



Mybatis高级（二）

T A H N K Y O U F O R W A T C H I N G



主讲老师Lison : 525765982



课程咨询依娜老师 : 2470523467

目录

CONTENTS



MyBatis核心流程分析

MyBatis核心流程分析



配置加载阶段

建造者模式
mybatis初始化过程



Mybatis的接口层

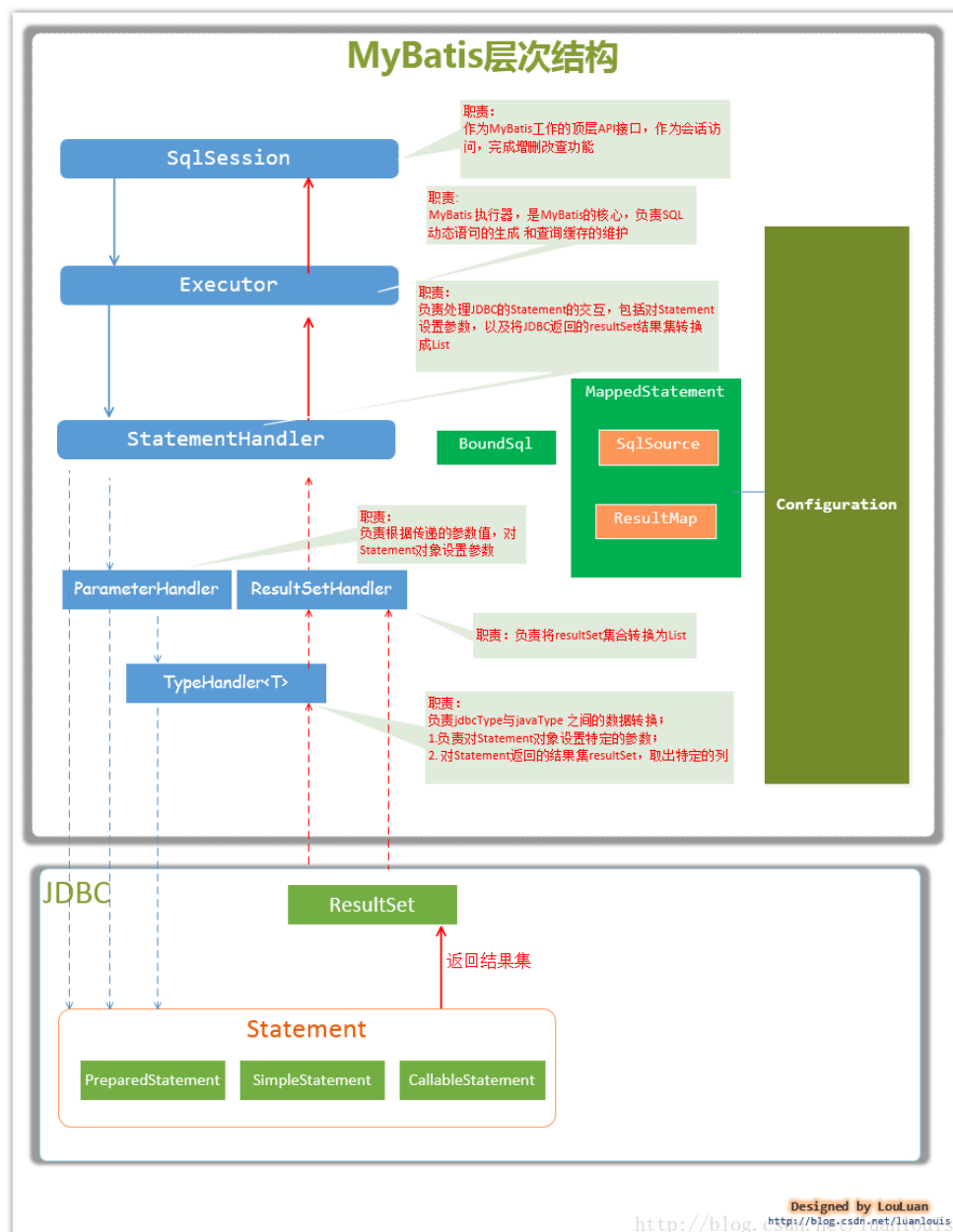
策略模式
sqlSession相关的类



核心组件Executor

模板模式
Executor组件分析
Executor的三个小弟


mybatis功能流程图





目录

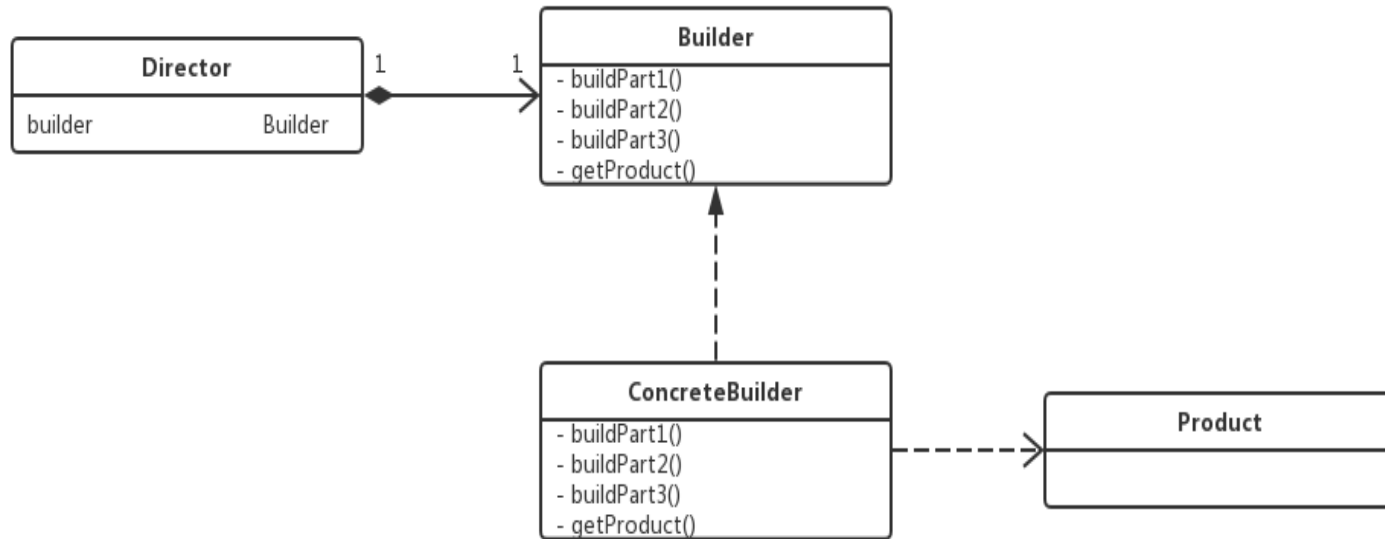
CONTENTS

			
MyBatis核心流程分析	配置加载阶段	Mybatis的接口层	核心组件Excutor
MyBatis核心流程分析	建造者模式 mybatis初始化过程	策略模式 sqlSession相关的类	模板模式 Excutor组件分析 Excutor的三个小弟



