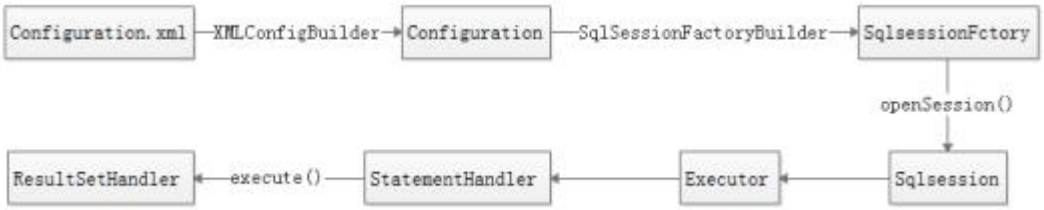
Mybatis 底层源码分析

一、 MyBatis 回顾

MyBatis 案例

二、 Mybatis 执行流程

1 MyBatis 流程图



2 MyBatis 源码分析

2.1 Configuration.xml

该配置文件是 MyBatis 的全局配置文件，在这个文件中可以配置诸多项目。常用的内容是别名设置，拦截器设置等。

2.1.1 Properties（属性）

将数据库连接参数单独配置在 db.properties 中，放在类路径下。这样只需要在 SqlMapConfig.xml 中加载 db.properties 的属性值。这样在 SqlMapConfig.xml 中就不需要对数据库连接参数硬编码。

将数据库连接参数只配置在 db.properties 中，原因：方便对参数进行统一管理

2.1.2 Settings（全局配置参数）

Mybatis 全局配置参数，全局参数将会影响 mybatis 的运行行为。比如：开启二级缓存、开启延迟加载

2.1.3 TypeAliases（类型别名）

类型别名是为 Java 类型命名一个短的名字。它只和 XML 配置有关, 只用来减少类完全限定名的多余部分

2.1.4Plugins（插件）

MyBatis 允许你在某一点拦截已映射语句执行的调用。默认情况下,MyBatis 允许使用插件来拦截方法调用

2.1.5Environments（环境集合属性对象）

MyBatis 可以配置多种环境。这会帮助你将 SQL 映射应用于多种数据库之中。但是要记得一个很重要的问题：你可以配置多种环境，但每个数据库对应一个 SqlSessionFactory。

所以，如果你想连接两个数据库，你需要创建两个 SqlSessionFactory 实例，每个数据库对应一个。

2.1.5.1 Environment（环境子属性对象）

2.1.5.1.1TransactionManager（事务管理）

在 MyBatis 中有两种事务管理器类型(也就是 type=”[JDBC|MANAGED]”)

2.1.5.1.2DataSource（数据源）

UNPOOLED|POOLED|JNDI

2.1.6Mappers（映射器）

指定映射配置文件位置

```
<mapper resource="org/mybatis/builder/AuthorMapper.xml"/>
<mapper url="file:///var/mappers/AuthorMapper.xml"/>
<mapper class="org.mybatis.builder.AuthorMapper"/>
<package name="org.mybatis.builder"/>
```

2.2Mapper.xml

Mapper.xml 映射文件中定义了操作数据库的 sql，每个 sql 是一个 statement，映射文件是 mybatis 的核心

2.2.1ResultMap

Mybatis 中可以使用 resultMap 完成高级输出结果映射。如果查询出来的列名和定义的 pojo 属性名不一致，就可以通过定义一个 resultMap 对列名和 pojo 属性名之间作一个映射关系。

2.2.2Cache

开启二级缓存

2.2.3Select

查询语句

2.2.4Insert

插入语句

2.2.5Update

更新语句

2.2.6Delete

删除语句

2.2.7Sql

可以重用的 SQL 块,也可以被其他语句引用

2.3Resources

Resources 工具类会从路径中加载资源，并返回一个输入流对象，对于资源文件的加载提供了简易的使用方法。

加载一个资源有很多方式：

对于简单的只读文本数据，加载为 Reader。

对于简单的只读二进制或文本数据，加载为 Stream。

对于可读写的二进制或文本文件，加载为 File。

对于只读的配置属性文件，加载为 Properties。

对于只读的通用资源，加载为 URL。

按以上的顺序，Resources 类加载资源的方法如下：

```
Reader getResourceAsReader(String resource);
Stream getResourceAsStream(String resource);
File getResourceAsFile(String resource);
Properties getResourceAsProperties(String resource);
Url getResourceAsUrl(String resource);
```

2.4SqlSessionFactoryBuilder

该类是 SqlSessionFactory（会话工厂）的构建者类，之前描述的操作其实全是从这里面开启的，首先就是调用 XMLConfigBuilder 类的构造器来创建一个 XML 配置构建器对象，利用这个构建器对象来调用其解析方法 parse()来完成 Configuration 对象的创建，之后以这个配置对象为参数调用会话工厂构建者类中的 build(Configuration config)方法来完成

SqlSessionFactory（会话工厂）对象的构建。

2.5 XMLConfigBuilder

该类是 XML 配置构建者类，是用来通过 XML 配置文件来构建 Configuration 对象实例，构建的过程就是解析 Configuration.xml 配置文件的过程，期间会将从配置文件中获取到的指定标签的值逐个添加到之前创建好的默认 Configuration 对象实例中。

