//2.创建statement 编写sql语句

String sql="select \* from takes natural join student natural join course where s\_id= ?";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, student.getID());

System.out.println(pstmt.toString());

//3.执行sql语句

rs=pstmt.executeQuery();

while( rs.next() )

{

Course aCourse = new Course();

aCourse.setName(rs.getString("cname"));

aCourse.setID(rs.getString("course\_id"));

Teacher aTeacher = new Teacher();

aTeacher.setID(rs.getString("t\_id"));

aCourse.setTeacher(aTeacher);

stdls.add(aCourse);

//student.setStdCourse();

//System.out.println("添加学生成功！");

}

student.setStdCourse(stdls);

}

catch(SQLException e){

e.printStackTrace();

//System.out.println("添加学生失败！");

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

dbconn.closeResult(rs);

}

return student;

}

}

1. CourseDAO

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

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import cn.edu.scau.cmi.zhangjiayi.domain.Course;

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

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

import cn.edu.scau.cmi.zhangjiayi.util.DBconnect;

public class CourseDAO {

/\*选课

\* 一门课程可以被多个学生选择 n对n

\*1.可以学生选课 insert takes

\*2.可以增删查改课程 course

\*3.可以查询选了某门课程的所有学生 takes student course

\*/

//添加新课程

public void newCourse(Course course, Teacher teacher)

{

Connection conn = null;

PreparedStatement pstmt = null;

DBconnect dbconn = new DBconnect();

try{

//1.连接数据库

conn = dbconn.connOpen();

//conn.setAutoCommit(false);

String sql;

sql="insert into Course(course\_id,cname,t\_id,credits) values(?,?,?,?)";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, course.getID());

pstmt.setString(2, course.getName());

pstmt.setString(3, teacher.getID());

pstmt.setString(4, course.getCredits());

System.out.println(pstmt.toString());

//3.执行sql语句

int count=pstmt.executeUpdate();

if(count>0)

{

//conn.commit();

System.out.println("添加课程成功！");

}

}

catch(SQLException e){

e.printStackTrace();

System.out.println("添加课程失败！");

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

}

}

//通过课程id查找课程

public Course findCoures(Course course)

{

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs=null;

DBconnect dbconn = new DBconnect();

//Course course = new Course();

try{

conn = dbconn.connOpen();

String sql;

sql="select \* from course where course\_id= ?";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, course.getID());

System.out.println(pstmt.toString());

rs=pstmt.executeQuery();

while( rs.next() )

{

course.setID(rs.getString("course\_id"));

course.setName(rs.getString("cname"));

Teacher teacher = new Teacher();

TeacherDAO teacherDAO = new TeacherDAO();

teacher = teacherDAO.findTeacher(rs.getString("t\_id"));

course.setTeacher(teacher);

course.setCredits(rs.getString("credits"));

}

}

catch(SQLException e){

e.printStackTrace();

System.out.println("学生信息获取失败");

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

dbconn.closeResult(rs);

}

return course;

}

public void deleteCourse(Course course)

{

Connection conn = null;

PreparedStatement pstmt = null;

DBconnect dbconn = new DBconnect();

try{

//1.连接数据库

conn = dbconn.connOpen();

String sql;

sql="delete from course where course\_id=?";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, course.getID());

System.out.println(pstmt.toString());

int count=pstmt.executeUpdate();

if(count==1)

{

System.out.println("删除课程成功！");

}

}

catch(SQLException e){

e.printStackTrace();

System.out.println("删除课程失败！");

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

}

}

public Course queryStudent(Course course)

{

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

ArrayList<Student> cslist=new ArrayList<Student>();

DBconnect dbconn = new DBconnect();

try{

//1.连接数据库

conn = dbconn.connOpen();

conn.setAutoCommit(false);

//2.创建statement 编写sql语句

String sql="select \* from takes natural join student natural join course where course\_id= ?";

pstmt=conn.prepareStatement(sql);

pstmt.setString(1, course.getID());

System.out.println(pstmt.toString());

//3.执行sql语句

rs=pstmt.executeQuery();

while( rs.next() )

{

Student astd = new Student();

astd.setName(rs.getString("sname"));

astd.setID(rs.getString("s\_id"));

astd.setClassName(rs.getString("className"));

cslist.add(astd);

}

course.setStdList(cslist);

}

catch(SQLException e){

e.printStackTrace();

}

finally{

dbconn.closeState(pstmt);

dbconn.connClose(conn);

dbconn.closeResult(rs);

}

return course;

}

}

# 数据库访问的公共类设计

根据课堂所讲的思想，设计相应的数据库防伪的公共类。将设计的类代码复制到下面的代码框中。

DBconnect

package cn.edu.scau.cmi.zhangjiayi.util;

import java.sql.\*;

public class DBconnect {

private String driver = "com.mysql.jdbc.Driver";

private String url="jdbc:mysql://localhost:3306/db\_scoresystem";

private String username="root";

private String password="1234";

//加载驱动

public DBconnect()

