作业 1

崔凯铭

18636995694

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？***

答：动态 sql 是指依据传递的参数不同，查询条件需要动态变化的 sql。其中包括关键字 where 后使用 if 完成多个条件自动拼接，以及使用 foreach 完成 sql 循环拼接。使用<where>标签以及使用<if>标签，if 标签中 test 属性判断是否并且该 if 中的 sql，若为 true 则拼接，若为 false 则不拼接该 sql 转而判断下一个 if 标签。foreach 标签主要有 collection 属性指明 list 或者 array，item 属性代表我们迭代时的每一个元素，open 和 close 属性指明开端以及结尾，separator 属性指明每一个元素之间的分隔符。动态 sql 就是依照参数，条件判断，拼接 sql 语句。

2. ***Mybatis 是否支持延迟加载？如果支持，它的实现原理是什么？***

答：Mybatis 只支持 association 以及 collection 所关联的对象的延迟加载。原理是使用 CGLIB 产生目标对象的代理对象，当调用目标 get 方法时，进入拦截器方法，再发现目标对象所需的关联对象为 null 时，会发出事先写好的 sql 查询关联的对象并调用 set 方法将查询出的关联对象赋予目标对象的引用，于是目标对象此刻才拥有了关联对象。

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

答：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:重用 statement 的执行器，执行 update 或 select 时，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。

BatchExecutor：执行 update 时，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

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）？***

答：Mybatis 的一级缓存是 sqlseesion 级别的，默认开启。其存储结构是一个 hashMap，key 值由 statementid，参数，boundsql 中的 sql 一级 rowbound 中的页信息 hash 运算后得到。同一个 sqlsession 内的方法执行可以使用其缓存，不同 sqlsession 具有不同的缓存，相互间独立。在执行了增删改事务操作后，会对该 sqlsession 的一级缓存清空以此确保不会有脏数据。当然，我们可以手动调用 sqlSession.clearCache()或者直接 close()来清空缓存。

Mybatis 的二级缓存是 mapper 级别的，需要手动开启。开启方法是，在 sqlMapConfig.xml 文件中<setting name="cacheEnabled" value="true”/>，之后可以在 mapper.xml 文件中用<cache>标签进行配置或者使用注解方式 @CacheNamespace(implementation =)。存储结构默认的 PerpetualCache 使用的依旧是 hashmap 结构，但是这样无法实现分布式的缓存支持。所以我们可以实现 mybatis 中的 Cache 接口来自定义 cache 组件或者使用 mybaits-redis 中提供的 redisCache 来利用 redis 实现分布式缓存。对于二级缓存，如果多个 sqlSession 在这个 mapper 上操作，那么这几个 sqlSession 共享此 mapper 的缓存。如果其中任意一个 sqlSession 执行了增删改并事务提交的相关的操作，那么 mapper 缓存中的数据会失效，清空，以此保证数据的可靠性。

5. 简述 Mybatis 的插件运行原理，以及如何编写一个插件？

答：Mybaits 的插件原理主要是利用了 JDK 动态代理，对 Mybatis 的四大核心组件(Executor，statementHandler，parameterHandler 以及 ResultSetHandler)产生原生对象的代理对象。当进行方法调用时就会触发 invoke 方法来对原生方法的功能进行灵活增强。Mybatis 的插件核心接口时 Interceptor 接口，编写一个插件的第一步是实现此接口，此接口包含三个方法：intercept()方法是插件的核心方法用于实现增强；plugin()方法用于产生代理对象并返回该代理对象，此方法会在 InterceptorChain.pluginAll()方法调用后迭代每一个 interceptor 时被调用，每一个 interceptor 的 plugin 方法都会被调用用来产生代理对象；setProperties()方法用于读取配置文件中的插件的参数。在插件类上，我们还应该使用@Intercepts 接口以及@Signature 接口来表面该 interceptor 会对哪个核心组件的哪个方法进行拦截增强。最后，我们还要在 sqlMapConfig.xml 文件中在<plugins>标签中用<plugin>标签声明我们的 interceptor。