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？

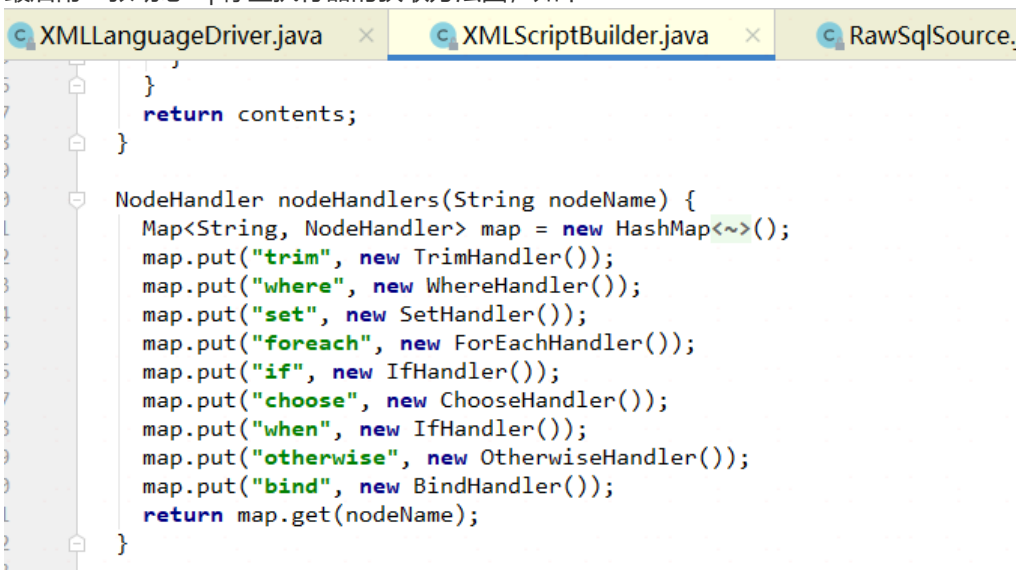
①所谓动态sql就是帮助我们简化对复杂sql语句的编写。个人理解可以大致分为两类：

一类：对字面上sql书写拼接的简化，如<where>,<set>,<trim>,<include>；

二类：比较方便的完成复杂参数的传递和一些逻辑运算等，如<if>,<choose>,<when>,<othwise>；

②关于动态sql的执行原理，就直接从XMLMapperBuilder说起了，大致流程如下：
XMLMapperBuilder.parse() -> parsePendingStatements() -> XMLStatementBuilder.parseStatementNode(langDriver.createSqlSource(configuration, context, parameterTypeClass)) -> XMLLanguageDriver.createSqlSource()->XMLScriptBuilder.parseScriptNode()
->parseDynamicTags()->nodeHandlers()
最终会到每个动态标签的解析器，封装到DynamicSqlSource或者RawSqlSource中，然后在SqlSource调用getBoundSql时候解析各自的动态标签内容，最终完成sql解析。

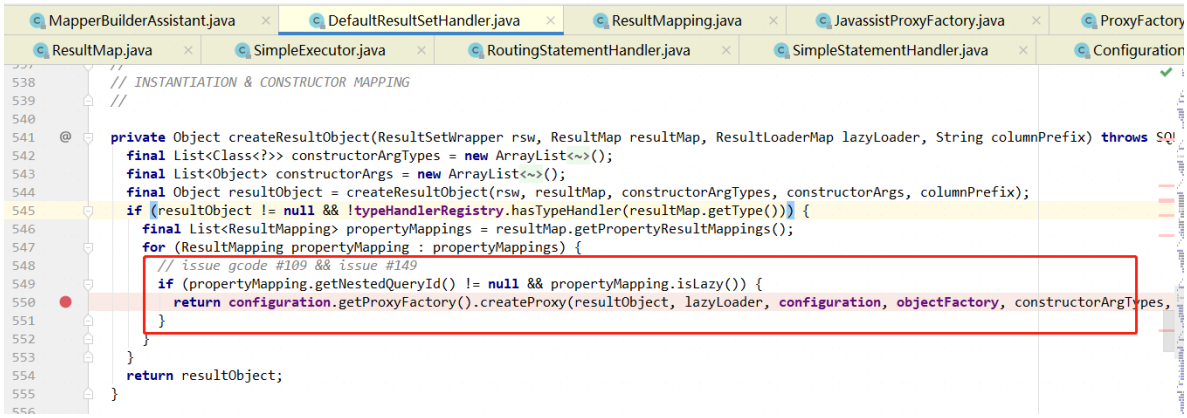
最后附一张动态sql标签执行器的获取方法图，如下：



2.Mybatis是否支持延迟加载？如果支持，它的实现原理是什么？

支持，原理大概是：ResultSetHandler在封装返回结果集的时候会判断每一个属性对应的列做判断，只要校验到有属性是懒加载，就会对整个结果集生成代理对象
（ps:后面代理逻辑源码没看太明白；还有就是调用getXxx时怎么实现延迟查询和属性回填的，也不太明白；setting.xml中的lazyLoadingEnabled 和 mapper.xml中的fetchType ，两个懒加载属性是怎么协调工作的?以上内容望老师可以讲解 ）
延迟加载源码跟踪大概如下：
SimpleExecutor.doQuery() -> RoutingStatementHandler.query() -> SimpleStatementHandler.query() -> DefaultResultSetHandler.handleResultSets() -> handleResultSet() -> handleRowValues() -> handleRowValuesForSimpleResultMap() -> getRowValue() -> createResultObject ()

