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。

1. ***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。

1. ***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

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）？***

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。

1. ***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。