**JAVA编程进阶上机报告**

****

第二次上机作业

**Lab3 Annotation and Reflection**

**学 院 智能与计算学部**

**专 业 软件工程**

**班 级 5班**

**学 号 3018216232**

**姓 名 刘昕**

1. **实验目的**

使用注解和反射来完成动态Sql编程

1. **实验要求**
2. 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

1. 实现对应的接口并进行测试：注意这里只是需要大家把sql语句打印出来即可，不需要进行真正的数据库交互
2. **实验过程**
3. 实现注解：

Column.java 描述用户属性值字段的注解

**package** java;

**import** **static** java.lang.annotation.ElementType.FIELD;

**import** **static** java.lang.annotation.RetentionPolicy.RUNTIME;

**import** java.lang.annotation.\*;

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

**public** **@interface** Column {

String value() **default** "default";

}

Table.java 表的名字

**package** java;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.\*;

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.TYPE)

**public** **@interface** Table {

String value() **default** "default";

}

1. 创建User.java 其get方法用反射实现

**package** java;

@Table("user")

**public** **class** User{

@Column("id")

**private** **int** id;

@Column("username")

**private** String username;

@Column("age")

**private** **int** age;

@Column("email")

**private** String email;

@Column("telephone")

**private** String telephone;

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

**this**.id = id;

}

**public** **void** setUsername(String userName) {

**this**.username = userName;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** **void** setTelephone(String telephone) {

**this**.telephone = telephone;

}

}

1. 创建接口 sqlUtil.java 注释在代码中

**package** java;

**import** java.util.List;

**public** **interface** SqlUtil {

// 根据传入的参数返回查询语句

String query(User user);

// 根据传入的参数返回插入语句

String insert(User user);

// 根据传入的参数返回插入语句

String insert(List<User> users);

// 根据传入的参数返回删除语句(删除id为user.id的记录)

String delete(User user);

// 根据传入的参数返回更新语句(将id为user.id的记录的其他字段更新成user中的对应值)

String update(User user);

}

1. 实现接口 MySql.java 具体注释在代码中

**package** java;

**import** java.lang.reflect.Field;

**import** java.util.List;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**public** **class** MySql **implements** SqlUtil{

**public** ArrayList<Field> getFields(User user) {

List<String> columns = **new** ArrayList();

List<String> values = **new** ArrayList();

ArrayList<Field> fields= **new** ArrayList<Field>(Arrays.asList(User.**class**.getFields()));

ArrayList<Field> fieldsPrivate = **new** ArrayList<Field>(Arrays.asList(User.**class**.getDeclaredFields()));

fields.addAll(fieldsPrivate);

**return** fields;

}

**public** String getValue(Field field, User user) {

field.setAccessible(**true**);

String result;

**try** {

Object val = field.get(user);

**if**(val == **null**) **return** **null**;

result = val **instanceof** String?(String)val:String.valueOf(val);

}**catch**(Exception e) {

e.printStackTrace();

result = **null**;

}

**return** result;

}

// 根据传入的参数返回查询语句

**public** String query(User user) {

ArrayList<String> conditions = **new** ArrayList<String>();

ArrayList<Field> fields = getFields(user);

**for**(Field field: fields) {

Column c = field.getAnnotation(Column.**class**);

**if**(c!=**null**) {

String condition = **null**;

String key = c.value();

field.setAccessible(**true**);

**try** {

Object val = field.get(user);

**if**(val!=**null** && val **instanceof** String) {

condition = field.getName() + " LIKE '"+(String)val+"'";

conditions.add(condition);

}**else** **if**(val != **null** && !val.equals(0)){

condition = field.getName() + "='" + String.valueOf(val);

conditions.add(condition);

}

}**catch**(Exception e) {

e.printStackTrace();

}

}

}

String condition = String.join(" AND ", conditions);

String sql = "SELECT \* FROM %s WHERE %s;";

Table table = user.getClass().getAnnotation(Table.**class**);

**return** table==**null**? **null**: String.format(sql, table.value(), condition);

}

// 根据传入的参数返回插入语句

**public** String insert(User user) {

String tableName = user.getClass().getAnnotation(Table.**class**).value();

List<String> columns = **new** ArrayList();

