# 一、Mybatis动态sql是做什么的？都有哪些动态sql？简述一下动态sql的执行原理？

1.Mybatis动态sql是指配置在mapper.xml中的sql，mybatis可以根据sql中的占位符，以及各种标签动态拼凑成最终真正执行的静态sql。

2.比较常用的动态sql标签有

<foreach>循环遍历一个集合，在拼sql时，每次把需要的值加到sql中，可以指定seperator，open和close。

<If>判断，通常是如果一个值为空时，可能会导致sql异常，为了避免异常，使用if标签判断不为空时才拼接标签内的sql

<Set>在update时，为了避免update tb\_user set where这种sql，没有一个条件满足，应该不需要set了，但是还写上了set。

<Where>可以避免where and这种尴尬的sql

<Include>引用xml中一段sql

3.执行原理。

首先是mybatis初始化时，把每个动态sql封装到mappedStatement对象中的SqlSource属性，存到configuration类：在SQLSessionFactoryBuilder.build创建SQLSessionFactory时，XMLStatementBuilder的parseStatementNode会调用builderAssistant.addMappedStatement方法，在addMappedStatement方法中，把封装好的mappedstatement放到核心配置类中configuration.addMappedStatement(statement)。

然后是在调用时，先拿到之前存好的BoundSql，并解析执行。首先，传给sqlsession一个mappedstatementId，然后在核心配置类中获取对应的mappedstatement，然后executor通过调用statementHandler，statementHandler再调用parameterHandler以及resultSetHandler执行sql并封装返回结果。

# 二、Mybatis是否支持延迟加载？如果支持，它的实现原理是什么？

支持。

实现原理是动态代理。Mybatis在封装resultset时，会判断对象的属性是否是引用属性，如果是，就会为该属性的get方法创建代理方法，这样在执行get方法时，会执行代理方法，从数据库查询。

# 三、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。

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。

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内，供下一次使用。简言之，就是重复使用Statement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executeBatch()批处理。与JDBC批处理相同。

# 四、简述下Mybatis的一级、二级缓存（分别从存储结构、范围、失效场景。三个方面来作答）？

一级缓存的底层数据结构是HashMap。范围是SqlSession。当sqlSession执行了增删改操作后，不管有没有commit，session都会失效。

二级缓存的底层结构也是HashMap，作用范围是Mapper的namespace。当相同namespace的sqlsession执行close或者commit方法时，mybatis会把查询结果存到二级缓存里，其他相同namespace的sqlsession就可以直接拿到结果，不用查数据库。

二级缓存失效条件：

1. 如果sqlsession1没有执行commit或者close方法，则二级缓存不生效。
2. 相同namespace的sqlsession执行了增删改操作。

# 五、简述Mybatis的插件运行原理，以及如何编写一个插件？

在mybatis初始化的时候，会将所有的interceptor放到configuration中，在openSession时，会给sqlsession对象创建executor，在newExecutor时，会遍历Executor对应的interceptor，然后执行plugin方法为当前Executor对象创建代理对象，如果有多个interceptor，则后一个又会代理前一个，形成一个代理chain，在执行时，会依次执行代理对象，而不是原来的Executor本身了。其他三大对象StatementHandler，ParameterHandler，ResultSetHandler的插件，也都是在创建各个对象的时候，遍历interceptor并生成代理对象，本质上都是动态代理，不再赘述。

自定义拦截器首先要实现Interceptor接口，使用@Intercept注解指定要拦截的对象，以及方法。然后重写Interceptor接口的三个方法，在intercept方法里实现拦截逻辑，在plugin方法里生成包装对象，在setProperties中可以设置参数。然后在SqlMapConfig核心配置文件中，配置上自定义的interceptor就可以了。