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，可以让我们在 XML 映射文件内，以 XML 标签的形式编写动态 SQL，完成逻辑判断和动态拼接 SQL 的功能。

Mybatis 提供了 9 种动态 SQL 标签：<if/>、<choose/>、<when/>、<otherwise/>、<trim/>、<when/>、<set/>、<foreach/>、<bind/>。

其执行原理为，使用 OGNL 的表达式，从 SQL 参数对象中计算表达式的值,根据表达式的值动态拼接 SQL，以此来完成动态 SQL 的功能。

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。

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。

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

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。

作用范围：Executor 的这些特点，都严格限制在 SqlSession 生命周期范围内。

4、 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：

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

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？

插件原理：

在四大对象创建的时候

1、每个创建出来的对象不是直接返回的，而是 interceptorChain.pluginAll(parameterHandler)；

2、获取到所有的 Interceptor（拦截器）（插件需要实现的接口），调用 interceptor.plugin(target)，返回 target 包装后的对象；

3、插件机制，我们可以使用插件为目标对象创建一个代理对象；AOP（面向切面），我们的插件可以为四大对象创建出代理对象，代理对象就可以拦截到四大对象的每一个执行；

编写插件

1、创建插件类实现 interceptor 接口并且使用注解标注拦截对象与方法

2、在配置文件中写入 plugins 标签