# JAVA编程进阶上机报告

****

**学 院 智能与计算学部**

**专 业 软件工程**

**班 级 5班**

**学 号 3018216259**

**姓 名 李澄**

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**：

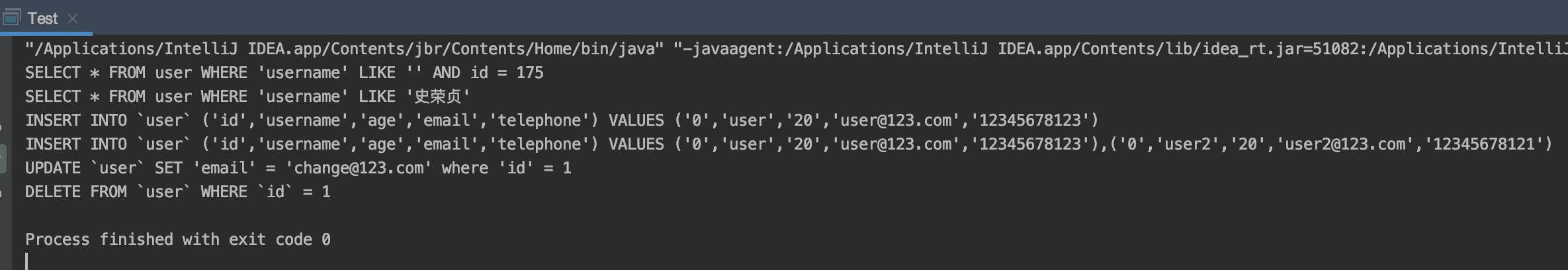
@Column：用来标注每个field对应的表中的字段是什么  
@Table：用来标记表的名字

1. **源代码**
2. @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;  
    }  
     
    public int getId() {  
    return id;  
    }  
     
    public String getUsername() {  
    return username;  
    }  
     
    public int getAge() {  
    return age;  
    }  
     
    public String getEmail() {  
    return email;  
    }  
     
    public String getTelephone() {  
    return telephone;  
    }  
   }

public interface SqlUtil {  
 */\*\*  
 \* 根据传入的参数返回查询语句  
 \** ***@param*** *user  
 \** ***@return*** *返回查询语句  
 \*/* String query(User user);  
  
 */\*\*  
 \* 根据传入的参数返回插入语句  
 \** ***@param*** *user  
 \** ***@return*** *返回插入语句  
 \*/* String insert(User user);  
  
 */\*\*  
 \* 根据传入的参数返回插入语句  
 \** ***@param*** *users  
 \** ***@return*** *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）  
 \** ***@param*** *user  
 \** ***@return*** *返回删除语句  
 \*/* String delete(User user);  
 */\*\*  
 \* 根据传入的参数返回更新语句（将id为user.id的记录的其它字段更新成user中的对应值）  
 \** ***@param*** *user  
 \** ***@return*** *返回更新语句  
 \*/* String update(User user);  
  
}

public class DataBase implements SqlUtil {  
 String sentence;  
  
 @Override  
 public String query(User user) {  
 String qid;  
 String qname;  
 String querySentence = "SELECT \* FROM user WHERE ";  
 List<String> condition = new LinkedList<String>();  
 Class c = User.class;  
 try {  
 Field fieldId = c.getDeclaredField("id");  
 Field fieldUsername = c.getDeclaredField("username");  
 fieldUsername.setAccessible(true);  
 String name = (String) fieldUsername.get(user);  
 fieldId.setAccessible(true);  
 int id = (int) fieldId.get(user);  
 if (name != null) {  
 qname = name;  
 condition.add("'username' LIKE '" + name + "'");  
 }  
 if (id != 0) {  
 qid = String.*valueOf*(id);  
 condition.add("id = " + qid);  
 }  
 } catch (NoSuchFieldException | IllegalAccessException e) {  
 e.printStackTrace();  
 }  
 Iterator<String> iterator = condition.iterator();  
 querySentence += iterator.next();  
 if (condition.size() > 1) {  
 querySentence += " AND ";  
 querySentence += iterator.next();  
 }  
  
 return querySentence;  
 }  
  
 @Override  
 public String insert(User user) {  
 sentence = "INSERT INTO `user` ('";  
 Class c = User.class;  
 List<String> values = new LinkedList<String>();  
 Field[] fields = c.getDeclaredFields();  
  
 for (Field f: fields) {  
 StringBuilder builder = new StringBuilder();  
 builder.append(f.getName());  
// list.add(builder);  
 sentence += builder;  
 sentence += "','";  
 f.setAccessible(true);  
 try {  
 String value = "";  
 if (f.get(user).getClass().toString().equals("class java.lang.Integer")) {  
 value = String.*valueOf*(f.get(user));  
 } else {  
 value = (String) f.get(user);  
 }  
 values.add(value);  
 } catch (IllegalAccessException e) {  
  
 e.printStackTrace();  
 }  
 }  
 sentence = sentence.substring(0,sentence.length()-2);  
 sentence += ") VALUES ('";  
 for (String item: values) {  
 sentence += item;  
 sentence += "','";  
 }  
 sentence = sentence.substring(0,sentence.length() - 2);  
 sentence += ")";  
  
 return sentence;  
 }  
  
 @Override  
 public String insert(List<User> users) {  
 sentence = "";  
 List<String> list = new LinkedList<String>();  
 for (User user: users) {  
 String str = insert(user);  
 String[] strArray = str.split("\\(");  
 sentence = strArray[0] + "(" + strArray[1];  
 String value = "(" + strArray[2];  
 list.add(value);  
 }  
 for (String item: list) {  
 sentence += item;  
 sentence += ",";  
 }  
 sentence = sentence.substring(0,sentence.length() - 1);  
 return sentence;  
 }  
  
 @Override  
 public String delete(User user) {  
 sentence = "DELETE FROM `user` WHERE `id` = ";  
 try {  
 Class c = user.getClass();  
 Field fieldId = c.getDeclaredField("id");  
 fieldId.setAccessible(true);  
 int id = (int) fieldId.get(user);  
 sentence += id;  
 } catch ( Exception e ) {  
 e.printStackTrace();  
 }  
  
 return sentence;  
 }  
  
 @Override  
 public String update(User user) {  
 sentence = "UPDATE `user` SET ";  
 Map<String,Object> map = fieldToMap(user);  
 Set<Map.Entry<String,Object>> set = map.entrySet();  
 int id = 0;  
 for (Map.Entry<String, Object> entry: set) {  
 if (entry.getKey().intern() == "id") {  
 id = (int) entry.getValue();  
 continue;  
 }  
 String str = "'" + entry.getKey() + "'" + " = " + "'" + entry.getValue() + "' ";  
 sentence += str;  
 sentence += "where 'id' = " + id;  
 }  
  
  
 return sentence;  
 }  
  
 private Map<String,Object> fieldToMap(User user) {  
 Map<String, Object> map = new HashMap<String, Object>();  
 Class c = User.class;  
 Field[] fields = c.getDeclaredFields();  
 for (Field f : fields) {  
 f.setAccessible(true);  
 try {  
 String value = "";  
// System.out.println(f.get(user).getClass().toString());  
 if (f.get(user).getClass().toString().intern() == "class java.lang.Integer") {  
 value = String.*valueOf*(f.get(user));  
 } else {  
 value = (String) f.get(user);  
 }  
 if (value.intern() != "" && value.intern() != "0") {  
 map.put(f.getName(), f.get(user));  
 }  
 } catch (IllegalAccessException e) {  
  
 e.printStackTrace();  
 }  
 }  
 return map;  
 }  
}

**三、实验结果**

****