{

try

{

Class.forName(driver) ;

System.out.println("驱动加载成功");

}

catch(ClassNotFoundException e)

{

System.out.println("找不到驱动程序类 ，加载驱动失败！");

e.printStackTrace() ;

}

}

//连接数据库

public Connection connOpen()

{

Connection conn=null;

try{

conn = DriverManager.getConnection(url,username,password);

System.out.println("数据库连接成功！");

}

catch(Exception e)

{

System.out.println("数据库连接失败！");

e.printStackTrace();

}

return conn;

}

public void closeResult(ResultSet rs){

if(rs != null)

try{

rs.close();

}

catch(SQLException e){

e.printStackTrace();

}

}

public void closeState(Statement stmt)

{

if(stmt != null)

try{

stmt.close();

}

catch(SQLException e){

e.printStackTrace();

}

}

public void connClose(Connection conn)

{

if(conn != null)

{

try{

if(!conn.isClosed() )

{

conn.close();

System.out.println("连接已关闭");

}

}

catch(SQLException e)

{

e.printStackTrace();

}

}

}

}

# 客户端设计

将客户端代码复制到下面的框中

public class scoreSystem {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("Hello");

//添加学生std1

Student std1 = new Student("201527010524","jiayizzz","15软工R5班");

StudentDAO std1DAO = new StudentDAO();

std1DAO.newStudent(std1);

//查询学号为“201527010523”的同学所选的所有课程

Student std2 = new Student();

String id="201527010523";

std2.setID(id);

StudentDAO stdDAO = new StudentDAO();

std2=stdDAO.queryCourse(std2);//返回的是一个student类型 的list<Course>

for (int i = 0; i < std2.getStdCourse().size(); i++) {

System.out.println(std2.getStdCourse().get(i).getName() );

}

//查询学号为“201527010523”的同学的信息

std2=stdDAO.findStudent(id);

System.out.println("姓名："+std2.getName());

System.out.println("班级："+std2.getSchoolClassName());

//删除学号为“201527010524”的同学

stdDAO.deleteStudent(stdDAO.findStudent("201527010524"));\*/

//添加课程

CourseDAO CSDAO = new CourseDAO();

Teacher teacher = new Teacher();

teacher.setID("000001");

Course course1 = new Course("100004","软件体系结构",teacher,"2.5");

CSDAO.newCourse(course1,teacher);

//查询课程“100005”

Course course2 = new Course();

course2 = CSDAO.findCoures("100005");

System.out.println("课程编号："+ course2.getID());

System.out.println("课程名称："+course2.getName());

System.out.println("课程教师："+course2.getTeacher().getID());

System.out.println("课程学分："+course2.getCredits() );

//删除课程

CSDAO.deleteCourse(course2);

//查询所有选择了‘100004’这门课程的所有学生

Course course3 = new Course();

String id="100004";

course3.setID(id);

course3=CSDAO.queryStudent(course3);//返回的是一个student类型 的list<Course>

for (int i = 0; i < course3.getStdList().size(); i++) {

System.out.println(course3.getStdList().get(i).getName() );

}

}

}

# 运行效果

**7.1 添加学生**

Student std1=**new** Student("201527010524","jiayizzz","15软工R5班");

StudentDAO std1DAO = **new** StudentDAO();

std1DAO.newStudent(std1);

在执行后，可以在数据库中看到student表中多了一个行数据



**7.2 通过学号，获得学生的所有信息**

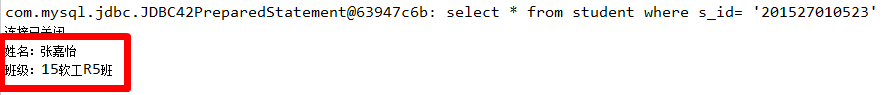
测试代码：

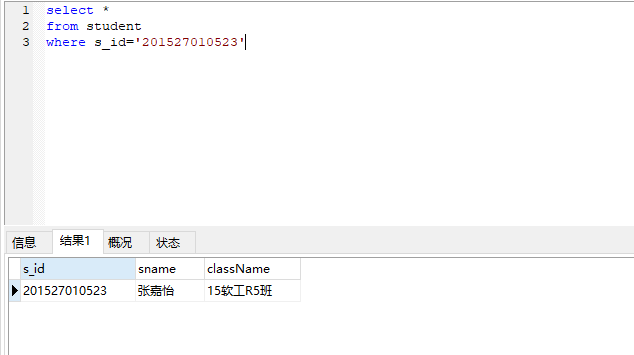
//查询学号为“201527010523”的同学的信息

std2=std2DAO.findStudent(id);

System.***out***.println("姓名："+std2.getName());

System.***out***.println("班级："+std2.getSchoolClassName());

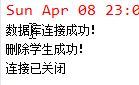




**7.3 删除学生**

//删除学号为“201527010524”的同学

stdDAO.deleteStudent(stdDAO.findStudent("201527010524"));



|  |  |
| --- | --- |
| 删除前： | 删除后： |
|  |  |

**7.4 查询学生所选的所有课程**

测试代码：

查询学号为“201527010523”的同学所选的所有课程

Student std2 = **new** Student();

std2.setID("201527010523");

