# 1 什么是MyBatis？

1、Mybatis是一个半ORM（对象关系映射）框架，它内部封装了JDBC，开发时只需要关注SQL语句本身，不需要花费精力去处理加载驱动、创建连接、创建statement等繁杂的过程。程序员直接编写原生态sql，可以严格控制sql执行性能，灵活度高。

2、MyBatis 可以使用 XML 或注解来配置和映射原生信息，将 POJO映射成数据库中的记录，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

3、通过xml文件或注解的方式将要执行的各种 statement 配置起来，并通过java对象和 statement中sql的动态参数进行映射生成最终执行的sql语句，最后由mybatis框架执行sql并将结果映射为java对象并返回。

# 2、MyBatis的优点和缺点？

优点：

1、基于SQL语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL写在XML里，解除sql与程序代码的耦合，便于统一管理；提供XML标签，支持编写动态SQL语句，并可重用。

2、与JDBC相比，减少了50%以上的代码量，消除了JDBC大量冗余的代码，不需要手动开关连接；

3、很好的与各种数据库兼容（因为MyBatis使用JDBC来连接数据库，所以只要JDBC支持的数据库MyBatis都支持）

4、能够与Spring很好的集成；

5、提供映射标签，支持对象与数据库的ORM字段关系映射；提供对象关系映射标签，支持对象关系组件维护。

缺点：

1、SQL语句的编写工作量较大，尤其当字段多、关联表多时，对开发人员编写SQL语句的功底有一定要求。

2、SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

# 3、#{}和${}的区别是什么？

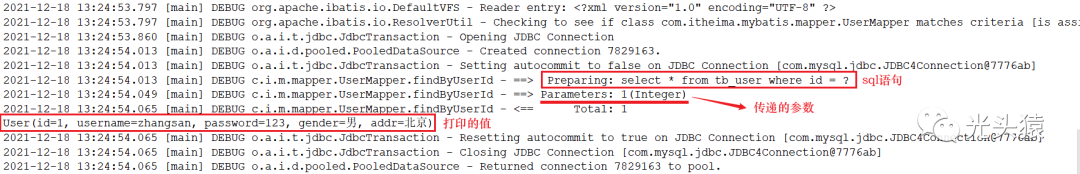
1、#{}是预编译处理，${}是字符串替换。

2、Mybatis在处理#{}时，会将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；

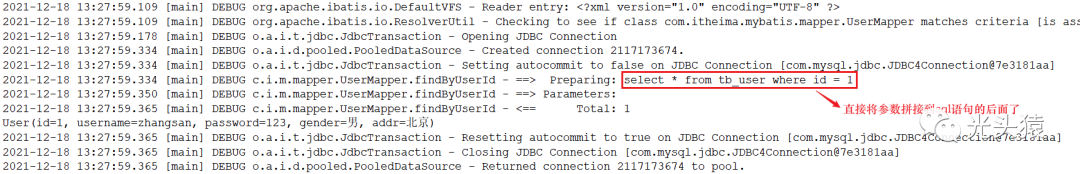
3、Mybatis在处理${}时，就是把${}替换成变量的值。

4、使用#{}可以有效的防止SQL注入，提高系统安全性。

#{}的日志如下所示：

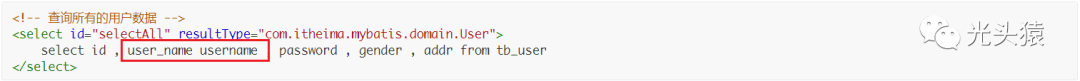


${}的日志如下所示：



# 4、当实体类中的属性名和表中的字段名不一样，怎么办？

第1种方案：通过在查询的sql语句中定义字段名的别名，让字段名的别名和实体类的属性名一致。



第2种方案：使用resultMap来定义字段和属性的映射关系



# 5、Mybatis是如何进行分页的？分页插件的原理是什么？

分页方式：

1、在执行sql语句的时候直接拼接分页参数

2、使用Mybatis的分页插件(PageHelper)进行分页

分页插件的原理：分页插件(PageInterceptor)的基本原理是使用Mybatis提供的插件接口(Interceptor)，实现自定义插件，在插件的拦截方法内拦截待执行的sql，然后重写sql，根据dialect方言，添加对应的物理分页语句和物理分页参数。

举例：select \* from student，拦截sql后重写为：select t.\* from （select \* from student）t limit 0，10

