项目：手写ORM小框架

# 功能

表结构--对象：

Query：使用sql语句查询得到对象

Insert：将PO对象插入表中

Delete：利用PO对象（对应记录）删除表中记录

Update：利用PO对象（对应记录）更改表中记录

## 增删改查

增删改： 根据对象生成sql语句操作数据库（对象🡪数据库）

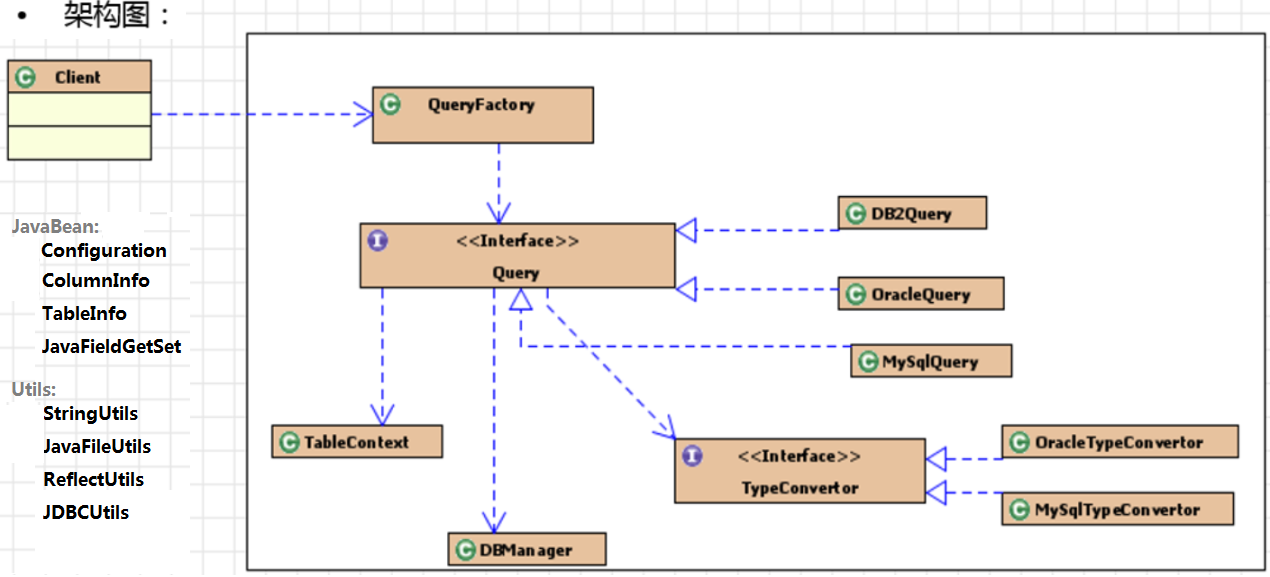
|  |
| --- |
| 增加 --根据对象， 生成sql语句，插入数据库  删除 --根据对象主键id， 生成sql语句，删除数据库表记录  修改 --根据对象， 生成sql语句，修改记录 |

查询： 查询数据库将获得的信息封装成对象（对象🡪数据库🡪对象）

|  |
| --- |
| 按结果集分类：  多行多列 🡪 list<JavaBean>  一行多列 🡪 JavaBean  一行一列  普通对象Object  数字 Number |

# 总体思路

## UML架构图



## 核心类

DBManager类：

根据配置信息，创建连接池，负责连接对象管理

TableContext接口：

负责维护SQL表结构和Java类结构的映射

Query接口：

核心服务类，负责增删改查操作

Utils:

JDBCUtils封装 连接关闭/关闭流

StringUtils封装 字符串操作

JavaFileUtils生成.Java文件源代码，生成.java文件

ReflectUtils封装 常用反射操作

## JavaBean

Configuration 封装配置文件 🡪 db.properties

ColumnInfo 封装字段信息 (字段名、字段类型、键类型)

TableInfo 封装表信息（存放各个字段）(只有一个主键)

表信息 private Map<String,ColumnInfo> columns;

库信息（多张表） public Map<String,TableInfo> tables = new HashMap<String,TableInfo>();

JavaFieldGetSet 将ColumnInfo字段信息封装为Java属性设置源码，getset方法源码

将多个JavaFeildGetSet类属性拼接起来，添加头尾，就可以生成一个表类的.java文件

# 设计流程：

## 第一步：将表的结构信息同步更新生成.java文件

2配置文件：db.properties

driver=com.mysql.jdbc.Driver

url=jdbc\:mysql\://localhost\:3306/sorm

user=root

pwd=888888

usingDB=mysql

srcPath=E\:\\javaworkspace\\workspace02\\SORM\\src

poPackage=com.bjsxt.po//存放由表结构生成的.java文件

4 Configuration配置信息类

负责存储配置信息，供DBManager调用

5 DBManager

包含Configuration对象conf，

Static块读取配置文件Properties，获取配置信息，给 Configuration对象conf属性赋值，

提供返回配置信息类conf方法

提供返回连接方法

提供关闭连接方法

提供关闭资源方法

6 ColumnInfo 封装字段信息 (字段名、字段类型、键类型)

TableInfo 封装表信息（存放各个字段）(只有一个主键)

