Mybatis基础

# 基础知识

## 概述

MyBatis是一个支持普通SQL查询、存储过程以及高级映射的持久层框架，它消除了几乎所有的JDBC代码和参数的手动设置以及对结果集的检索，并使用简单的XML或注解进行配置和原始映射，用以将接口和Java的POJO映射成数据库中的记录。

MyBatis也被称为ORM框架。ORM框架的工作原理如下图。



POJO：（Plain Ordinary Java Object）简单的Java对象。可以理解为一个类，这个类不包含任何业务逻辑，跟数据表字段一一对应，每个字段都有对应的get和set方法。在某些框架中，跟VO、Entity其实是差不多的概念。具体可以参考如下链接：

<https://baike.baidu.com/item/POJO/3311958?fr=aladdin>

ORM框架：ORM（Object Relational Mapping）框架采用[元数据](https://baike.baidu.com/item/%E5%85%83%E6%95%B0%E6%8D%AE/1946090)来描述对象——关系映射细节，元数据一般采用XML格式，并且存放在专门的对象一映射文件中。具体可以参考如下链接：

https://baike.baidu.com/item/ORM%E6%A1%86%E6%9E%B6/15541111?fr=aladdin

## MyBatis和Hibernate的比较

Hebernate和MyBatis都是常见的ORM框架，区别如下：

Hebernate：一个全表映射的框架。适合场景不太复杂且对性能要求不高的项目中使用。

优点：

1. 开发者只需定义好持久化对象（即POJO）到数据库表的映射关系，就可以通过Hibername提供的方法完成持久层操作。
2. 开发者不需要熟练地掌握SQL语句的编写，Hibername会根据制定的存储逻辑，自动生成对应的SQL，并调用JDBC接口来执行。
3. 开发效率会高于MyBatis（当然这种说法是基于存在牛逼架构师和十分理解Hibernate使用方法的前提）。
4. 跨数据库开发迁移方便。

缺点：

1. 封装度强，学习门槛高。
2. 封装度太强，多表关联时，对SQL查询的支持较差。
3. 更新数据时，需要发送所有字段。
4. 不支持存储过程。
5. 不能通过优化SQL来优化性能。

MyBatis：一个半自动映射的框架。适合一些复杂的和需要优化性能的项目使用。

优点：

1. 可以配置动态SQL并优化SQL。
2. 可以通过配置决定SQL的映射规则。
3. 支持存储过程。
4. 学习门槛和Hibernate对比低。

缺点：

1. 需要手动匹配提供POJO、SQL和映射关系，相对Hibernate多了手动配置SQL的操作。
2. 开发工作量比较大。

## MyBatis的工作原理

原理如下图：



**mybatis-config.xml：**配置MyBatis的运行环境信息，其中主要内容是数据库连接。

**Mapper.xml：**Mapper.xml文件即SQL映射文件，该文件配置了各种操作数据库的SQL语句。这种文件需要在mybatis-config.xml中加载才能执行。mybatis-config.xml可以加载多个配置文件，每个配置文件对应数据库的一张表。

**SqlSessionFactory：**会话工厂是一个接口类，基本上方法的功能主要是获取mybatis-config.xml的环境配置信息。在源码中可以看到有一个DefaultSqlSessionFactory类和SqlSessionManager实现SqlSssionFactory类的方法。初步看DefaultSqlSessionFactory类单纯实现了SqlSessionFactory的方法，而SqlSessionManager类既实现了SqlSessionFactory类和SqlSession类的方法。

**SqlSession：**SqlSession对象由会话工厂创建，该对象包含了执行SQL的所有方法。包括增删查改、提交回退等等的方法。在源码中可以看到有一个DefaultSqlSession和SqlSessionManager实现SqlSession类的方法。其中有一个selectOne方法，源码能看到如果SQL查询的结果超过一条，是会报异常的。

**Executor：**MyBatis底层定义了一个Executor接口来操作数据库，它会根据SqlSession传递的参数动态地生成需要执行的SQL语句，同时负责查询缓存的维护。从源码看来，Executor也有增删查改的方法，不过没有SqlSession这么多元化，例如查这方面只有query方法，增删改主要是update方法，区别就在于接受SqlSession传递的参数执行数据库操作了。在源码中，初步能看到一个抽象类BaseExecutor和另一个类CachingExecutor实现Executor接口的方法。

**MappedStatement：**在Executor接口的执行方法中，包含一个MappedStatement类型的参数，该参数是对映射信息的封装，用于存储要映射的SQL语句的id、参数等。Mapper.xml文件中一个SQL对应一个MappedStatement对象，SQL的id即是MappedStatement的id。从源码中可以看到，MappedStatement这个类是一个final修饰的类，也就是说不允许存在子类的类。另外，里面存在一个Builder的内部类。

**输入参数映射：**在执行方法时，MappedStatement对象会对用户执行SQL语句输入参数进行定义（可以定义为Map、List类型、基本类型和POJO类型），Executor执行器会通过MappedStatement对象在执行SQL前，将输入的Java对象映射到SQL语句中。这里对输入参数的映射过程就类似于JDBC编程中preparedStatement对象设置参数的过程。

**输出参数映射：**在执行完SQL语句后，MappedStatement对象会对执行输出的结果进行定义（可以定义为Map、List类型、基本类型和POJO类型），Executor执行器会通过MappedStatement对象在执行SQL语句后，将输出结果映射到Java对象中。这种输出结果映射到Java对象的过程类似于JDBC编程中对结果的解析处理过程。

# 学习过程中踩过的坑

1. Mybatis-config.xml里面，environments中可以有多个environment，每个environment都有自己的id，而environments的default属性用于选择用哪一个environment。用处是可以通过简单的配置切换开发、测试、UAT以及生产环境。
2. Mapper.xml中的SQL配置，如果参数类型parameterType为String类型，则变量名必须为value或者\_parameter，否则会报异常“Cause: org.apache.ibatis.reflection.ReflectionException: There is no getter for property named 'username' in 'class java.lang.String'”
3. Mapper.xml中的SQL配置，如果是模糊查询（即使用like），参数不能只用’%#{value}%’，会报错。网上很多方法，例如’%${value}%’这种非预编译的方式，但是这样不能解决SQL注入的问题；又例如CONCAT('%',#{name},'%')、'%'+#{name}+'%'或者'%'||#{name}||'%'，这种又会存在不同数据库存在不同语法的问题。比较好的方式是使用bind标签。Bind标签同样需要使用value或者\_parameter去支撑String类型的参数，SQL语句引用Bind标签的时候不能使用单引号。
4. 针对增删改的SQL，mybatis在执行完的时候基本都会返回变更的条数，如果SQL存在语法等问题导致无法执行，一般都会通过报异常的方式报错，而不会返回数值。
5. 针对增删改的SQL，经过测试，如果sqlSession不主动commit数据，SQL执行完毕的事务是会自动回退的