# 6、Mybatis是如何将sql执行结果封装为目标对象并返回的？都有哪些映射形式？

1、使用标签，逐一定义数据库列名和对象属性名之间的映射关系。

2、使用sql列的别名功能，将列的别名书写为对象属性名。

有了列名与属性名的映射关系后，Mybatis通过反射创建对象，同时使用反射给对象的属性逐一赋值并返回，那些找不到映射关系的属性，是无法完成赋值的。

# 7、如何执行批量插入数据？

批量插入数据：

1、mybatis的接口方法参数需要定义为集合类型List<User>

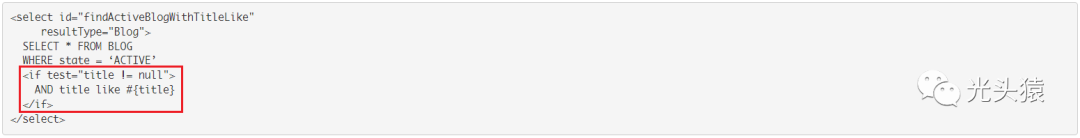
图片

2、在映射文件中通过forEach标签遍历集合，获取每一个元素作为insert语句的参数值

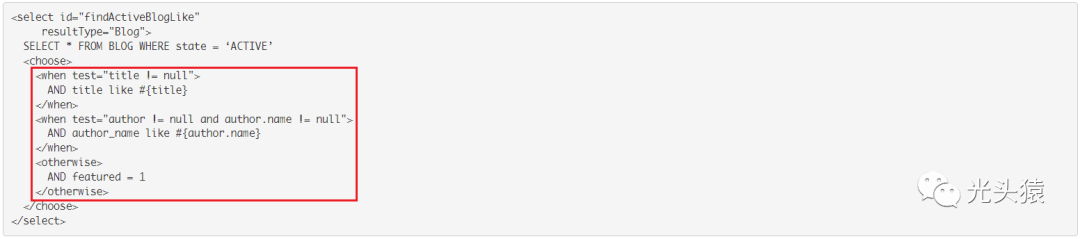


# 8、xml映射文件中，除了常见的select|insert|updae|delete标签之外，还有哪些标签？

**if**：使用动态 SQL 最常见情景是根据条件包含 where 子句的一部分。比如：

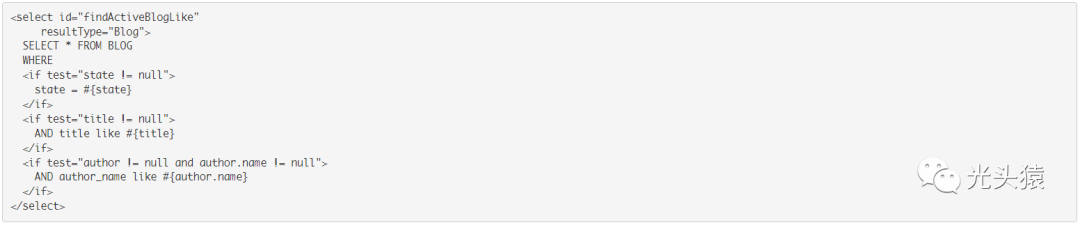


**choose、when、otherwise**：有时候，我们不想使用所有的条件，而只是想从多个条件中选择一个使用。针对这种情况，MyBatis 提供了 choose 元素，它有点像 Java 中的 switch 语句。



还是上面的例子，但是策略变为：传入了 “title” 就按 “title” 查找，传入了 “author” 就按 “author” 查找的情形。若两者都没有传入，就返回标记为 featured 的 BLOG。

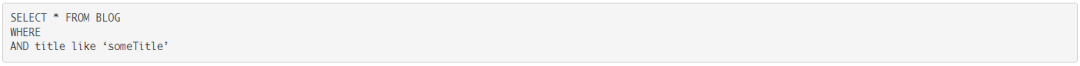
**where、set**: 如下sql语句的定义：



如果没有匹配的条件会怎么样？最终这条 SQL 会变成这样：

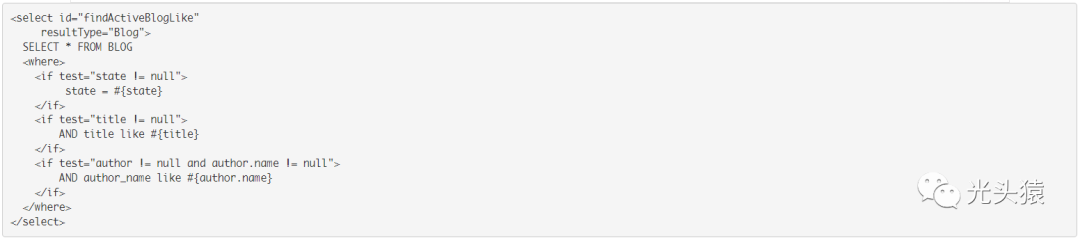
图片

这会导致查询失败。如果匹配的只是第二个条件又会怎样？这条 SQL 会是这样:



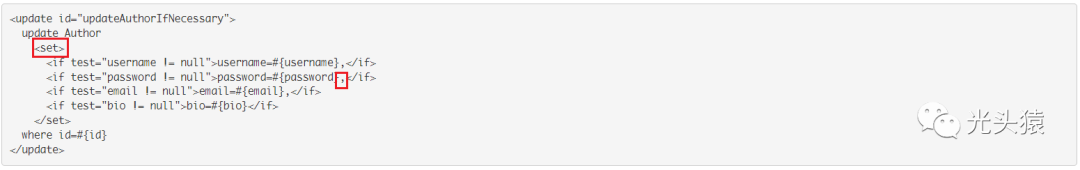
这个查询也会失败。这个问题可以使用简单地用条件元素来解决，比如在where后面添加一个恒等的条件：1 = 1。

MyBatis 有一个简单且适合大多数场景的解决办法。而在其他场景中，可以对其进行自定义以符合需求。而这，只需要一处简单的改动：



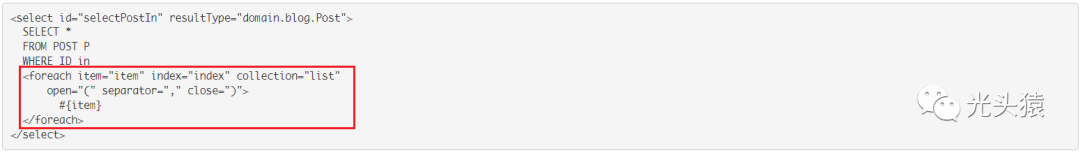
where元素只会在子元素返回任何内容的情况下才插入 “WHERE” 子句。而且，若子句的开头为 “AND” 或 “OR”，*where* 元素也会将它们去除。

set 元素可以用于动态包含需要更新的列，忽略其它不更新的列。比如：



这个例子中，set元素会动态地在行首插入 SET 关键字，并会删掉额外的逗号（这些逗号是在使用条件语句给列赋值时引入的）。

**foreach**: 动态 SQL 的另一个常见使用场景是对集合进行遍历（尤其是在构建 IN 条件语句的时候）。比如：



foreach 元素的功能非常强大，它允许你指定一个集合，声明可以在元素体内使用的集合项（item）和索引（index）变量。它也允许你指定开头与结尾的字符串以及集合项迭代之间的分隔符。这个元素也不会错误地添加多余的分隔符！

# 9、MyBatis实现一对一有几种方式？具体怎么操作的？

两种方式：

1、联合查询

2、嵌套查询

联合查询操作：联合查询是几个表联合查询,只查询一次, 通过在resultMap使用**association**标签配置查询到的关联数据的映射关系。

如下所示：在查询订单的同时需要将订单所对应的用户数据也查询出来。

① 更改订单实体类



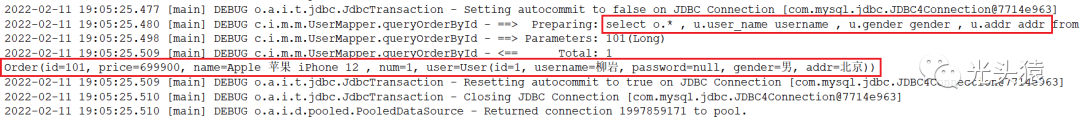
② 映射文件中定义sql语句

图片

③ resultMap定义



④ 执行结果



只发送了一条sql语句，并且进行了数据的封装。

嵌套查询操作：嵌套查询是先查一个表，根据这个表里面的结果的外键id，去再另外一个表里面查询数据,也是通过**association**配置，但另外一个表的查询通过select属性配置。