Mybatis的初始化 建造者模式

- 建造者模式 (Builder Pattern) 使用多个简单的对象一步一步构建成一个复杂的对象。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。



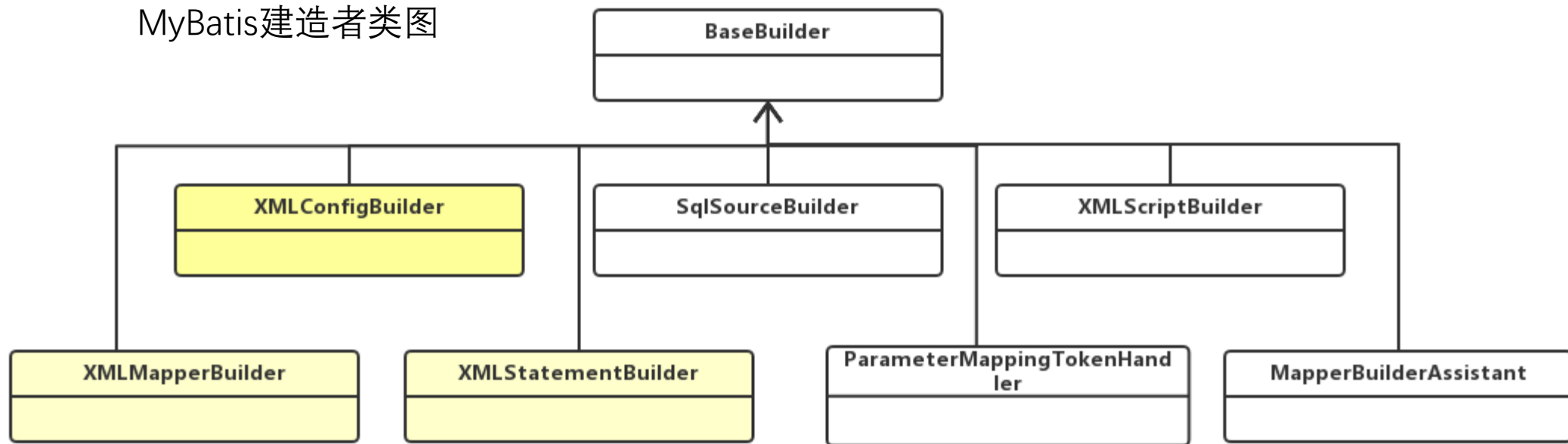
- ✓ **Builder** : 给出一个抽象接口，以规范产品对象的各个组成成分的建造。这个接口规定要实现复杂对象的哪些部分的创建，并不涉及具体的对象部件的创建；
- ✓ **ConcreteBuilder** : 实现Builder接口，针对不同的商业逻辑，具体化复杂对象的各部分的创建。在建造过程完成后，提供产品的实例；
- ✓ **Director** : 调用具体建造者来创建复杂对象的各个部分，在指导者中不涉及具体产品的信息，只负责保证对象各部分完整创建或按某种顺序创建；
- ✓ **Product** : 要创建的复杂对象



