**1.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？**

**Mybatis动态sql是**：在进行sql操作的时候**动态的根据属性值来拼接数据库执行的sql语句**，比如查询或变更操作，根据传入的属性值不同，动态拼接出不同的可执行sql；

**动态sql有：**

1、<if>:

if是为了判断传入的值是否符合某种规则，比如是否不为空；

2、<where>:

 where标签可以用来做动态拼接查询条件，当和if标签配合的时候，不用显示的声明类似where 1=1这种无用的条件；

3、<choose><when><otherwise>:

这是一组组合标签，他们的作用类似于 Java 中的 switch、case、default。只有一个条件生效，也就是只执行满足的条件 when，没有满足的条件就执行 otherwise，表示默认条件；

4、<foreach>:

foreach标签可以把传入的集合对象进行遍历，然后把每一项的内容作为参数传到sql语句中，里面涉及到 item(具体的每一个对象), index(序号), open(开始符), close(结束符), separator(分隔符)；

5、<include>:

include可以把大量重复的代码整理起来，当使用的时候直接include即可，减少重复代码的编写；

6、<set>:

适用于更新中，当匹配某个条件后，才会对该字段进行更新操作

7、<trim>:

是一个格式化标签，主要有4个参数:

    prefix(前缀)；

    prefixOverrides(去掉第一个标记)；

    suffix(后缀)；

    suffixOverrides(去掉最后一个标记)；

**动态sql的执行原理：**

第一部分：在启动加载解析xml配置文件的时候进行解析，根据关键标签封装成对应的handler处理对象，封装成sqlSource对象存在mappedStatement。

调用流程：

I、SqlSessionFactoryBuilder对builder对象的时候，调用XMLConfigBuilder解析sqlMapConfig.xml配置文件，在解析过程中使用到了私有的mapperElement(XNode parent)方法

II、上面方法中通过构建XMLMapperBuilder,获取到所有的配置mapper配置，

在调用private void configurationElement(XNode context)方法进行解析mapper.xml,通过void buildStatementFromContext(List<XNode> list, String requiredDatabaseId)方法解析mapper.xml内的每一个标签

III、循环中构建XMLStatementBuilder对象，调用parseStatementNode()方法来封装mappedStatment对象，

IIII、在过程中需要构建sqlSource对象，通过XMLLanguageDriver对象进行处理，在XMLLanguageDriver中构建解析动态标签对象XMLScriptBuilder

第二部分：在执行过程中获取sqlSource中获取bondSql对象时，执行相应的标签handler

调用查询执行到BaseExecutor的query方法时候会去getBoundSql并且将参数传进去，

在sqlSource接口DynamicSqlSource实现类中，调用getBoundSql方法执行过程共创建DynamicContext对象进行判定解析封装成SqlSource对象返回。

**2、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。

原理：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。

1. **Mybatis都有哪些Executor执行器？它们之间的区别是什么？**

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。

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。

默认是SimplExcutor，需要配置在创建SqlSession对象的时候指定执行器的类型即可。

1. **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）？**

1）一级缓存：

Mybatis的一级缓存是指SqlSession级别的，作用域是SqlSession，Mybatis默认开启一级缓存，在同一个SqlSession中，相同的Sql查询的时候，第一次查询的时候，就会从缓存中取，如果发现没有数据，那么就从数据库查询出来，并且缓存到HashMap中，如果下次还是相同的查询，就直接从缓存中查询，就不在去查询数据库，对应的就不在去执行SQL语句。当查询到的数据，进行增删改的操作的时候，缓存将会失效。在spring容器管理中每次查询都是创建一个新的sqlSession，所以在分布式环境中不会出现数据不一致的问题

2）二级缓存：

二级缓存是mapper级别的缓存，多个SqlSession去操作同一个mapper的sql语句，多个SqlSession可以共用二级缓存，二级缓存是跨SqlSession。第一次调用mapper下的sql 的时候去查询信息，查询到的信息会存放到该mapper对应的二级缓存区域，第二次调用namespace下的mapper映射文件中，相同的SQL去查询，回去对应的二级缓存内取结果，使用值需要开启cache标签，在select上添加useCache属性为true，在更新和删除时候需要手动开启flushCache刷新缓存。

**5.简述Mybatis的插件运行原理，以及如何编写一个插件？**

1.mybatis可以编写针对Executor、StatementHandler、ParameterHandler、ResultSetHandler四个接口的插件，mybatis使用JDK的动态代理为需要拦截的接口生成代理对象，然后实现接口的拦截方法，所以当执行需要拦截的接口方法时，会进入拦截方法（AOP面向切面编程的思想）

## 2.1）.编写Intercepror接口的实现类

## 2）.设置插件的签名，告诉mybatis拦截哪个对象的哪个方法

## 3）.最后将插件注册到全局配置文件中