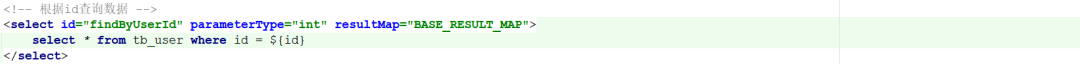
如下所示：在查询订单的同时需要将订单所对应的用户数据也查询出来。

1. 更改订单实体类

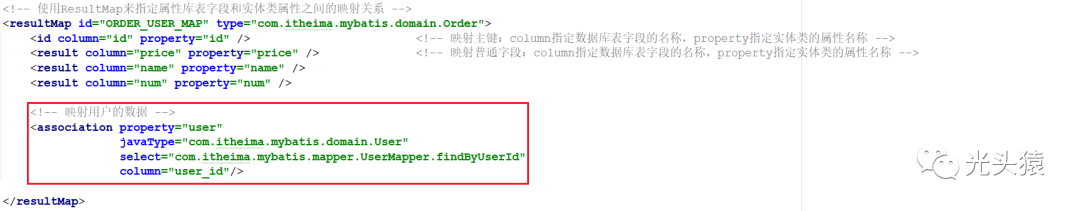


② 映射文件中定义sql语句

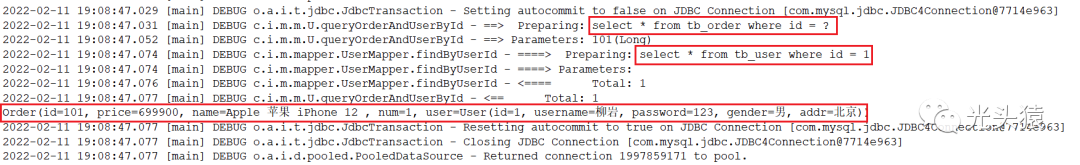
图片



③ resultMap定义



④ 执行结果



只发送了两条sql语句，并且进行了数据的封装。

# 10、Mybatis是否支持延迟加载？如果支持，它的实现原理是什么？

Mybatis仅支持association关联对象和collection关联集合对象的延迟加载，association指的就是一对一，collection指的就是一对多查询。在Mybatis配置文件中，可以配置是否启用延迟加载。

图片

默认情况下延迟加载是关闭的。

实现原理：

它的原理是，使用CGLIB创建**目标对象的代理对象**，当调用目标方法时，进入拦截器方法，比如调用order.getUser().getUserName()，拦截器invoke()方法发现order.getUser()是null值，那么就会单独

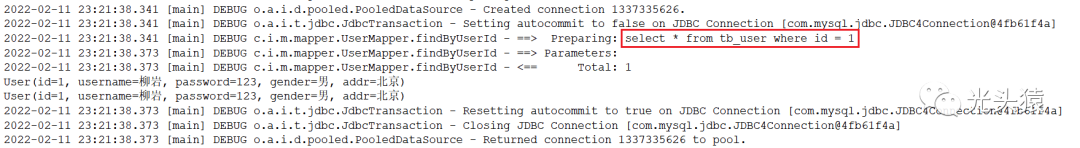
发送事先保存好的查询关联User对象的sql，把User查询上来，然后调用order.setUser(user)，于是order的对象user属性就有值了，接着完成order.getUser().getUserName()方法的调用。

# 11、什么是Mybatis的一级缓存和二级缓存？

一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当Session进行flush或close之后，该Session中的所有Cache就将清空，默认打开一级缓存。如下所示：



使用同一个sqlSession对象获取两次UserMapper对象，进行了两次用户数据的查询。控制台的输出结果如下所示：

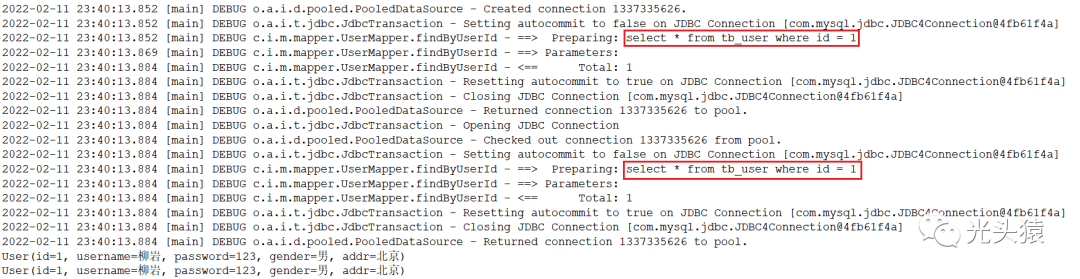


只执行了一次sql语句。说明第二次查询的时候使用的是缓存数据。

二级缓存：二级缓存是基于namespace和mapper的作用域起作用的，不是依赖于SQL session，默认也是采用 PerpetualCache，HashMap 存储。如下代码：