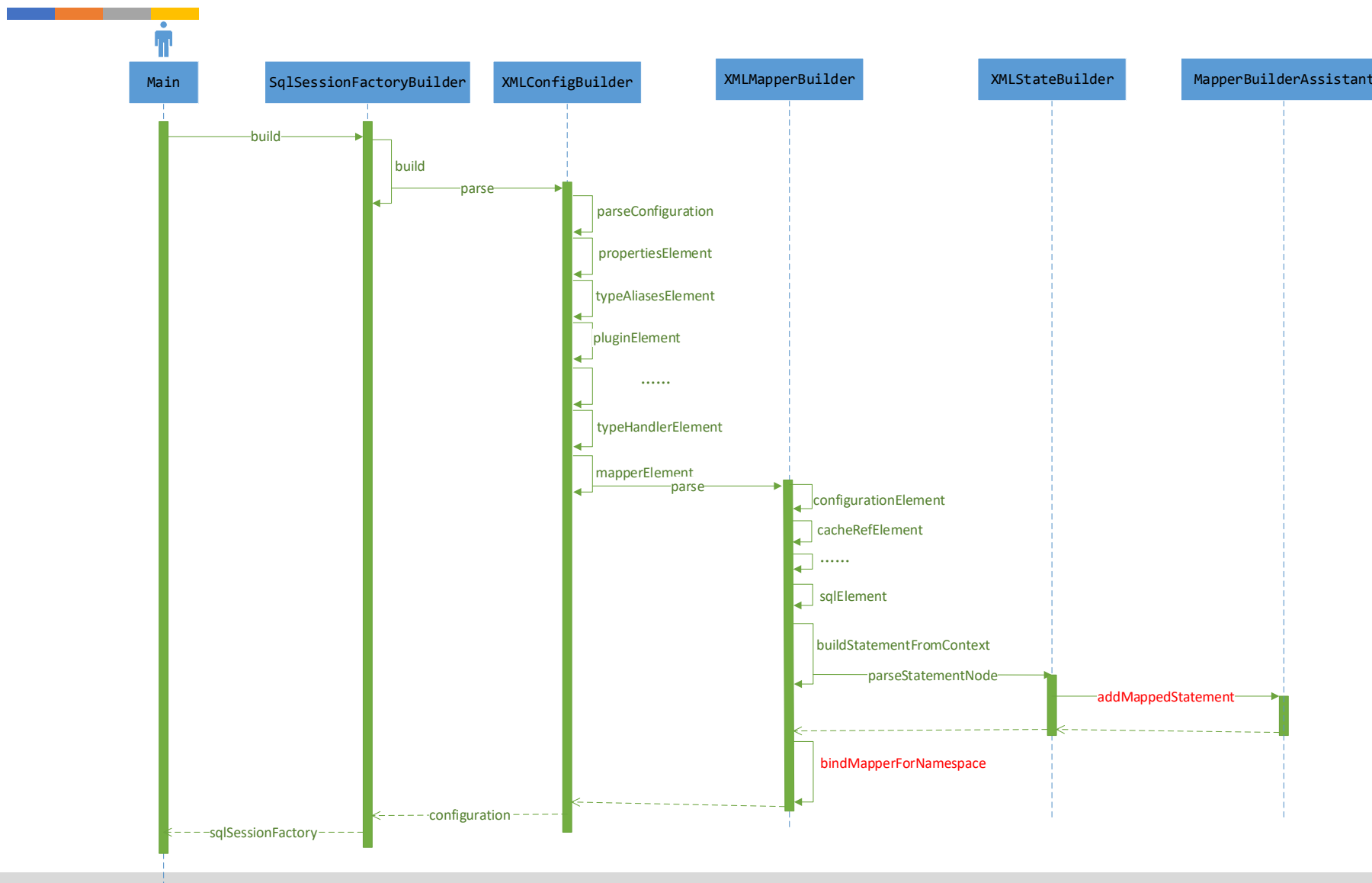
建造者模式 使用场景

- ✓ 需要生成的对象具有复杂的内部结构，实例化对象时要屏蔽掉对象内部的细节，让上层代码与复杂对象的实例化过程解耦，可以使用建造者模式；简而言之，如果“遇到多个构造器参数时要考虑用构建器”；
- ✓ 一个对象的实例化是依赖各个组件的产生以及装配顺序，关注的是一步一步地组装出目标对象，可以使用建造器模式；

