*/\*\*  
 \* description: 查询多条记录，返回对应的对象的集合  
 \*   
 \** ***@param:*** *[clazz, sql, args]  
 \** ***@return:*** *java.util.List*<*T*>  
 *\** ***@author:*** *nino  
 \** ***@date:*** *2019/3/28 16:16  
 \*/***public** <T> List<T> getForList(Class<T> clazz, String sql,  
 Object... args) {  
 List<T> list = **new** ArrayList<>();  
  
 Connection conn = **null**;  
 PreparedStatement ps = **null**;  
 ResultSet rs = **null**;  
  
 **try** {  
 *//得到结果集* conn = JDBCUtil.*getConnection*();  
 ps = conn.prepareStatement(sql);  
 **for** (**int** i = 0; i < args.**length**; i++) {  
 ps.setObject(i + 1, args[i]);  
 }  
 rs = ps.executeQuery();  
  
 *//5.准备一个List<Map<String, Object>>：  
 // 键：存放列的别名，值：存放列的值。其中一个Map对象对应着一条记录* List<Map<String, Object>> values = **new** ArrayList<>();  
  
 ResultSetMetaData rsmd = rs.getMetaData();  
 Map<String, Object> map = **null**;  
  
 *//7.处理ResultSet，使用while循环* **while** (rs.next()) {  
 map = **new** HashMap<>();  
  
 **for** (**int** i = 0; i < rsmd.getColumnCount(); i++) {  
 String columnLabel = rsmd.getColumnLabel(i + 1);  
 Object value = rs.getObject(i + 1);  
  
 map.put(columnLabel, value);  
 }  
 *//11.把一条记录的一个Map对象放入5准备的List中* values.add(map);  
 }  
  
 *//12.判断List是否为空集合，若不为空，  
 // 则遍历List，得到一个一个的Map对象，再把一个Map对象转为一个Class  
 // 参数对应的Object对象* T bean = **null**;  
  
 **if** (values.size() > 0) {  
 **for** (Map<String, Object> m : values) {  
 bean = clazz.newInstance();  
 **for** (Map.Entry<String, Object> entry : m.entrySet()) {  
 String propertyName = entry.getKey();  
 Object value = entry.getValue();  
  
 BeanUtils.*setProperty*(bean, propertyName, value);  
 }  
 *//13.把Object对象放入到list中。* list.add(bean);  
 }  
 }  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 } **finally** {  
 JDBCUtil.*releaseDB*(rs, ps, conn);  
 }  
  
 **return** list;  
}

进一步优化上述代码：

*/\*\*  
 \* description: 查询多条记录，返回对应的对象的集合  
 \*  
 \** ***@param:*** *[clazz, sql, args]  
 \** ***@return:*** *java.util.List*<*T*>  
 *\** ***@author:*** *nino  
 \** ***@date:*** *2019/3/28 16:16  
 \*/***public** <T> List<T> getForList(Class<T> clazz, String sql,  
 Object... args) {  
 List<T> list = **new** ArrayList<>();  
  
 Connection conn = **null**;  
 PreparedStatement ps = **null**;  
 ResultSet rs = **null**;  
  
 **try** {  
 *//1.得到结果集* conn = JDBCUtil.*getConnection*();  
 ps = conn.prepareStatement(sql);  
 **for** (**int** i = 0; i < args.**length**; i++) {  
 ps.setObject(i + 1, args[i]);  
 }  
 rs = ps.executeQuery();  
  
 *//2.处理结果集，得到Map的List，  
 // 其中一个Map对象就是一条记录，Map的key就是rs中列的别名，value为列的值* List<Map<String, Object>> values = handleResultSetToMapList(rs);  
  
 *//3.把Map的List转为Clazz对应的List，  
 // 其中Map的key即为clazz对应的对象的propertyName，  
 // 而Map的value即为clazz对应的对象的propertuValue* list = transferMapListToBeanList(clazz, values);  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 } **finally** {  
 JDBCUtil.*releaseDB*(rs, ps, conn);  
 }  
  
