**DAO（Data Access Object）**

* why：实现功能的模块化。更有利于代码的维护和升级。DAO可以被子类继承或者直接使用。
* what：访问数据信息的类。包含了对数据的CRUD（create，read，update，delete）。而不包含任何业务相关的信息。
* how：使用JDBC编写Dao可能会包含的方法：

//INSERT,UPDATE,DELETE操作都可以包含在其中

void update(String sql, Object ... args);

//查询一条记录，返回对应的对象

<T> T get(Class<T> clazz, String sql, Object ... args);

//查询多条记录，返回对应的对象的集合

<T> List<T> getForList(Class<T> clazz, String sql, Object ... args);

//返回某条记录的某一个字段的值，或一个统计的值（一个有多少条记录等。）

<E> E getForValue(String sql, Object ... args);

**public class** Dao {  
  
 */\*\*  
 \* description: INSERT,UPDATE,DELETE操作都可以包含在其中  
 \*  
 \** ***@param:*** *[sql, args]  
 \** ***@return:*** *void  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/3/28 9:14  
 \*/* **public void** update(String sql, Object... args) {  
 Connection conn = **null**;  
 PreparedStatement ps = **null**;  
  
 **try** {  
 conn = JDBCUtil.*getConnection*();  
 ps = conn.prepareStatement(sql);  
  
 **for** (**int** i = 0; i < args.**length**; i++) {  
 ps.setObject(i + 1, args[i]);  
 }  
  
 ps.executeUpdate();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 } **finally** {  
 JDBCUtil.*releaseDB*(**null**, ps, conn);  
 }  
 }  
  
 *//查询一条记录，返回对应的对象* **public** <T> T get(Class<T> clazz, String sql, Object... args) {  
 T entity = **null**;  
  
 Connection conn = **null**;  
 PreparedStatement ps = **null**;  
 ResultSet rs = **null**;  
  
 **try** {  
 *//1.获取Connection* conn = JDBCUtil.*getConnection*();  
 *//2.获取PreparedStatement* ps = conn.prepareStatement(sql);  
 *//3.填充占位符* **for** (**int** i = 0; i < args.**length**; i++) {  
 ps.setObject(i + 1, args[i]);  
 }  
 *//4.进行查询，得到ResultSet* rs = ps.executeQuery();  
 *//5.若ResultSet中有记录，  
 // 准备一个Map<String, Object>：键：存放列的别名，值：存放列的值* **if** (rs.next()) {  
 Map<String, Object> values = **new** HashMap<>();  
 *//6.得到ResultSetMetaData对象* ResultSetMetaData rsmd = rs.getMetaData();  
 *//7.处理ResultSet，把指针向下移动一个单位  
  
 //8.由得到ResultSetMetaData对象得到结果集中有多少列* **int** columnCount = rsmd.getColumnCount();  
 *//9.由得到ResultSetMetaData对象得到每一列的别名，由ResultSet得到每一列的值* **for** (**int** i = 0; i < columnCount; i++) {  
 String columnLabel = rsmd.getColumnLabel(i + 1);  
 Object columnValue = rs.getObject(i + 1);  
  
 *//10.填充Map对象* values.put(columnLabel, columnValue);  
 }  
 *//11.用反射创建Class对应的对象* entity = clazz.newInstance();  
 *//12.遍历Map对象，用反射填充对象的属性值：属性名为Map中的key，属性值为Map中的value* **for** (Map.Entry<String, Object> entry : values.entrySet()) {  
 String propertyName = entry.getKey();  
 Object value = entry.getValue();  
  
 ReflectionUtils.*setFieldValue*(entity, propertyName, value);  
 }  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 } **finally** {  
 JDBCUtil.*releaseDB*(rs, ps, conn);  
 }  
 **return** entity;  
 }  
  
 *//查询多条记录，返回对应的对象的集合* **public** <T> List<T> getForList(Class<T> clazz, String sql, Object... args) {  
  
 **return null**;  
 }  
  
 *//返回某条记录的某一个字段的值，或一个统计的值（一个有多少条记录等。）* **public** <E> E getForValue(String sql, Object... args) {  
  
 **return null**;  
 }  
  
}