MyBatis建造者类图



MyBatis初始化过程





映射器的关键类



待完善：

- ✓ ResultMap
- ✓ MappedStatement
- ✓ SqlSource
- ✓ BoundSql
- ✓ MapperRegistry



目录

CONTENTS



MyBatis核心流程分析

MyBatis核心流程分析



配置加载阶段

建造者模式
mybatis初始化过程



Mybatis的接口层

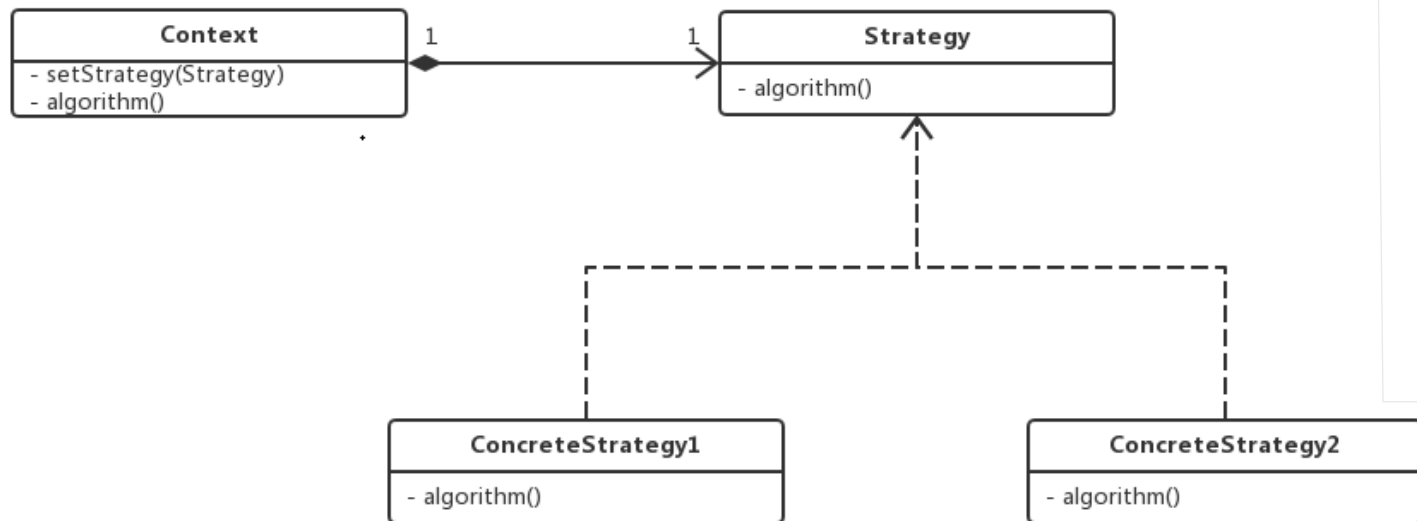
策略模式
sqlSession相关的类



核心组件Executor

模板模式
Executor组件分析
Executor的三个小弟

- **策略模式 (Builder Pattern)** 策略模式定义了一系列的算法，并将每一个算法封装起来，而且使他们可以相互替换，让算法独立于使用它的客户而独立变化。



策略模式的使用场景：

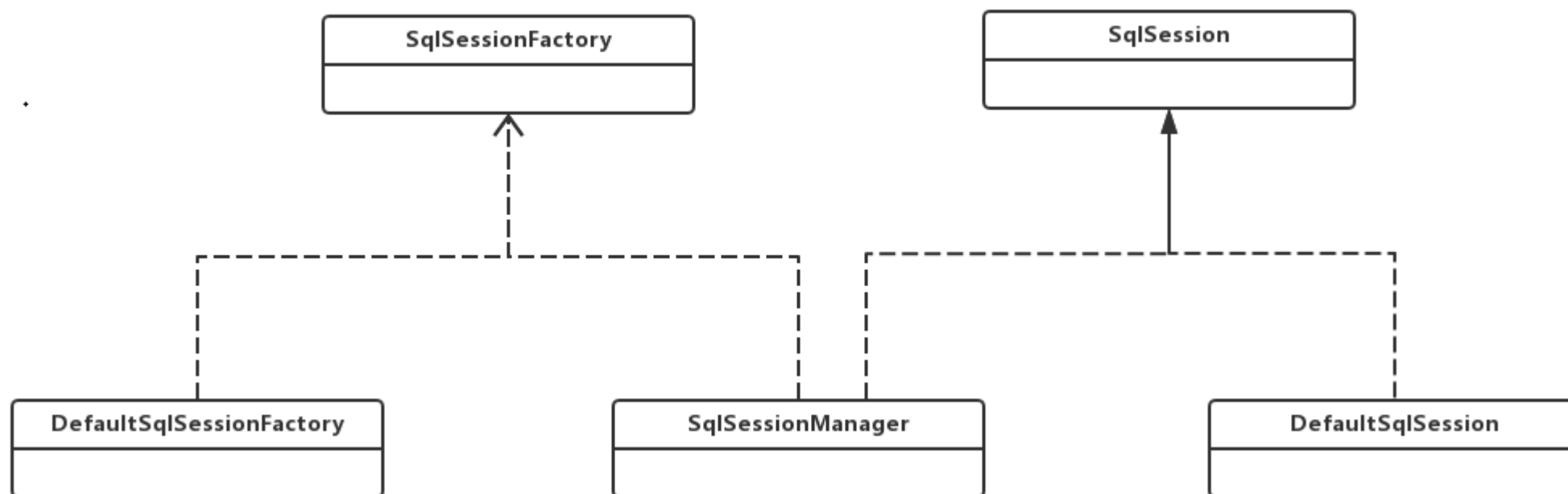
- ✓ 针对同一类型问题的多种处理方式，仅仅是具体行为有差别时；
- ✓ 出现同一抽象类有多个子类，而又需要使用 if-else 或者 switch-case 来选择具体子类时。

