学 院　智能与计算

专 业　软件工程

班 级　三班

学 号　3018216144

姓 名　王文君

一、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。

2. **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。

3. **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

```
package reflection;

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.List;

//MySqlUtil类实现接口SqlUtil
public class MySqlUtil implements SqlUtil{

    @Override
    public String query(User user) {
        StringBuffer sql = new StringBuffer();
        Class  c = user.getClass();
    // 判断该类是否包含Table的注解，返回true/false
        boolean flag = c.isAnnotationPresent(Table.class);
        if(!flag){
            return null;
        }
     // 获取到注解对象
        Table table = (Table) c.getAnnotation(Table.class);
     // 获取到注解对象的值
        String tableName = table.value();

        sql.append("select * from ").append(tableName).append(" where");

        Field[] fields = c.getDeclaredFields();
        for (int i = 0; i < fields.length; i++) {
            boolean fieldFalg = fields[i].isAnnotationPresent(Column.class);
            if(!fieldFalg){
                continue;
            }
            Column column = fields[i].getAnnotation(Column.class);
            // 字段的名称
            String fieldName = column.value();
            // 字段的值
            Object fieldValue = null;
            // 获取方法名
            String methodName =
"get"+fieldName.substring(0,1).toUpperCase()+fieldName.substring(1).toString();
            Method method;
            // 字段的值获取,使用反射
            try {
                method = c.getMethod(methodName);
                // 通过反射invoke获取到值
                fieldValue = method.invoke(user,null);
            }  catch (NoSuchMethodException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
```

```
            } catch (InvocationTargetException e) {
                e.printStackTrace();
            }
            if(fieldValue == null || (fieldValue instanceof Integer &&
(Integer)fieldValue == 0)){
                continue;
            }
            if(fieldValue instanceof String){
                sql.append(" '").append(fields[i].getName()).append("' like
'").append(fieldValue).append("'");
            }else if(fieldValue instanceof Integer){
                sql.append("  ").append(fields[i].getName()).append(" =
").append(fieldValue);
            }

        }
        return sql.toString();
    }

    @Override
    public String insert(User user) {
        // TODO Auto-generated method stub
        StringBuffer sql = new StringBuffer();
        StringBuffer name = new StringBuffer();
        name.append("(");
        StringBuffer value = new StringBuffer();
        value.append("(");
        Class  c = user.getClass();
        boolean flag = c.isAnnotationPresent(Table.class);
        if(!flag){
            return null;
        }
        Table table = (Table) c.getAnnotation(Table.class);
        String tableName = table.value();

        sql.append("INSERT INTO '").append(tableName).append("' ");

        Field[] fields = c.getDeclaredFields();
        for (int i = 0; i < fields.length; i++) {
            boolean fieldFalg = fields[i].isAnnotationPresent(Column.class);
            if(!fieldFalg){
                continue;
            }
            Column column = fields[i].getAnnotation(Column.class);
            String fieldName = column.value();
            Object fieldValue = null ;
            String methodName =
"get"+fieldName.substring(0,1).toUpperCase()+fieldName.substring(1).toString();
            Method method;
            try {
                method = c.getMethod(methodName);
                 fieldValue = method.invoke(user,null);
            }  catch (NoSuchMethodException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            } catch (InvocationTargetException e) {
                e.printStackTrace();
```

```
            }
            if(fieldValue == null || (fieldValue instanceof Integer &&
(Integer)fieldValue == 0)){
                continue;
            }
               name.append(" '").append(fields[i].getName()).append("' ,");
               if(fieldValue instanceof String)
                   value.append(" '").append(fieldValue).append("' ,");
               else if(fieldValue instanceof Integer)
                   value.append(" ").append(fieldValue).append(" ,");

        }
        name.deleteCharAt(name.length() - 1).append(")");
        value.deleteCharAt(value.length() - 1).append(")");
        return sql.toString()+name.toString()+" VALUES "+value.toString();
    }

    @Override
    public String insert(List<User> users) {
        // TODO Auto-generated method stub
        StringBuffer sql = new StringBuffer();
        StringBuffer name = new StringBuffer();
        name.append("(");
        StringBuffer value = new StringBuffer();

        for(int i=0;i<users.size();i++)
        {
            User user=users.get(i);
            value.append("(");
            Class  c = user.getClass();
            boolean flag = c.isAnnotationPresent(Table.class);
            if(!flag){
                return null;
            }
            Table table = (Table) c.getAnnotation(Table.class);
            String tableName = table.value();
            if(sql.length()==0)
            sql.append("INSERT INTO '").append(tableName).append("' ");
            Field[] fields = c.getDeclaredFields();
            for (int j = 0; j < fields.length; j++) {
                boolean fieldFalg = fields[j].isAnnotationPresent(Column.class);
                if(!fieldFalg){
                    continue;
                }
                Column column = fields[j].getAnnotation(Column.class);
                String fieldName = column.value();
                Object fieldValue = null ;
                String methodName =
"get"+fieldName.substring(0,1).toUpperCase()+fieldName.substring(1).toString();
                Method method;
                try {
                    method = c.getMethod(methodName);
                     fieldValue = method.invoke(user,null);
                }  catch (NoSuchMethodException e) {
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    e.printStackTrace();
                } catch (InvocationTargetException e) {
```

```
                    e.printStackTrace();
                }
                if(fieldValue == null || (fieldValue instanceof Integer &&
(Integer)fieldValue == 0)){
                    continue;
                }
                if(i==0)
                   name.append(" '").append(fields[j].getName()).append("' ,");
               if(fieldValue instanceof String)
                   value.append(" '").append(fieldValue).append("' ,");
               else if(fieldValue instanceof Integer)
                   value.append(" ").append(fieldValue).append(" ,");

            }
            name.deleteCharAt(name.length() - 1).append(")");
            value.deleteCharAt(value.length() - 1).append(",");
        }
        return sql.toString()+name.toString()+" VALUES
"+value.deleteCharAt(value.length() - 1).toString();

    }

    @Override
    public String delete(User user) {
        // TODO Auto-generated method stub
         StringBuffer sql = new StringBuffer();

            Class c = user.getClass();
            boolean flag = c.isAnnotationPresent(Table.class);
            if(!flag){
                return null;
            }
            Table table = (Table) c.getAnnotation(Table.class);
            String tableName = table.value();
            sql.append("DELETE FROM '").append(tableName).append("' WHERE");
            Field[] fields = c.getDeclaredFields();
            for (int i = 0; i < fields.length; i++) {
                boolean fieldFalg = fields[i].isAnnotationPresent(Column.class);
                if(!fieldFalg){
                    continue;
                }
                Column column = fields[i].getAnnotation(Column.class);
                String fieldName = column.value();
                Object fieldValue = null ;
                try {
                    String methodName =
"get"+fieldName.substring(0,1).toUpperCase()+fieldName.substring(1).toString();
                    Method method = c.getMethod(methodName);
                    fieldValue = method.invoke(user,null);
                } catch (NoSuchMethodException e) {
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    e.printStackTrace();
                } catch (InvocationTargetException e) {
                    e.printStackTrace();
                }
```

```
                if(fieldValue == null || (fieldValue instanceof Integer &&
(Integer)fieldValue == 0)){
                    continue;
                }
                if(fieldValue instanceof String){
                    sql.append(" '").append(fields[i].getName()).append("' like
'").append(fieldValue).append("'");
                }else if(fieldValue instanceof Integer){
                    sql.append(" '").append(fields[i].getName()).append("' =
").append(fieldValue);
                }
            }

            return sql.toString();

    }

    @Override
    public String update(User user) {
        // TODO Auto-generated method stub
         StringBuffer sql = new StringBuffer();
         StringBuffer sql2 = new StringBuffer();
            Class c = user.getClass();
            boolean flag = c.isAnnotationPresent(Table.class);
            if(!flag){
                return null;
            }
            Table table = (Table) c.getAnnotation(Table.class);
            String tableName = table.value();
            sql.append("UPDATE '").append(tableName).append("' SET ");
            Field[] fields = c.getDeclaredFields();
            for (int i = 0; i < fields.length; i++) {
                boolean fieldFalg = fields[i].isAnnotationPresent(Column.class);
                if(!fieldFalg){
                    continue;
                }
                Column column = fields[i].getAnnotation(Column.class);
                String fieldName = column.value();
                Object fieldValue = null ;
                try {
                    String methodName =
"get"+fieldName.substring(0,1).toUpperCase()+fieldName.substring(1).toString();
                    Method method = c.getMethod(methodName);
                    fieldValue = method.invoke(user,null);
                } catch (NoSuchMethodException e) {
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    e.printStackTrace();
                } catch (InvocationTargetException e) {
                    e.printStackTrace();
                }

                if(fieldValue == null || (fieldValue instanceof Integer &&
(Integer)fieldValue == 0)){
                    continue;
                }
                if(fieldValue instanceof String){
```

```
 sql.append(fields[i].getName()).append("='").append(fieldValue).append("'").app
end("and");
                }else if(fieldValue instanceof Integer){
                    sql2.append(" WHERE
'").append(fields[i].getName()).append("' =").append(fieldValue);
                }
            }

            return sql.toString().substring(0, sql.length()-3)+sql2.toString();

    }

}


import java.lang.annotation.*;

/*Table注解*/
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)

public @interface Table {

    String value();

}

/*Column注解*/
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Column {

    String value();
}
```