当执行完sqlSession1.close()方法时一级缓存就一斤被清空掉了。再次获取了一个新的sqlSession对象，那么此时就需要再次查询数据，因此控制台的输出如下所示：



可以看到进行了两次查询。

默认情况下二级缓存并没有开启，要想使用二级缓存，那么就需要开启二级缓存，如下所示：

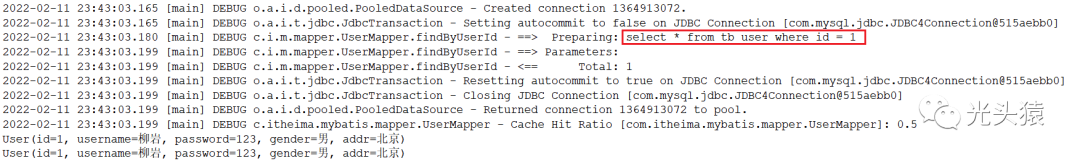
① 全局配置文件

图片

② 映射文件

图片

运行程序进行测试，控制台输出结果如下所示：



只进行了一次查询，那么就说明数据已经进入到了二级缓存中。

3、对于缓存数据更新机制，当某一个作用域(一级缓存 Session/二级缓存Namespaces)的进行了新增、修改、删除操作后，默认该作用域下所有 select 中的缓存将被 clear。

注意事项：

1、二级缓存需要缓存的数据实现Serializable接口

2、只有会话提交或者关闭以后，一级缓存中的数据才会转移到二级缓存中

3、可自定义存储源，如 Ehcache。

# 12、请说说MyBatis的工作原理？



1、读取 MyBatis 配置文件：mybatis-config.xml 为 MyBatis 的全局配置文件，配置了 MyBatis 的运行环境等信息，例如数据库连接信息。

2、加载映射文件。映射文件即 SQL 映射文件，该文件中配置了操作数据库的 SQL 语句，需要在MyBatis 配置文件 mybatis-config.xml 中加载。mybatis-config.xml 文件可以加载多个映射文件，每

个文件对应数据库中的一张表。

3、构造会话工厂：通过 MyBatis 的环境等配置信息构建会话工厂 SqlSessionFactory。

4、创建会话对象：由会话工厂创建 SqlSession 对象，该对象中包含了执行 SQL 语句的所有方法。

5、Executor执行器：MyBatis底层定义了一个Executor 接口来操作数据库，它将根据 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。

6、MappedStatement 对象：在 Executor 接口的执行方法中有一个 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、参数等信息。

7、输入参数映射：输入参数类型可以是 Map、List 等集合类型，也可以是基本数据类型和 POJO 类型。输入参数映射过程类似于 JDBC 对 preparedStatement 对象设置参数的过程。

8、输出结果映射：输出结果类型可以是 Map、 List 等集合类型，也可以是基本数据类型和 POJO 类型。输出结果映射过程类似于 JDBC 对结果集的解析过程。

# 13、Mybatis都有哪些Executor执行器？它们之间的区别是什么？

Mybatis有三种基本的Executor执行器，SimpleExecutor、ReuseExecutor、BatchExecutor。

1、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，也是Mybatis默认使用的执行器。

2、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。简言之，就是重复使用Statement对象。

3、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executeBatch()批处理。与JDBC批处理相同。

# 14、如何获取自动生成的(主)键值?

使用insert标签中的useGeneratedKeys和keyProperty 属性。使用方式如下所示：



属性说明：

1、useGeneratedKeys：是够获取自动增长的主键值。true表示获取。

2、keyProperty  ：指定将获取到的主键值封装到哪儿个属性里

# 15、使用MyBatis的代理开发有哪些要求？

使用Mapper代理方式，必须满足以下要求：

1、定义与SQL映射文件同名的Mapper接口，并且将Mapper接口和SQL映射文件放置在同一目录下。

2、设置SQL映射文件的namespace属性为Mapper接口全限定名

3、在 Mapper 接口中定义方法，方法名就是SQL映射文件中sql语句的id属性值，并保持参数类型和返回值类型一致