**JAVA编程进阶上机报告**

****

上机作业

**学 院\_智能与计算学部\_**

**专 业\_\_ 软件工程\_\_\_\_\_**

**姓 名\_\_\_ 李济孚\_\_ \_ \_**

**学 号\_\_ 3018216260\_\_\_**

**年 级\_\_\_\_ 2018\_\_\_\_\_\_**

**班 级\_\_\_\_\_\_ 5\_\_\_\_\_\_\_\_**

# 一、实验要求

## 实验目的

使用注解和反射来完成动态Sql编程

## 实验要求

1. **提供用户表：user**

表中包含字段：

id，用户名，性别，邮箱，电话等信息。

1. **要求通过注解和反射的方式封装一个小型的sql操作类，可以通过对应的方法生成增、删、改、查等操作的SQL语句。**
2. **要求实现注解**：

@Column：用来标注每个field对应的表中的字段是什么  
@Table：用来标记表的名字

# 二、源代码

Column类:

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Target({ElementType.FIELD})

@Retention(RetentionPolicy.RUNTIME)

public @interface Column {

String value();

}

Table类：

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Target({ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

public @interface Table {

String value();

}

User类:

import Annotation\_Reflection.\*;

@Table("user")

public class User {

@Column("id")

private int id;

private boolean isIdSet = false;

@Column("username")

private String username;

@Column("age")

private int age;

private boolean isAgeSet = false;

@Column("email")

private String email;

@Column("telephone")

private String telephone;

public int getid()

{

return id;

}

public String getusername()

{

return this.username;

}

public int getage()

{

return age;

}

public String getemail()

{

return this.email;

}

public String gettelephone()

{

return this.telephone;

}

public void setid(int i)

{

this.id = i;

this.isIdSet = true;

}

public void setusername(String i)

{

this.username = i;

}

public void setage(int i)

{

this.age = i;

this.isAgeSet = true;

}

public void setemail(String i)

{

this.email = i;

}

public void settelephone(String i)

{

this.telephone = i;

}

}

SqlUtil类：

import java.lang.reflect.Field;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class SqlUtil {

private StringBuilder stringbuilder = new StringBuilder();

private int flag = 0; //判断是否是table类型，不是则拒绝调用

private Class<? extends Object> cl; //类对象

private Field[] fields;

private boolean field\_flag;

private Column column;

private String field\_name;

private Method menthod;

private Object val;

private Object object;

public String query(Object object) throws NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException

{

StringBuilder sb = new StringBuilder();

Class j = object.getClass();

if(j.isAnnotationPresent(Table.class))

{

cl = j;

flag = 1;

fields = cl.getDeclaredFields();

this.object = object;

}

if(flag == 1)

{

Table table = (Table) cl.getDeclaredAnnotation(Table.class);

sb.append("SELECT \* FROM ").append(table.value());

for(Field i : fields)

{

field\_flag = i.isAnnotationPresent(Column.class);

if (!field\_flag) {

continue;

}

column = i.getDeclaredAnnotation(Column.class);

field\_name = i.getName();

menthod = cl.getDeclaredMethod("get" + field\_name);

val = menthod.invoke(object);

// 判断该字段是否为空，不为空，才进行append

if (!Objects.isNull(val)) {

if((val instanceof Integer) && (Integer) val == 0){

continue;

}

sb.append(" AND ").append(column.value()).append(" = ");

sb.append(val);

}

}

}

return sb.toString();

}

public String insert(Object object) throws NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException

{

if(object.getClass() == ArrayList.class)

{

int count = 0;

StringBuilder sb = new StringBuilder();

String str = "";

StringBuilder temp = new StringBuilder();

for(Object ob : (List)object)

{

Class j = ob.getClass();

if(j.isAnnotationPresent(Table.class))

{

cl = j;

flag = 1;

fields = cl.getDeclaredFields();

this.object = ob;

}

if(flag == 1)

{

Table table = (Table) cl.getDeclaredAnnotation(Table.class);

if(count == 0)

{

sb.append("INSERT INTO ").append(table.value()).append("(");

for(Field i : fields)

{

field\_flag = i.isAnnotationPresent(Column.class);

if (!field\_flag) {

continue;

}

column = i.getDeclaredAnnotation(Column.class);

field\_name = i.getName();

menthod = cl.getDeclaredMethod("get" + field\_name);

val = menthod.invoke(ob);

// 判断该字段是否为空，不为空，才进行append

if (!Objects.isNull(val)) {

if((val instanceof Integer) && (Integer) val == 0){

continue;

}

sb.append(field\_name + ", ");

}

}

str += sb.toString().substring(0, sb.length()-2);

str += ") VALUES (";

count++;

}

for(Field i : fields)

{

field\_flag = i.isAnnotationPresent(Column.class);

if (!field\_flag) {

continue;

}

column = i.getDeclaredAnnotation(Column.class);

field\_name = i.getName();

menthod = cl.getDeclaredMethod("get" + field\_name);

val = menthod.invoke(ob);

// 判断该字段是否为空，不为空，才进行append

if (!Objects.isNull(val)) {

if((val instanceof Integer) && (Integer) val == 0){

continue;

}

temp.append(val + ", ");

}

}

temp.delete(temp.lastIndexOf(","), temp.length());

temp.append("), (");