懒加载校验和生成代理的源码如下：



3.Mybatis都有哪些Executor执行器？它们之间的区别是什么？

Executor作为接口定义执行器的基本功能方法，BaseExecutor是接口实现的抽象类，完成了通用功能的实现和具体策略方法的定义，同时起到适配器的作用，供下面具体的实现类完成，具体执行器有：SimpleExecutor，ReuseExecutor、BatchExecutor、以及CachingExecutor（这个和二级缓存有关），下面分别简单介绍：

BaseExecutor：作为一个上层抽象类，个别方法单独列出，对下面具体执行器能有好的说明。

①doFlushStatements：刷新声明列表，被调用的地方有(commit|rollback)，同时在(BatchExecutor的doQuery开头)也有调用；

SimpleExecutor：作为Configuration类中定义的默认实现执行器，功能实现比较简单。

①doUpdate和doQuery方法，获取连接 -> 获取声明 -> sql执行 -> 关闭声明；

②doFlushStatements就是空实现了

ReuseExecutor：相比较 SimpleExecutor，它内部维护了一个HashMap，用于存放<sql, Statement>的映射关系。

①在获取声明的时候就会先去map中拿，有就返回，没有就获取连接和声明同时存入map中；

②doFlushStatements就会遍历map关闭声明，同时清空map集合；

③这种方式能够减少一个sqlSession中相同sql语句的获取连接声明（当然前提没有commit|rollback），同时如果没有触发doFlushStatements可能会造成暂时的内存泄漏和资源浪费吧；

BatchExecutor：字面意思上是批量执行器，内部维护了Statement列表和BatchResult列表。

①doQuery方法：正常执行流程，无批量处理逻辑；

②doUpdate：只看的明白对(Statement列表和BatchResult列表)操作逻辑，以及doFlushStatements中大概对两个列表遍历执行的操作； ***(ps:难道真正的批处理在doFlushStatements时候才会执行？还是handler.batch呢？没搞太明白)***

③doFlushStatements： ***(ps:这个逻辑也有些小复杂，老师讲一下呗)***

CachingExecutor：这个执行器仅和二级缓存相关，具体的sql执行仍有sqlSession初始化时指定的BaseExecutor子类执行。

①query：会先从二级缓存中获取数据，如果有直接返回；如果没有，会调用原始执行器delegate的query方法，也就是BaseExecutor，在走一遍query方法，同时追加二级缓存

②其他方法：大都是调用其内部维护的BaseExecutor的同名方法执行

4.简述下Mybatis的一级、二级缓存（分别从存储结构、范围、失效场景三个方明作答）？

一级缓存：

存储结构：BaseExecutor.PerpetualCache.cache，一个hashMap

范围：统一sqlSession下可用

失效场景：update|commet|rollback|isFlushCacheRequired

二级缓存 ***(ps:这个老师没讲具体实现原理，好像还挺牛逼挺复杂的，我没搞太懂，强烈建议老师讲一下)***：

存储结构：TransactionalCacheManager中维护一个Map<Cache, TransactionalCache>，但是这个map好像只是能够获取到二级缓存内容，貌似真正存储在Configuration.caches中，源码跟踪到MapperBuilderAssistant.useNewCache

范围：统一namespace下可用

失效场景：MappedStatement.flushCacheRequired=true

5.简述Mybatis的插件运行原理，以及如何编写一个插件?

如何编写一个插件：

①自定义类实现Interceptor接口，实现代理方法plugin和增强方法intercept；
②注解标注拦截的类名、方法、参数；
③在sqlMapConfig中引入自定义插件类

插件运行原理：

①在自定义拦截器实现plugin方法时，需要调用Plugin.wrap()方法生成源目标对象的动态代理对象，而这个代理对象的执行器是InvocationHandler的实现类Plugin；同时(目标对象|拦截器|拦截的方法列表)会作为参数传递到Plugin对象的成员变量中，执行invoke方法时会用到；
②Plugin.invoke方法在执行时会校验目标对象的方法是否在拦截的方法列表中，如果不在，目标方法自己执行，然后返回；
③如果在拦截的方法列表中，会执行拦截器的intercept方法，而入参是Invocation对象，这个对象中包含（目标对象|方法|参数），可以直接调用这个对象的proceed方法执行目标方法；
④Configuration中维护着interceptorChain，保存着sqlMapConfig.xml中所有的拦截器；在生成四大核心对象时会调用interceptorChain.pluginAll()方法把所有的拦截器链路生成核心对象的代理对象，执行时会重复上面①②③