- ✓ **Context**：算法调用者，使用setStrategy方法灵活的选择策略（strategy）；
- ✓ **Strategy**：算法的统一接口；
- ✓ **ConcreteStrategy**：算法的具体实现；

SqlSession相关类UML



- **SqlSession**是MyBaits对外提供的最关键的接口，通过它可以执行数据库读写命令、获取映射器、管理事务等；

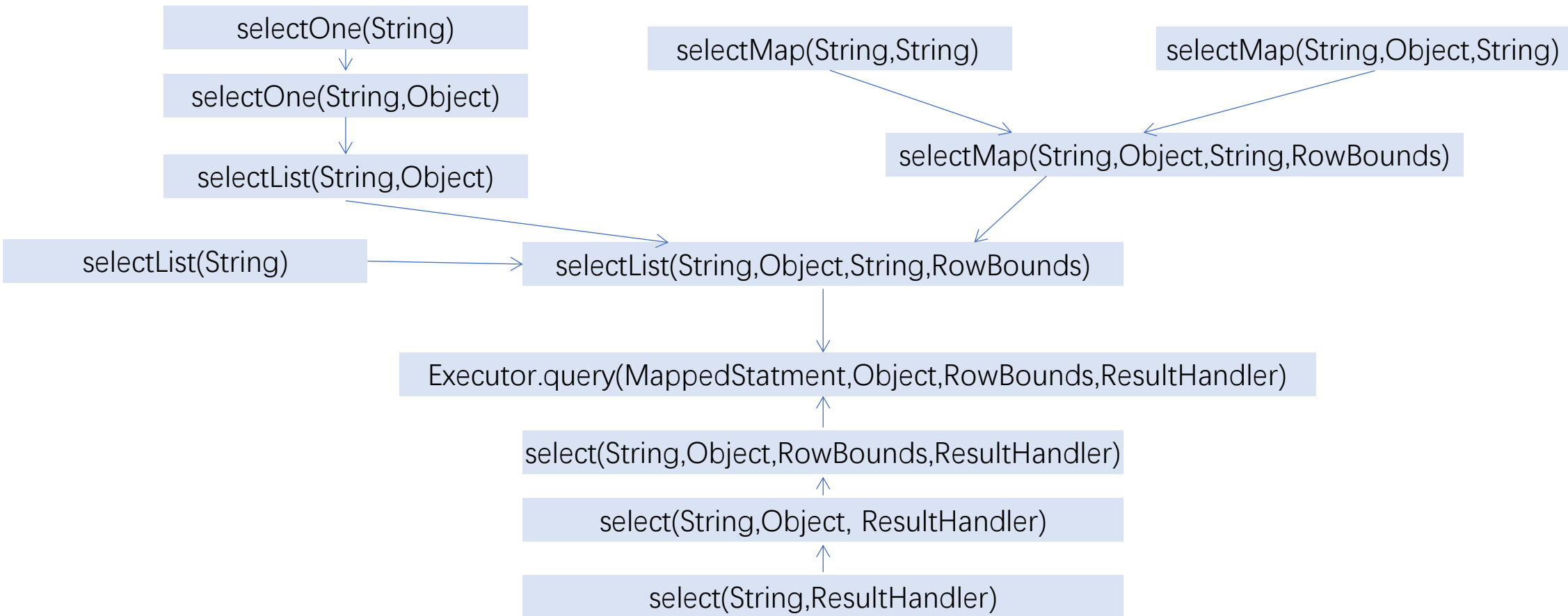




- sqlSessionManager同时继承了SqlSession接口和SqlSessionFactory接口，提供了创建SqlSession对象和操纵数据库的能力；
- SqlSessionManager有两种获取SqlSession的模式：
 - ✓ 第一种模式和SqlSessionFactory 相同，同一个线程每次访问数据库，每次都可以创建新的SqlSession对象；
 - ✓ 第二种模式，同一个线程每次访问数据库，都是使用同一个SqlSession对象,通过localSqlSession实现；