 **return** list;  
}  
  
*/\*\*  
 \* description: 把Map的List转为Clazz对应的List  
 \*  
 \** ***@param:*** *[clazz, list, values]  
 \** ***@return:*** *void  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/3/28 16:45  
 \*/***private** <T> List<T> transferMapListToBeanList(Class<T> clazz,  
 List<Map<String, Object>> values) **throws** InstantiationException,  
 IllegalAccessException,  
 InvocationTargetException {  
 List<T> result = **new** ArrayList<>();  
  
 T bean = **null**;  
  
 **if** (values.size() > 0) {  
 **for** (Map<String, Object> m : values) {  
 bean = clazz.newInstance();  
 **for** (Map.Entry<String, Object> entry : m.entrySet()) {  
 String propertyName = entry.getKey();  
 Object value = entry.getValue();  
  
 BeanUtils.*setProperty*(bean, propertyName, value);  
 }  
 *//13.把Object对象放入到list中。* result.add(bean);  
 }  
 }  
  
 **return** result;  
}  
  
*/\*\*  
 \* description: 处理结果集，得到Map的一个List，其中一个Map对象对应一条记录  
 \*  
 \** ***@param:*** *[rs]  
 \** ***@return:*** *java.util.List*<*java.util.Map* < *java.lang.String* , *java.lang.Object*>*>  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/3/28 16:41  
 \*/***private** List<Map<String, Object>> handleResultSetToMapList(ResultSet rs) **throws** SQLException {  
 *//5.准备一个List<Map<String, Object>>：  
 // 键：存放列的别名，值：存放列的值。其中一个Map对象对应着一条记录* List<Map<String, Object>> values = **new** ArrayList<>();  
  
 List<String> columnLabels = getColumnLabels(rs);  
 Map<String, Object> map = **null**;  
  
 *//7.处理ResultSet，使用while循环* **while** (rs.next()) {  
 map = **new** HashMap<>();  
  
 **for** (String columnLabel : columnLabels) {  
 Object value = rs.getObject(columnLabel);  
  
 map.put(columnLabel, value);  
 }  
 *//11.把一条记录的一个Map对象放入5准备的List中* values.add(map);  
 }  
 **return** values;  
}  
  
*/\*\*  
 \* description: 获取结果集的ColumnLabel对应的List  
 \*  
 \** ***@param:*** *[rs]  
 \** ***@return:*** *java.util.List*<*java.lang.String*>  
 *\** ***@author:*** *nino  
 \** ***@date:*** *2019/3/28 16:26  
 \*/***private** List<String> getColumnLabels(ResultSet rs) **throws** SQLException {  
 List<String> labels = **new** ArrayList<>();  
  
 ResultSetMetaData rsmd = rs.getMetaData();  
 **for** (**int** i = 0; i < rsmd.getColumnCount(); i++) {  
 labels.add(rsmd.getColumnLabel(i + 1));  
 }  
  
 **return** labels;  
}

*/\*\*  
 \* description: 查询一条记录，返回对应的对象  
 \*  
 \** ***@param:*** *[clazz, sql, args]  
 \** ***@return:*** *T  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/3/28 14:30  
 \*/***public** <T> T get(Class<T> clazz, String sql, Object... args) {  
 List<T> result = getForList(clazz,sql,args);  
 **if** (result.size()>0) {  
 **return** result.get(0);  
 }  
 **return null**;  
}

*/\*\*  
 \* description: 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。）  
 \*  
 \** ***@param:*** *[sql, args]  
 \** ***@return:*** *E  
 \** ***@author:*** *nino  
 \** ***@date:*** *2019/3/29 16:23  
 \*/***public** <E> E getForValue(String sql, Object... args) {  
 *//1.获得结果集：该结果集应该只有一行且只有一列* Connection conn = **null**;  
 PreparedStatement ps = **null**;  
 ResultSet rs = **null**;  
  
 **try** {  
 *//1.得到结果集* conn = JDBCUtil.*getConnection*();  
 ps = conn.prepareStatement(sql);  
 **for** (**int** i = 0; i < args.**length**; i++) {  
 ps.setObject(i + 1, args[i]);  
 }  
 rs = ps.executeQuery();  
  
 **if** (rs.next()) {  
 **return** (E) rs.getObject(1);  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 } **finally** {  
 JDBCUtil.*releaseDB*(rs, ps, conn);  
 }  
 *//2.取得结果* **return null**;  
}