SORM框架

- 我们希望设计一个可以实现对象和SQL自动映射的框架，但是整体用法和设计比Hibernate简单。砍掉不必要的功能。
- 会穿插使用设计模式

- 增加
  - 将对象对应成sql语句，执行sql，插入数据库中
- 删除
  - 根据对象主键的值，生成sql，执行，从库中删除
- 修改
  - 根据对象需要修改属性的值，生成sql，执行

SORM框架

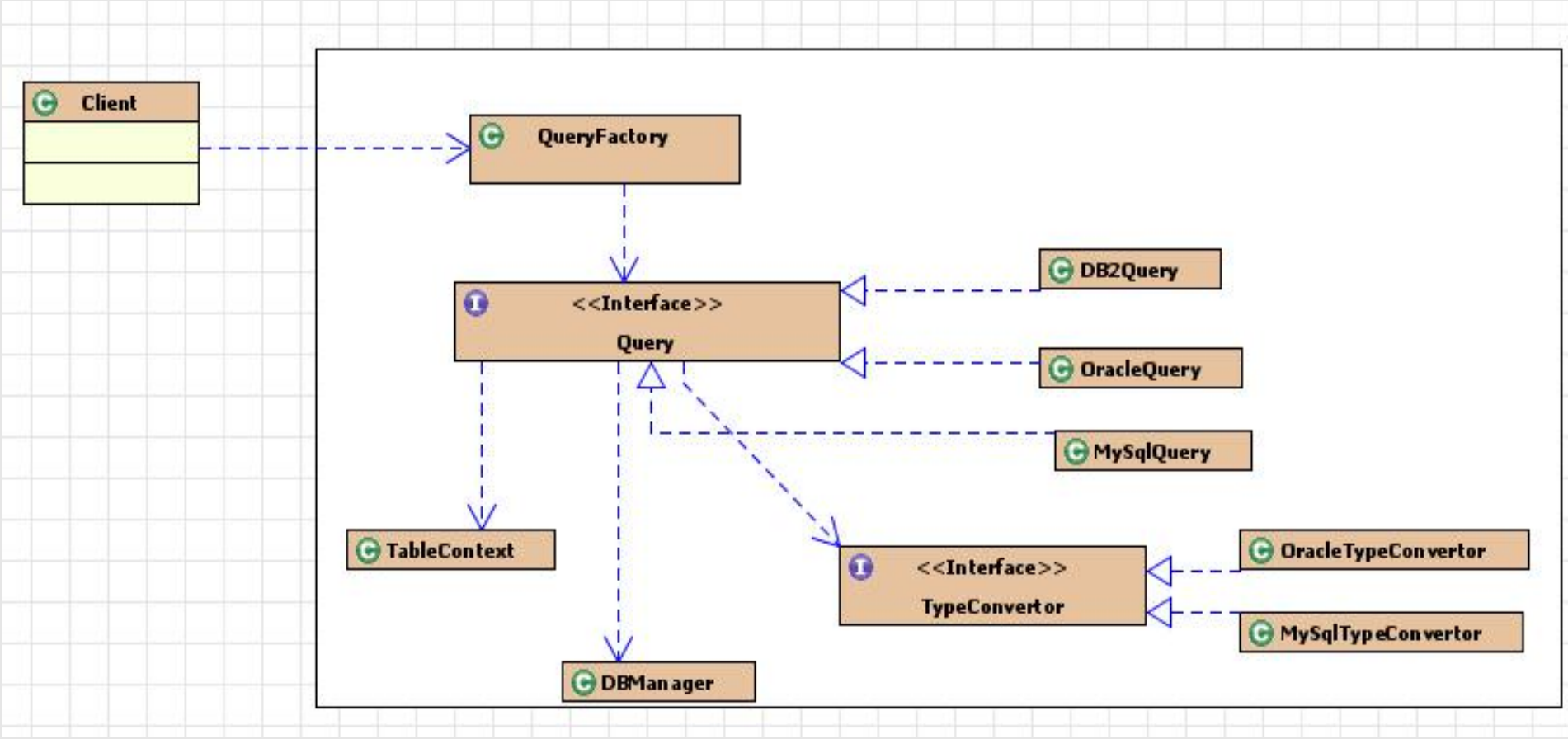
- 查询
  - 根据结果分类：
    - 多行多列：List<Javabean>
    - 一行多列：Javabean
    - 一行一列：
      - 普通对象：Object
      - 数字：Number

SORM框架

- 核心架构：
  - **Query接口：负责查询（对外提供服务的核心类）**
  - **QueryFactory类：负责根据配置信息创建query对象**
  - **TypeConvertor接口：负责类型转换**
  - **TableContext类：**
    - 负责获取管理数据库所有表结构和类结构的关系，并可以根据表结构生成类结构。
  - **DBManager类：**
    - 根据配置信息，维持连接对象的管理(增加连接池功能)
  - 工具类：
    - JDBCUtils封装常用JDBC操作
    - JavaFileUtils封装java文件操作
    - StringUtils封装常用字符串操作
    - ReflectUtils封装常用反射操作

SORM框架

- 架构图：



SORM框架

- 核心bean，封装相关数据：
  - ColumnInfo　　封装表中一个字段的信息(字段类型、字段名、键类型)
  - Configuration　　封装配置文件信息
  - TableInfo　　封装一张表的信息

- 针对SORM框架的说明：
  - **核心思想：使用简单、性能高、极易上手！**
  - 配置文件
    - 目前使用资源文件、后期项目复杂后可以增加XML文件配置和注解。
  - 类名由表名生成，只有首字母大写有区别，其他无区别
  - Java对象的属性由表中字段生成，完全对应
  - 目前，只支持表中只有一个主键，联合主键不支持

模板方法模式优化Query

- 回调接口

```
public interface CallBack {
        public Object doExecute(Connection conn,PreparedStatement ps,ResultSet rs);
}
```

- 模板方法

```
public Object executeQueryTemplate(String sql,Object[] params,Class clazz,CallBack back){
        Connection conn = DBManager.getConn();
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
                ps = conn.prepareStatement(sql);
                //给sql设参
                JDBCUtils.handleParams(ps, params);
                System.out.println(ps);
                rs = ps.executeQuery();

                return  back.doExecute(conn, ps, rs);

        } catch (Exception e) {
                e.printStackTrace();
                return null;
        }finally{
                DBManager.close(ps, conn);
        }
}
```

模板方法模式优化Query

- 调用示例

```
public Object queryValue(String sql,Object[] params){
        return executeQueryTemplate(sql, params, null, new CallBack() {
                @Override
                public Object doExecute(Connection conn, PreparedStatement ps, ResultSet rs) {
                        Object value = null;
                        try {
                                while(rs.next()){
                                        value = rs.getObject(1);
                                }
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                        return value;
                }
        });
}
```

QueryFactory

- 使用工厂模式统计管理Query的创建

- 使用克隆模式提高Query对象的创建效率

连接池

- **连接池(Connection Pool)**
  - 就是将Connection对象放入List中，反复重用
  - 连接池的初始化：
    - 事先放入多个连接对象。
  - 从连接池中取连接对象
    - 如果池中有可用连接，则将池中最后一个返回。同时，将该连接从池中remove，表示正在使用。
    - 如果池中无可用连接，则创建一个新的。
  - 关闭连接
    - 不是真正关闭连接，而是将用完的连接放入池中
- 市面上的连接池产品：
  - DBCP
  - c3p0
  - proxool