1、每次先判断是否包含Table的注解 boolean flag = c.isAnnotationPresent(Table.class);

2、 再获取到注解对象和注解对象的值
 Table table = (Table) c.getAnnotation(Table.class);
 String tableName = table.value();

3、 获取类中所有的属性，接着判断是否包含Column的注解，获取方法名

Field[] fields = c.getDeclaredFields();

Column column = fields[i].getAnnotation(Column.class);
String fieldName = column.value();

String methodName =
"get"+fieldName.substring(0,1).toUpperCase()+fieldName.substring(1).toString();

4 、通过反射提取到值

Method method
method = c.getMethod(methodName);
fieldValue = method.invoke(user,null);

5、 拼接SQL语句

三、运行结果

```
Console
<terminated> Main (2) [Java Application] C:\Program Files\Java\jre1.8.0_102\bin\javaw.exe (2020年4月7日 下午3:09:17)
select * from user where  id = 175
select * from user where 'username' like '史荣贡'
INSERT INTO 'user' ( 'username' , 'age' , 'email' , 'telephone' ) VALUES ( 'user' , 20 , 'user@123.com' , '12345678123' )
INSERT INTO 'user' ( 'username' , 'age' , 'email' , 'telephone' ) VALUES ( 'user' , 20 , 'user@123.com' , '12345678123' ),(
'user2' , 20 , 'user2@123.com' , '12345678121' )
UPDATE 'user' SET email='change@123.com' WHERE 'id' =1
DELETE FROM 'user' WHERE 'id' = 1
```