MyBatis 的真正强大在于它的映射语句，这是它的魔力所在。由于它的异常强大，映射器的 XML 文件就显得相对简单。如果拿它跟具有相同功能的 JDBC 代码进行对比，你会立即发现省掉了将近 95% 的代码。MyBatis 为聚焦于 SQL 而构建，以尽可能地为你减少麻烦。

SQL 映射文件只有很少的几个顶级元素（按照应被定义的顺序列出）：

* cache – 对给定命名空间的缓存配置。
* cache-ref – 对其他命名空间缓存配置的引用。
* resultMap – 是最复杂也是最强大的元素，用来描述如何从数据库结果集中来加载对象。
* ~~parameterMap – 已被废弃！老式风格的参数映射。更好的办法是使用内联参数，此元素可能在将来被移除。文档中不会介绍此元素。~~
* sql – 可被其他语句引用的可重用语句块。
* insert – 映射插入语句
* update – 映射更新语句
* delete – 映射删除语句
* select – 映射查询语句

下一部分将从语句本身开始来描述每个元素的细节。

## select

查询语句是 MyBatis 中最常用的元素之一，光能把数据存到数据库中价值并不大，只有还能重新取出来才有用，多数应用也都是查询比修改要频繁。对每个插入、更新或删除操作，通常间隔多个查询操作。这是 MyBatis 的基本原则之一，也是将焦点和努力放在查询和结果映射的原因。简单查询的 select 元素是非常简单的。比如：

<select id="selectPerson" parameterType="int" resultType="hashmap">

SELECT \* FROM PERSON WHERE ID = #{id}</select>

这个语句被称作 selectPerson，接受一个 int（或 Integer）类型的参数，并返回一个 HashMap 类型的对象，其中的键是列名，值便是结果行中的对应值。

注意参数符号：

#{id}

这就告诉 MyBatis 创建一个**预处理语句**（PreparedStatement）参数，在 JDBC 中，这样的一个参数在 SQL 中会由一个“?”来标识，并被传递到一个新的预处理语句中，就像这样：

// 近似的 JDBC 代码，非 MyBatis 代码...String selectPerson = "SELECT \* FROM PERSON WHERE ID=?";PreparedStatement ps = conn.prepareStatement(selectPerson);

ps.setInt(1,id);

当然，使用 JDBC 意味着需要更多的代码来提取结果并将它们映射到对象实例中，而这就是 MyBatis 节省你时间的地方。参数和结果映射还有更深入的细节。这些细节会分别在后面单独的小节中呈现。

select 元素允许你配置很多属性来配置每条语句的作用细节。

<select

id="selectPerson"

parameterType="int"

parameterMap="deprecated"

resultType="hashmap"

resultMap="personResultMap"

flushCache="false"

useCache="true"

timeout="10"

fetchSize="256"

statementType="PREPARED"

resultSetType="FORWARD\_ONLY">

|  |
| --- |
|  |

### resultType&resultMap

参看这里：github\poeny-mybatis-all\NativeMyBatis\src\main\resources\XMLMappingSqlFile\resultMapUseDemo

### statementType说明

1.原文链接：https://blog.csdn.net/u012060033/article/details/85678693  
 在mapper文件中可以使用statementType标记使用什么的对象操作SQL语句。   
  
 statementType：标记操作SQL的对象   
  
 要实现动态传入表名、列名，需要做如下修改 ，添加属性statementType=”STATEMENT” ，同时sql里的属有变量取值都改成${xxxx}，而不是#{xxx}  
  
  
 注意！注意！注意！重点！重点！重点！  
 取值说明：   
 1、STATEMENT:直接操作sql，不进行预编译，获取数据：$—Statement   
 2、PREPARED:预处理，参数，进行预编译，获取数据：#—–PreparedStatement:默认   
 3、CALLABLE:执行存储过程————CallableStatement   
  
 另外说明下：如果只为STATEMENT，那么sql就是直接进行的字符串拼接，这样为字符串需要加上引号，  
 如果为PREPARED，是使用的参数替换，也就是索引占位符，我们的#会转换为?再设置对应的参数的值

具体实例在这里：

D:\innergitjava\own\aaaaa-----github\poeny-mybatis-all\NativeMyBatis\src\main\resources\XMLMappingSqlFile\resultMapUseDemo\MapperUser.xml