}

}

str += temp.toString().substring(0, temp.length()-3);

return str;

}

else

{

StringBuilder sb = new StringBuilder();

String str = "";

StringBuilder temp = new StringBuilder();

Class j = object.getClass();

if(j.isAnnotationPresent(Table.class))

{

cl = j;

flag = 1;

fields = cl.getDeclaredFields();

this.object = object;

}

if(flag == 1)

{

Table table = (Table) cl.getDeclaredAnnotation(Table.class);

sb.append("INSERT INTO ").append(table.value()).append("(");

temp.append(") VALUES (");

for(Field i : fields)

{

field\_flag = i.isAnnotationPresent(Column.class);

if (!field\_flag) {

continue;

}

column = i.getDeclaredAnnotation(Column.class);

field\_name = i.getName();

menthod = cl.getDeclaredMethod("get" + field\_name);

val = menthod.invoke(object);

// 判断该字段是否为空，不为空，才进行append

if (!Objects.isNull(val)) {

if((val instanceof Integer) && (Integer) val == 0){

continue;

}

sb.append(field\_name + ", ");

temp.append(val + ", ");

}

}

str += sb.toString().substring(0, sb.length()-2);

str += temp.toString().substring(0, temp.length()-2);

}

return str+")";

}

}

public String delete(Object object) throws NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException

{

StringBuilder sb = new StringBuilder();

Class j = object.getClass();

if(j.isAnnotationPresent(Table.class))

{

cl = j;

flag = 1;

fields = cl.getDeclaredFields();

this.object = object;

}

if(flag == 1)

{

Table table = (Table) cl.getDeclaredAnnotation(Table.class);

sb.append("DELETE FROM ").append(table.value());

for(Field i : fields)

{

field\_flag = i.isAnnotationPresent(Column.class);

if (!field\_flag) {

continue;

}

column = i.getDeclaredAnnotation(Column.class);

field\_name = i.getName();

menthod = cl.getDeclaredMethod("get" + field\_name);

val = menthod.invoke(object);

// 判断该字段是否为空，不为空，才进行append

if (!Objects.isNull(val)) {

if((val instanceof Integer) && (Integer) val == 0){

continue;

}

sb.append(" WHERE ").append(column.value()).append(" = ");

sb.append(val);

}

}

}

return sb.toString();

}

public String update(Object object) throws NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException

{

StringBuilder sb = new StringBuilder();

int count = 0;

Class j = object.getClass();

if(j.isAnnotationPresent(Table.class))

{

cl = j;

flag = 1;

fields = cl.getDeclaredFields();

this.object = object;

}

if(flag == 1)

{

Table table = (Table) cl.getDeclaredAnnotation(Table.class);

sb.append("UPADTE ").append(table.value());

for(Field i : fields)

{

field\_flag = i.isAnnotationPresent(Column.class);

if (!field\_flag) {

continue;

}

column = i.getDeclaredAnnotation(Column.class);

field\_name = i.getName();

menthod = cl.getDeclaredMethod("get" + field\_name);

val = menthod.invoke(object);

// 判断该字段是否为空，不为空，才进行append

if (!Objects.isNull(val) && count == 0) {

if((val instanceof Integer) && (Integer) val == 0){

continue;

}

sb.append(" SET ").append(column.value()).append(" = ");

sb.append(val + " ");

count++;

}

else if(!Objects.isNull(val) && count == 1)

{

if((val instanceof Integer) && (Integer) val == 0){

continue;

}

sb.append("WHERE ").append(column.value()).append(" = ");

sb.append(val);

}

}

}

return sb.toString();

}

}

Main类：

import java.lang.reflect.InvocationTargetException;

import java.util.ArrayList;

import java.util.List;

public class Main {

public static void main(String[] args) throws NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException

{

SqlUtil util = new SqlUtil();

// test query1

User user = new User();

user.setid(175);

System.out.println(util.query(user));

// print: SELECT \* FROM user WHERE id = 175

// test query2

user = new User();

user.setusername("史荣贞");

System.out.println(util.query(user));

// print: SELECT \* FROM `user` WHERE `username` LIKE '史荣贞';

// test insert

user = new User();

user.setusername("user");

user.settelephone("12345678123");

user.setemail("user@123.com");

user.setage(20);

System.out.println(util.insert(user));

// print: INSERT INTO `user` (`username`, `telephone`, `email`, `age`) VALUES ('user', '12345678123', 'user@123.com', 20)

// test insert list

User user2 = new User();

user2.setusername("user2");

user2.settelephone("12345678121");

user2.setemail("user2@123.com");

user2.setage(20);

List<User> list = new ArrayList<>();

list.add(user);

list.add(user2);

System.out.println(util.insert(list));

// print: INSERT INTO `user` (`username`, `telephone`, `email`, `age`) VALUES ('user', '12345678123', 'user@123.com', 20), ('user2', '12345678121', 'user2@123.com', 20)

// test update

user = new User();

user.setid(1);

user.setemail("change@123.com");

System.out.println(util.update(user));

// print: UPDATE `user` SET `email` = 'change@123.com' WHERE `id` = 1;

// test delete

user = new User();

user.setid(1);

System.out.println(util.delete(user));

// print: DELETE FROM `user` WHERE `id` = 1;

}

}

# 三、运行结果