表信息 private Map<String,ColumnInfo> columns;

主键 private List<ColumnInfo> priKeys;

7 TableContext

包含存储<string名字，TableInfo表结构>的Map对象Map<String, TableInfo> tables

包含存储<Class表类，TableInfo表结构>的Map对象Map<Class, TableInfo> poClassTableMap

static静态块

将数据库表信息封装到tables

通过DBManager#getconn( )获取连接，

将结果集，封装到tables

将tables表结构信息，通过字符串拼接，生成对应的JavaBean的.java的字符文件，在项目路径下创建PO包文件夹，导入.java文件。（这样就差不多实现了自动映射）

**待8/9/10完成后继续**

调用**updateJavaPOFile()**生成表类.java文件

疑问：项目

Bug：不手动刷新PO文件夹和.java文件，loadPOTables()方法运行时加载Class对象，不会生成.class文件

手动刷新PO文件夹和.java文件（项目右键refresh），得到对应的.class文件，Map<Class,TableInfo> poClassTableMap加载Class

调用**loadPOTables()（调用createJavaPOFile()）**加载表类Class，填充poClassTableMap, 实现了从表结构到类结构转化

加载java类文件Class.forName(“com.bjsxt.po.Emp”);生成Class对象，

利用Map<Class,TableInfo> poClassTableMap将po的class对象和表信息对象关联起来更新PO包

8 MySqlTypeConvertor

**databaseType2JavaType()**实现数据库类型到Java类型转换，为生成.java源码信息做准备

9 JavaFieldGetSet 记录字段信息类

将ColumnInfo字段信息封装为Java属性设置源码，getset方法源码

将多个JavaFeildGetSet类属性拼接起来，就可以生成一个表类的.java文件

10.实现JavaFileUtils利用表信息生成表对应类的源码、po包下的.java文件

**createFieldGetSetSRC()**

根据字段信息ColumnInfo生成属性信息类JavaFieldGetSet

**createJavaSRC()**

将表信息中的字段信息处理生成属性信息类，

组合属性信息生成.java文件源代码

**createJavaPOFile()**

利用字符文件输出流FileWriter将表类源码输出到PO包路径下,生成表类.java文件

**跳转至7继续执行**

## 第二步：删/增/改（将表类对象同步到数据库）🡪 查（利用JDBC执行SQL查询语句，将查询结果封装到表类对象）

insert into user (id, uname, pwd) values (?,?,? );

delete from user where id=2;

update user set uname=?,pwd=?,where id=?;;

select \* from emp where id=?;

public String databaseType2JavaType(String columnTye){

if(“varchar”.equalIgnoreCase(columnType) || )

}

varchar char String

int tinyint Integer

Long

Double

11. <删>delete方法：删除Object对象在数据库中对应的记录

通过 类名Class 找到 表名tableinfo（绑定）

public static Map<Class, TableInfo> *poClassTableMap*

= new HashMap<>();

利用反射获取Object对象具体信息，传递给数据库执行语句类（将反射方法封装到ReflectUtils）

12.封装executeDML方法（封装传参方法），供<增删改>操作调用，测试delete方法

13. <增>insert方法：将一个对象存储到数据库

14. <改>update方法：更新对象对应的记录，并且只更新指定的字段的值

15. <查>query方法：将SQL查询语句传递给JDBC执行，获得查询信息记录，创建存储对象，用查询获得的信息记录给存储对象赋值，实现信息封装。（将查到的信息放到emp，dept对象中，将对象返回，即是查询结果）

多行多列 List<Javabean> queryRows()

一行一列 Javabean queryUniqueRow()

联合查询EmpVO（需要新建存放存储查询结果的EmpVO对象）

查询值 queryValue()

查询数字 queryNumber()

## 第三大步：优化

17.（模板模式）executeQueryTemplate

将Query流程提取为executeQueryTemplate(CallBack back)模板方法，供queryRows()，queryValue()调用

传递模板钩子对象给模板函数执行模板操作；模板函数执行回调函数，回调函数执行业务操作。

18. （工厂模式）QueryFactory将创建Query对象分离，修改方便。提供统一接口，提高开发效率。

配置Query对象路径属性，利用QueryFactory创建具体Query对象：

QueryFactory静态块：加载DBManager/TableContext，更新表结构对应的表类，保证增删改查操作顺利进行

（单例模式）将工厂构造函数私有化，保证只有一个工厂，产生的Query类为单例。

19.DBConnPool连接池对象（提高连接效率）（已经有成熟的连接池DBCP,c3p0）

取/关连接都需要同步

public synchronized Connection getConnection() {}

public synchronized void close(Connection conn){}

DBManager中包含连接池，控制连接池的初始化/取/还

getConn()直接去连接池中取connection对象，不再创建连接对象

Close()将connection对象放回DBConnPool中，不再关闭连接

20.生成SORM.jar包/API文档/SRC/readme/数据库创建语句

## 第四步：测试Jar包

21.测试SORM.jar包，

bug:

空指针：表字段为空

解法：提前判断对象是否为null

## 说明文件：readme

1. 在src下建立db.properties

2. po尽量使用包装类，方便使用obj==null判断对象，不要使用基本数据类型。

3．多线程不安全