| **属性** | **描述** |
| --- | --- |
| id | 在命名空间中唯一的标识符，可以被用来引用这条语句。 |
| parameterType | 将会传入这条语句的参数类的完全限定名或别名。这个属性是可选的，因为 MyBatis 可以通过类型处理器（TypeHandler） 推断出具体传入语句的参数，默认值为未设置（unset）。 |
| ~~parameterMap~~ | ~~这是引用外部 parameterMap 的已经被废弃的方法。请使用内联参数映射和 parameterType 属性。~~ |
| resultType | 从这条语句中返回的期望类型的类的完全限定名或别名。 注意如果返回的是集合，那应该设置为集合包含的类型，而不是集合本身。可以使用 resultType 或 resultMap，但不能同时使用。 |
| resultMap | 外部 resultMap 的命名引用。结果集的映射是 MyBatis 最强大的特性，如果你对其理解透彻，许多复杂映射的情形都能迎刃而解。可以使用 resultMap 或 resultType，但不能同时使用。 |
| flushCache | 将其设置为 true 后，只要语句被调用，都会导致本地缓存和二级缓存被清空，默认值：false。 |
| useCache | 将其设置为 true 后，将会导致本条语句的结果被二级缓存缓存起来，默认值：对 select 元素为 true。 |
| timeout | 这个设置是在抛出异常之前，驱动程序等待数据库返回请求结果的秒数。默认值为未设置（unset）（依赖驱动）。 |
| fetchSize | 这是一个给驱动的提示，尝试让驱动程序每次批量返回的结果行数和这个设置值相等。 默认值为未设置（unset）（依赖驱动）。 |
| statementType | STATEMENT，PREPARED 或 CALLABLE 中的一个。这会让 MyBatis 分别使用 Statement，PreparedStatement 或 CallableStatement，默认值：PREPARED。 |
| resultSetType | FORWARD\_ONLY，SCROLL\_SENSITIVE, SCROLL\_INSENSITIVE 或 DEFAULT（等价于 unset） 中的一个，默认值为 unset （依赖驱动）。 |
| databaseId | 如果配置了数据库厂商标识（databaseIdProvider），MyBatis 会加载所有的不带 databaseId 或匹配当前 databaseId 的语句；如果带或者不带的语句都有，则不带的会被忽略。 |
| resultOrdered | 这个设置仅针对嵌套结果 select 语句适用：如果为 true，就是假设包含了嵌套结果集或是分组，这样的话当返回一个主结果行的时候，就不会发生有对前面结果集的引用的情况。 这就使得在获取嵌套的结果集的时候不至于导致内存不够用。默认值：false。 |
| resultSets | 这个设置仅对多结果集的情况适用。它将列出语句执行后返回的结果集并给每个结果集一个名称，名称是逗号分隔的。 |

## insert, update 和 delete

数据变更语句 insert，update 和 delete 的实现非常接近：

<insert

id="insertAuthor"

parameterType="domain.blog.Author"

flushCache="true"

statementType="PREPARED"

keyProperty=""

keyColumn=""

useGeneratedKeys=""

timeout="20">

<update

id="updateAuthor"

parameterType="domain.blog.Author"

flushCache="true"

statementType="PREPARED"

timeout="20">

<delete

id="deleteAuthor"

parameterType="domain.blog.Author"

flushCache="true"

statementType="PREPARED"

timeout="20">

| **属性** | **描述** |
| --- | --- |
| id | 命名空间中的唯一标识符，可被用来代表这条语句。 |
| parameterType | 将要传入语句的参数的完全限定类名或别名。这个属性是可选的，因为 MyBatis 可以通过类型处理器推断出具体传入语句的参数，默认值为未设置（unset）。 |
| ~~parameterMap~~ | ~~这是引用外部 parameterMap 的已经被废弃的方法。请使用内联参数映射和 parameterType 属性。~~ |
| flushCache | 将其设置为 true 后，只要语句被调用，都会导致本地缓存和二级缓存被清空，默认值：true（对于 insert、update 和 delete 语句）。 |
| timeout | 这个设置是在抛出异常之前，驱动程序等待数据库返回请求结果的秒数。默认值为未设置（unset）（依赖驱动）。 |
| statementType | STATEMENT，PREPARED 或 CALLABLE 的一个。这会让 MyBatis 分别使用 Statement，PreparedStatement 或 CallableStatement，默认值：PREPARED。 |
| useGeneratedKeys | （仅对 insert 和 update 有用）这会令 MyBatis 使用 JDBC 的 getGeneratedKeys 方法来取出由数据库内部生成的主键（比如：像 MySQL 和 SQL Server 这样的关系数据库管理系统的自动递增字段），默认值：false。 |
| keyProperty | （仅对 insert 和 update 有用）唯一标记一个属性，MyBatis 会通过 getGeneratedKeys 的返回值或者通过 insert 语句的 selectKey 子元素设置它的键值，默认值：未设置（unset）。如果希望得到多个生成的列，也可以是逗号分隔的属性名称列表。 |
| keyColumn | （仅对 insert 和 update 有用）通过生成的键值设置表中的列名，这个设置仅在某些数据库（像 PostgreSQL）是必须的，当主键列不是表中的第一列的时候需要设置。如果希望使用多个生成的列，也可以设置为逗号分隔的属性名称列表。 |
| databaseId | 如果配置了数据库厂商标识（databaseIdProvider），MyBatis 会加载所有的不带 databaseId 或匹配当前 databaseId 的语句；如果带或者不带的语句都有，则不带的会被忽略。 |