SqlSession查询接口嵌套关系





目录

CONTENTS



MyBatis核心流程分析

MyBatis核心流程分析



配置加载阶段

建造者模式
mybatis初始化过程



Mybatis的接口层

策略模式
sqlSession相关的类



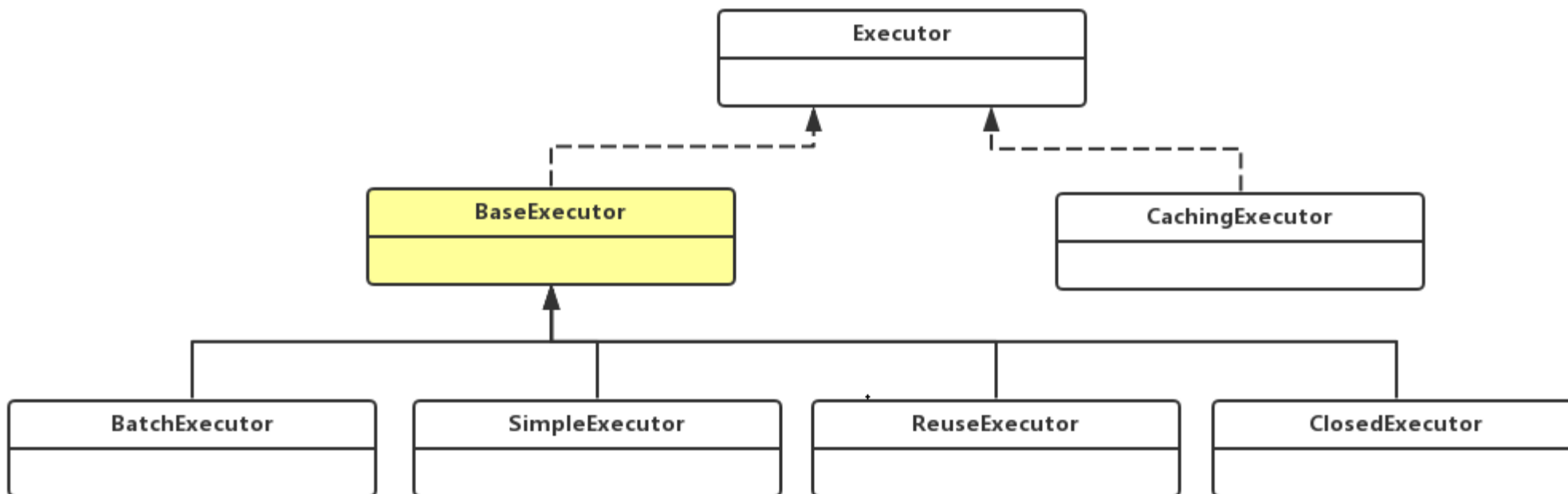
核心组件Executor

模板模式
Executor组件分析
Executor的三个小弟

Executor组件分析



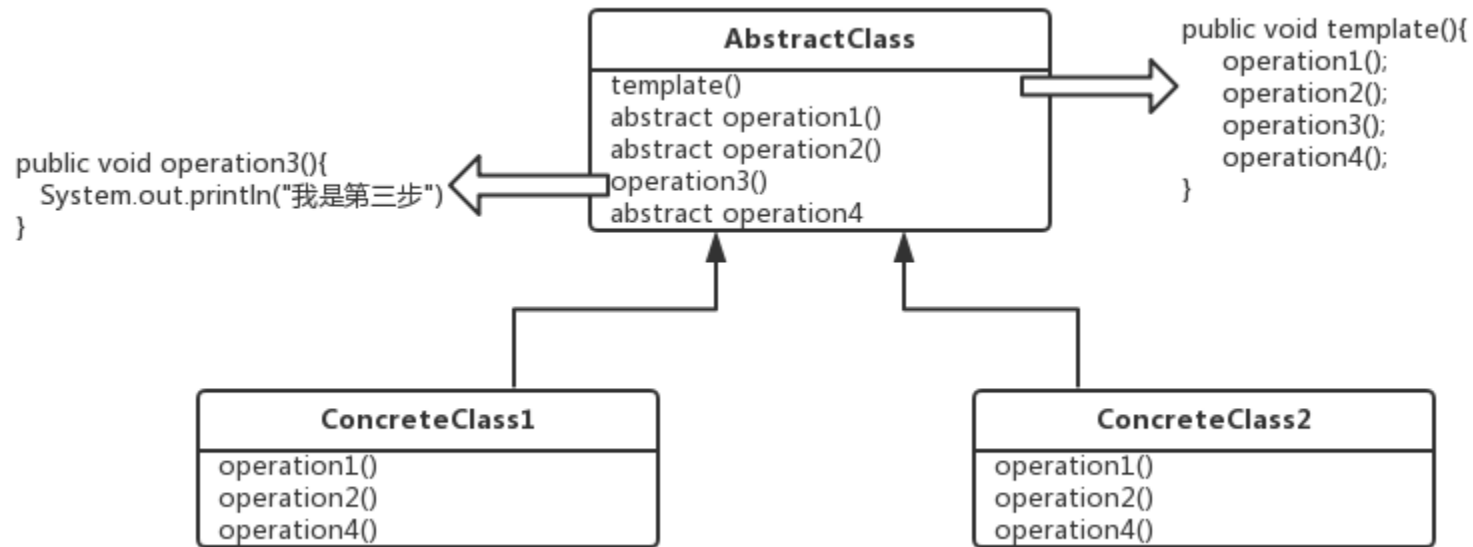
- **Executor**是MyBaits核心接口之一，定义了数据库操作最基本的方法，SqlSession的功能都是基于它来实现的；