List<String> values = **new** ArrayList();

ArrayList<Field> fields = getFields(user);

**for**(Field field: fields) {

Column c = field.getAnnotation(Column.**class**);

**if**(c!=**null**){

columns.add(c.value());

String value = getValue(field, user)==**null**?"null":getValue(field, user);

values.add("'"+(value)+"'");

}

}

String columnsName = String.join(", ", columns);

String value = String.join(", ", values);

String sql = "INSERT INTO %s(%s) VALUES(%s);";

**return** String.format(sql, tableName, columnsName, value);

}

// 根据传入的参数返回插入语句

**public** String insert(List<User> users) {

User user = users.get(0);

String tableName = user.getClass().getAnnotation(Table.**class**).value();

ArrayList<Field> fields = getFields(user);

List<String> columns = **new** ArrayList<String>();

**for**(Field field: fields) {

Column c = field.getAnnotation(Column.**class**);

**if**(c!=**null**){

columns.add(c.value());

}

}

String fieldNames = String.join(", ",columns);

ArrayList<String> totalValues = **new** ArrayList<String>();

**for**(User user2: users) {

ArrayList<Field> fields2 = getFields(user2);

ArrayList<String> values2 = **new** ArrayList<String>();

**for**(Field field2: fields2) {

**if**(field2.getAnnotation(Column.**class**)!=**null**){

values2.add("'"+ getValue(field2, user2)==**null**?"null":getValue(field2, user2) +"'");

}

}

totalValues.add("("+String.join(", ",values2) +")");

}

**return** String.format("INSERT INTO %s(%s) VALUES%s'", tableName, fieldNames, String.join(", ", totalValues));

}

// 根据传入的参数返回删除语句(删除id为user.id的记录)

**public** String delete(User user) {

String tableName = user.getClass().getAnnotation(Table.**class**).value();

ArrayList<Field> fields = getFields(user);

String where="";

**for**(Field field: fields) {

Column c = field.getAnnotation(Column.**class**);

**if**(c!=**null**) {

**if**(field.getName().equals("id")) {

where = field.getName()+"="+getValue(field, user);

**break**;

}

}

}

**return** String.format("DELETE FROM %s WHERE %s;", tableName, where);

}

// 根据传入的参数返回更新语句(将id为user.id的记录的其他字段更新成user中的对应值)

**public** String update(User user) {

String tableName = user.getClass().getAnnotation(Table.**class**).value();

ArrayList<Field> fields = getFields(user);

String where="";

ArrayList<String> datas = **new** ArrayList<String>();

**for**(Field field: fields) {

Column c = field.getAnnotation(Column.**class**);

**if**(c!=**null** && getValue(field, user)!=**null**) {

**if**(field.getName().equals("id")) {

where = field.getName()+"="+getValue(field, user);

}**else** {

datas.add(field.getName()+"='"+getValue(field, user)+"'");

}

}

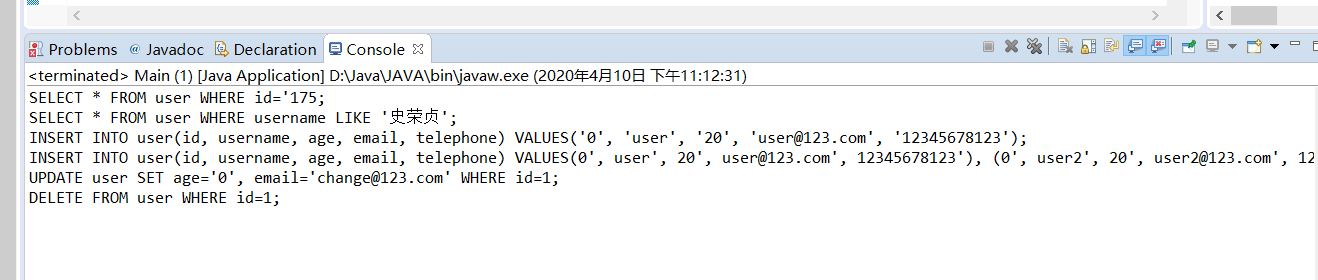
}

**return** String.format("UPDATE %s SET %s WHERE %s;", tableName, String.join(", ", datas), where);

}

}

**实验结果：**

****