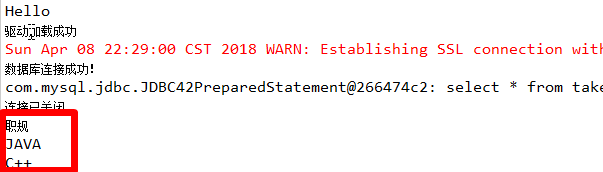
StudentDAO std2DAO = **new** StudentDAO();

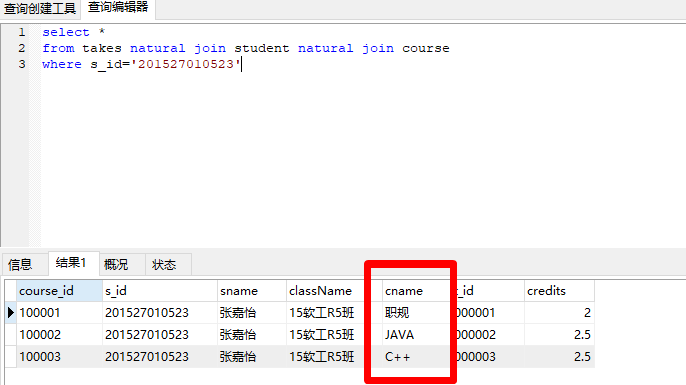
std2=std2DAO.queryCourse(std2);//返回的是一个student类型 的list<Course>

**for** (**int** i = 0; i < std2.getStdCourse().size(); i++) {

System.***out***.println(std2.getStdCourse().get(i).getName() );

}





**7.5添加课程**

测试代码：

//添加课程

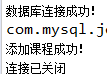
Teacher teacher = **new** Teacher();

teacher.setID("000001");

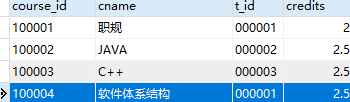
Course course1 = **new** Course("100004","软件体系结构",teacher,"2.5");

CourseDAO CSDAO = **new** CourseDAO();

CSDAO.newCourse(course1,teacher);

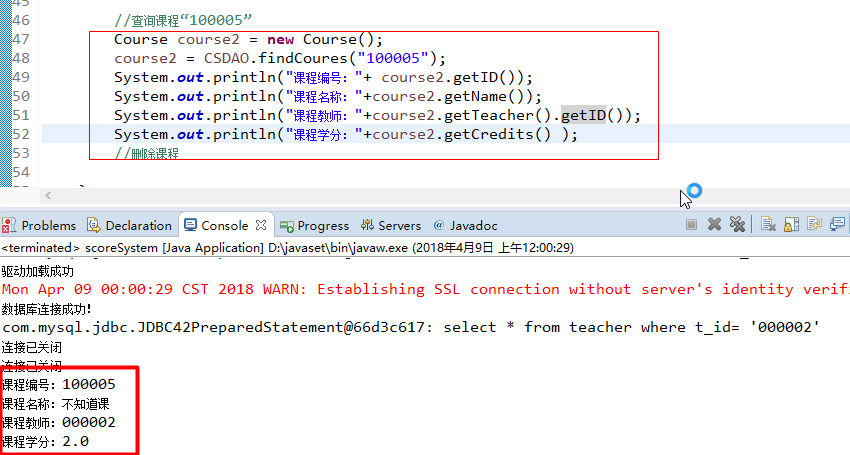


在控制台显示课程添加成功后，在数据库中看到已经多了新添加的记录



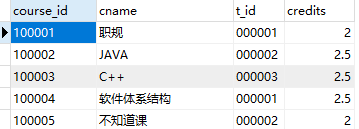
**7.6 查询课程信息**

查询课程编号为”100005”的课程信息

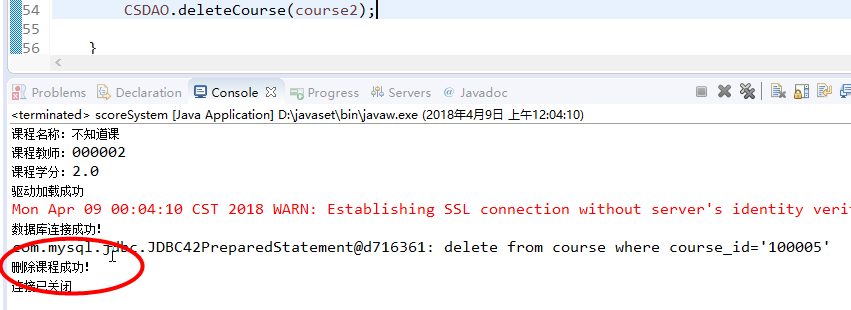


**7.7删除课程**

初始课程：



控制台输出：



删除后数据库内容：



**7.8查询课程的所有学生列表**

所有选择了‘100004’这么课程的学生