模板模式

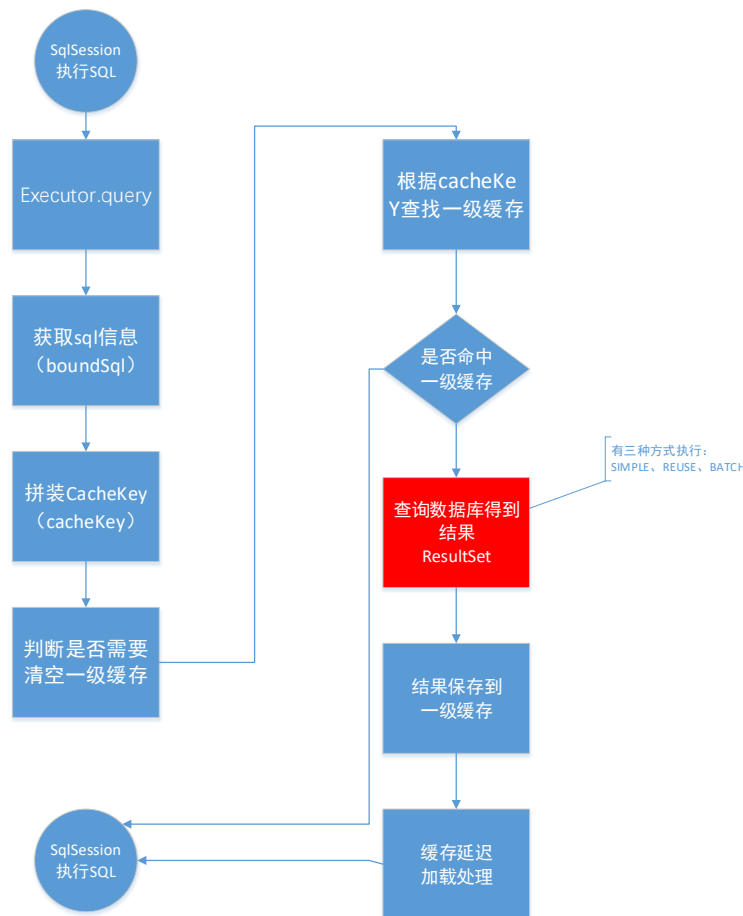
- **模板模式 (Template Pattern)**：一个抽象类公开定义了执行它的方法的方式/模板。它的子类可以按需要重写方法实现，但调用将以抽象类中定义的方式进行。定义一个操作中的算法的骨架，而将一些步骤延迟到子类中。模板方法使得子类可以不改变一个算法的结构即可重定义该算法的某些特定实现；



模板模式应用场景



遇到由一系列步骤构成的过程需要执行，这个过程从高层次上看是相同的，但是有些步骤的实现可能不同，这个时候就需要考虑用模板模式了。比如：Executor查询操作流程：





Executor的三个实现类解读

- SimpleExecutor：默认配置，使用statement对象访问数据库，每次访问都要创建新的statement对象；
- ReuseExecutor：使用预编译PreparedStatement对象访问数据库，访问时，会重用缓存中的statement对象；
- BatchExecutor：实现批量执行多条SQL语句的能力；



