



## 1. #{}和\${}的区别是什么?

#{}是预编译处理，\${}是字符串替换。

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

Mybatis 在处理\${}时，就是把\${}替换成变量的值。

使用#{}可以有效的防止 SQL 注入，提高系统安全性。

## 2. 通常一个 Xml 映射文件，都会写一个 Dao 接口与之对应，请问，这个 Dao 接口的工作原理是什么？ Dao 接口里的方法，参数不同时，方法能重载吗？

Dao 接口，就是人们常说的 Mapper 接口，接口的全限定名，就是映射文件中的 namespace 的值，接口的方法名，就是映射文件中 MappedStatement 的 id 值，接口方法内的参数，就是传递给 sql 的参数。Mapper 接口是没有实现类的，当调用接口方法时，接口全限定名+方法名拼接字符串作为 key 值，可唯一定位一个 MappedStatement，举例：

com.mybatis3.mappers.StudentDao.findStudentById，可以唯一找到 namespace 为 com.mybatis3.mappers.StudentDao 下面 id = findStudentById 的

MappedStatement。在 Mybatis 中，每一个<select>、<insert>、<update>、<delete> 标签，都会被解析为一个 MappedStatement 对象。

Dao 接口里的方法，是不能重载的，因为是全限定名+方法名的保存和寻找策略。

Dao 接口的工作原理是 JDK 动态代理，Mybatis 运行时会使用 JDK 动态代理为 Dao 接口生成代理 proxy 对象，代理对象 proxy 会拦截接口方法，转而执行 MappedStatement 所代表的 sql，然后将 sql 执行结果返回。

## 3. Mybatis 是如何进行分页的？分页插件的原理是什么？

Mybatis 使用 RowBounds 对象进行分页，它是针对 ResultSet 结果集执行的内存分页，而非物理分页，可以在 sql 内直接书写带有物理分页的参数来完成物理分页功能，也可以使用分页插件来完成物理分页。

分页插件的基本原理是使用 Mybatis 提供的插件接口，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。

## 4. 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？

第一种是使用<resultMap>标签，逐一定义列名和对象属性名之间的映射关系。第二种是使用 sql 列的别名功能，将列别名书写为对象属性名，比如 T\_NAME AS NAME，

对象属性名一般是 name，小写，但是列名不区分大小写，Mybatis 会忽略列名大小写，智能找到与之对应对象属性名，你甚至可以写成 T\_NAME AS NaMe，Mybatis 一样可以正常工作。

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。

## 5. Xml 映射文件中，除了常见的 select|insert|update|delete 标签之外，还有哪些标签？

注：这道题出自京东面试官。

还有很多其他的标签，加上动态 sql 的 9 个标签，trim|where|set|foreach|if|choose|when|otherwise|bind 等，其中为 sql 片段标签，通过标签引入 sql 片段，为不支持自增的主键生成策略标签。

## 6. 简述 Mybatis 的插件运行原理，以及如何编写一个插件

Mybatis 仅可以编写针对 ParameterHandler、ResultSetHandler、StatementHandler、Executor 这 4 种接口的插件，Mybatis 使用 JDK 的动态代理，为需要拦截的接口生成代理对象以实现接口方法拦截功能，每当执行这 4 种接口对象的方法时，就会进入拦截方法，具体就是 InvocationHandler 的 invoke() 方法，当然，只会拦截那些你指定需要拦截的方法。实现 Mybatis 的 Interceptor 接口并复写 intercept() 方法，然后在给插件编写注解，指定要拦截哪一个接口的哪些方法即可，记住，还需要在配置文件中配置你编写的插件。

## 7. 一级、二级缓存

1) 一级缓存：基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当 Session flush 或 close 之后，该 Session 中的所有 Cache 就将清空。

2) 二级缓存与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap 存储，不同在于其存储作用域为 Mapper(Namespace)，并且可自定义存储源，如 Ehcache。要开启二级缓存，你需要在你的 SQL 映射文件中添加一行：<cache/>

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 C/U/D 操作后，默认该作用域下所有 select 中的缓存将被 clear。

## 8. 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 lazyLoadingEnabled=true|false。

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 a.getB().getName()，拦截器 invoke() 方法发现 a.getB() 是 null 值，那么就会单独发送事先保存好的查询关联 B 对象的 sql，把 B 查询上来，然后调用 a.setB(b)，于是 a 的对象 b 属性就有值了，接着完成 a.getB().getName() 方法的调用。这就是延迟加载的基本原理。

## 9. Mybatis 映射文件中，如果 A 标签通过 include 引用了 B 标签的内容，请问，B 标签能否定义在 A 标签的后面，还是说必须定义在 A 标签的前面？

虽然 Mybatis 解析 Xml 映射文件是按照顺序解析的，但是，被引用的 B 标签依然可以定义在任何地方，Mybatis 都可以正确识别。

原理是，Mybatis 解析 A 标签，发现 A 标签引用了 B 标签，但是 B 标签尚未解析到，尚不存在，此时，Mybatis 会将 A 标签标记为未解析状态，然后继续解析余下的标签，包含 B 标签，待所有标签解析完毕，Mybatis 会重新解析那些被标记为未解析的标签，此时再解析 A 标签时，B 标签已经存在，A 标签也就可以正常解析完成了。

## 10. 简述 Mybatis 的 Xml 映射文件和 Mybatis 内部数据结构之间的映射关系？

Mybatis 将所有 Xml 配置信息都封装到 All-In-One 重量级对象 Configuration 内部。在 Xml 映射文件中，<parameterMap> 标签会被解析为 ParameterMap 对象，其每个子元素会被解析为 ParameterMapping 对象。<resultMap> 标签会被解析为 ResultMap 对象，其每个子元素会被解析为 ResultMapping 对象。每一个 <select>、<insert>、<update>、<delete> 标签均会被解析为 MappedStatement 对象，标签内的 sql 会被解析为 BoundSql 对象。