2.6 Configuration

该对象是 Mybatis 的上下文对象，实例化这个类的目的就是为了使用其对象作为项目全局配置对象，这样通过配置文件配置的信息可以保存在这个配置对象中，而这个配置对象在创建好之后是保存在 JVM 的 Heap 内存中的，方便随时读取。不然每次需要配置信息的时候都要临时从磁盘配置文件中获取，代码复用性差的同时，也不利于开发

2.7 DefaultSqlSessionFactory

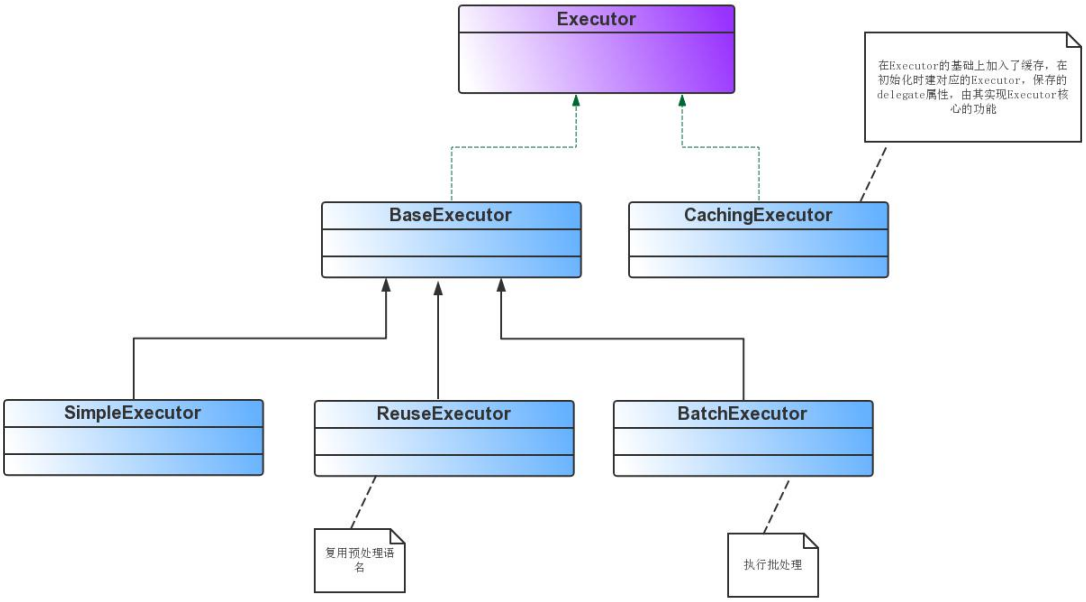
SqlsessionFactory 该接口是会话工厂，是用来生产会话的工厂接口，DefaultSqlSessionFactory 是其实现类，是真正生产会话的工厂类，这个类的实例的生命周期是全局的，它只会在首次调用时生成一个实例（单例模式），就一直存在直到服务器关闭。

2.8 Executor

执行器接口，SqlSession 会话是面向程序员的，而内部真正执行数据库操作的却是 Executor 执行器，可以将 Executor 看作是面向 MyBatis 执行环境的，SqlSession 就是门面货，Executor 才是实干家。通过 SqlSession 产生的数据库操作，全部是通过调用 Executor 执行器来完成的。

Executor 是跟 SqlSession 绑定在一起的，每一个 SqlSession 都拥有一个新的 Executor 对象，由 Configuration 创建。

2.8.1Executor 继承结构



2.8.2BaseExecutor

SimpleExecutor：每执行一次 update 或 select，就开启一个 Statement 对象，用完立刻关闭 Statement 对象。（可以是 Statement 或 PrepareStatement 对象）

ReuseExecutor：执行 update 或 select，以 sql 作为 key 查找 Statement 对象，存在就使用，不存在就创建，用完后，不关闭 Statement 对象，而是放置于 Map<String, Statement>内，供下一次使用。（可以是 Statement 或 PrepareStatement 对象）

BatchExecutor：执行 update（没有 select，JDBC 批处理不支持 select），将所有 sql 都添加到批处理中（addBatch()），等待统一执行（executeBatch()），它缓存了多个 Statement 对象，每个 Statement 对象都是 addBatch()完毕后，等待逐一执行

2.8.3CachingExecutor

先从缓存中获取查询结果，存在就返回，不存在，再委托给 Executor delegate 去数据库取，delegate 可以是上面任一的 SimpleExecutor、ReuseExecutor、BatchExecutor。

2.9StatementHandler

该类是 Statement 处理器，封装了 Statement 的各种数据库操作方法 execute()，可见 MyBatis 其实就是将操作数据库的 JDBC 操作封装起来的一个框架，同时还实现了 ORM 罢了。

RoutingStatementHandler，这是一个封装类，它不提供具体的实现，只是根据 Executor 的类型，创建不同的类型 StatementHandler。

2.10ResultSetHandler

结果集处理器，如果是查询操作，必定会有返回结果，针对返回结果的操作，就要使用 ResultSetHandler 来进行处理，这个是由 StatementHandler 来进行调用的。这个处理器的作用就是对返回结果进行处理。