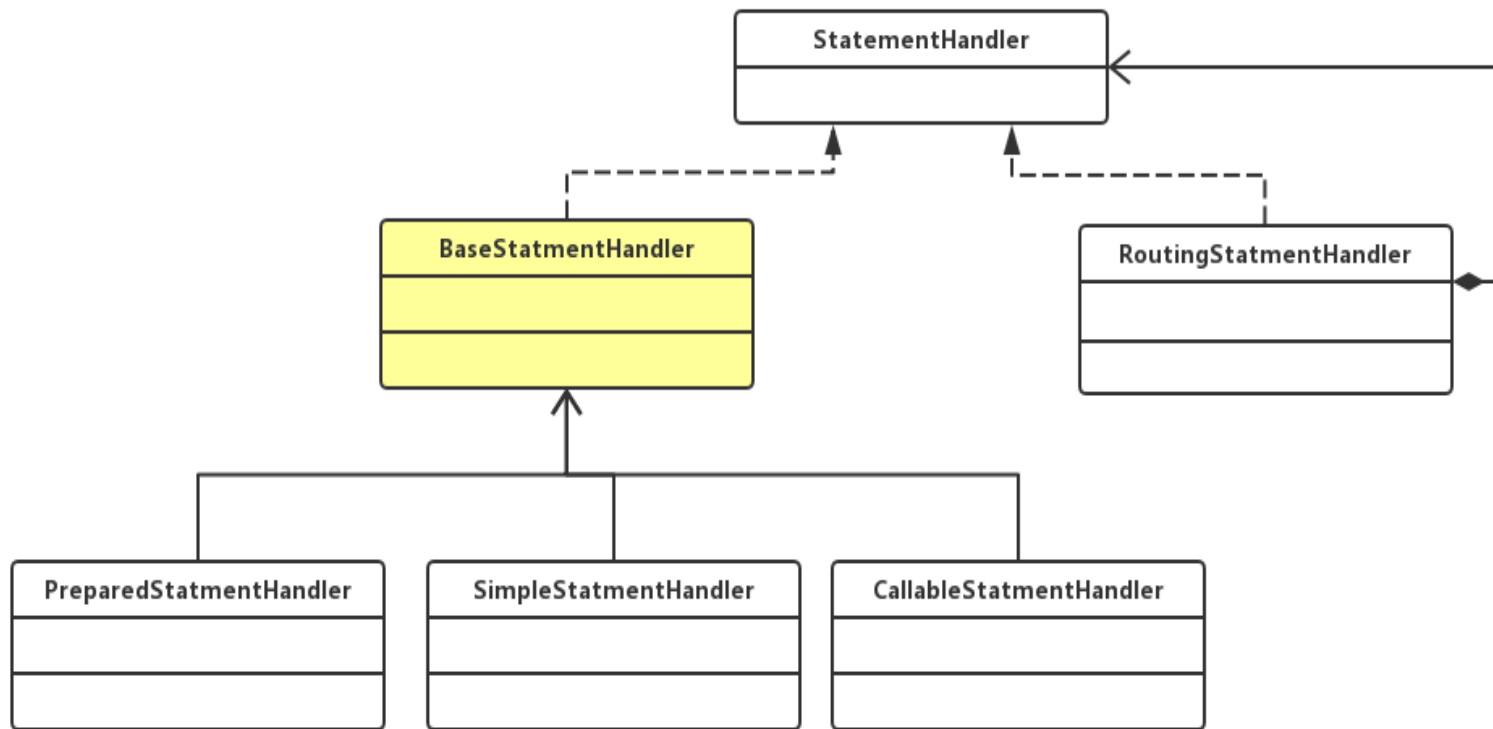
Executor的三个重要小弟

- 通过对SimpleExecutor doQuery()方法的解读发现，Executor是个指挥官，它在调度三个小弟工作：
 - StatementHandler：它的作用是使用数据库的Statement或PreparedStatement执行操作，启承上启下作用；
 - ParameterHandler：对预编译的SQL语句进行参数设置，SQL语句中的占位符“？”都对应BoundSql.parameterMappings集合中的一个元素，在该对象中记录了对应的参数名称以及该参数的相关属性
 - ResultHandler：对数据库返回的结果集（ResultSet）进行封装，返回用户指定的实体类型；



StatementHandler分析

- StatementHandler完成Mybatis最核心的工作，也是Executor实现的基础；功能包括：创建statement对象，为sql语句绑定参数，执行增删改查等SQL语句、将结果映射集进行转化；



- ✓ **BaseStatementHandler**：所有子类的抽象父类，定义了初始化statement的操作顺序，由子类实现具体的实例化不同的statement（模板模式）；
- ✓ **RoutingStatementHandler**：Excutor组件真正实例化的子类，使用静态代理模式，根据上下文决定创建哪个具体实体类；
- ✓ **SimpleStatmentHandler**：使用statement对象访问数据库，无须参数化；
- ✓ **PreparedStatmentHandler**：使用预编译PrepareStatement对象访问数据库；
- ✓ **CallableStatmentHandler**：调用存储过程；

ResultHandler分析



为什么使用mapper接口就能操作数据库？



sqlSession对数据库执行一次查询操作的时序图？描述主要流程

