▶**Mybatis动态sql**

◇**做什么**

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。

◇**哪些**

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

◇**执行原理**

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。

▶**Mybatis原理**

•加载mybatis配置文件、数据源、mapper映射文件，解析配置文件，MyBatis基于XML配置文件生成 Configuration，和一个个MappedStatement，其对应着<select | update | delete | insert>标签 项。

•SqlSessionFactoryBuilder通过Configuration对象生成SqlSessionFactory，用来开启SqlSession。

•SqlSession（getMapper）拿到Mapper对象的MapperProxy

•MapperProxy->SqlSession->Executor(CachingExecutor)->CRUD 借助MappedStatement中的结果映射关 系，将返回结果转化成HashMap、JavaBean等存储结构并返回。

▶**Mybatis延迟加载**

◇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。

◇实现原理：动态代理，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。

▶**Mybatis都有哪些Executor执行器**

•**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。

▶**Mybatis缓存**

◇**一级缓存**

•**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。

•**一级缓存的问题**

一级缓存是本地内存，不适用分布式环境下。

分布式环境下在使用事务的过程中可能存在脏读。

•**如何将一级缓存都关闭**

○所有sql你可以用插件去拦截拼接这种方式where 随机=随机

○分布式环境可以设置一级缓存为STATEMENT级别

<setting name="localCacheScope" value="SESSION"/>//默认

○select标签中设置flushCache为true

◇**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，如果在相同的namespace下的mapper映射文件 中增删改，并且提交了失误，就会失效。

▶**Mybatis的插件**

◇运行原理

•Mybatis仅可以编写针对ParameterHandler、ResultSetHandler、StatementHandler、Executor这 4种接口的插件

•Mybatis使用JDK的动态代理，为需要拦截的接口生成代理对象以实现接口方法拦截功能，每当执 行这4种接口对象的方法时，就会进入拦截方法，

•实现Mybatis的Interceptor接口并复写intercept()方法，然后在给插件编写注解，指定要拦截 哪一个接口的哪些方法即可，在配置文件中配置你编写的插件。

◇四大核心对象

•ParameterHandler：处理SQL的参数对象

•ResultSetHandler：处理SQL的返回结果集

•StatementHandler：数据库的处理对象，用于执行SQL语句

•Executor：MyBatis的执行器，用于执行增删改查操作

◇Mybatis插件接口-Interceptor

•Intercept方法，插件的核心方法

•plugin方法，生成target的代理对象

•setProperties方法，传递插件所需参数

▶**#{}和${}的区别**

1. #{}是预编译处理，${}是字符串替换。
2. Mybatis 在处理#{}时，会将 sql 中的#{}替换为?号，调用 PreparedStatement 的 set 方法来 赋值； 3）Mybatis 在处理${}时，就是把${}替换成变量的值。

4）使用#{}可以有效的防止 SQL 注入，提高系统安全性。

▶**JDBC的执行过程**

•加载驱动

•建立连接

•执行Sql语句（建立PreparedStatement对象 ）

•访问结果记录集ResultSet

•释放资源

▶**Mybatis 是如何进行分页的？分页插件的原理是什么？**

1. Mybatis 使用 RowBounds 对象进行分页，也可以直接编写 sql实现分页，也可以使用 Mybatis 的分 页插件。
2. 分页插件的原理：实现 Mybatis 提供的接口，实现自定义插件，在插件的拦截方法内拦 截待执行的 sql，然后重写 sql。

举例：select \* from student，拦截 sql 后重写为：select t.\* from （select \* from student） t limit 0，10

▶**简述 Mybatis 的 Xml 映射文件和 Mybatis 内部数据结构之间的映射关系？**

Mybatis 将所有 Xml 配置信息都封装到 All-In-One 重量级对象 Configuration 内部。在 Xml 映射文件中，标签会被解析为 ParameterMap 对象，其每个子元素会被解析 为 ParameterMapping 对象。标签会被解析为 ResultMap 对象，其每个子元素会被解析为 ResultMapping 对象。每一个<select>、<insert>、<update>、<delete>标签均会被解析为 MappedStatement 对象，标签内的 sql 会被解析为 BoundSql 对象。

▶**MyBatis 实现一对一有几种方式?具体怎么操作的？**

有联合查询和嵌套查询,联合查询是几个表联合查询,只查询一次,通过在resultMap里面配置 association节点配置一对一的类就可以完成;嵌套查询是先查一个表,根据这个表里面的 结果的外键 id,去再另外一个表里面查询数据,也是通过 association 配置,但另外一个表的查询 通过 select 属性配置。

▶**如何获取自动生成的(主)键值？**

配置文件设置 usegeneratedkeys 为 true

▶**resultType resultMap 的区别？**

1）类的名字和数据库相同时，可以直接设置 resultType 参数为 Pojo 类

2）若不同，需要设置 resultMap 将结果名字和 Pojo 名字进行转换

▶**Mybatis 比 IBatis 比较大的几个改进是什么？**

1）有接口绑定,包括注解绑定 sql 和 xml 绑定 Sql

2）动态sql由原来的节点配置变成OGNL表达式

3）在一对一,一对多的时候引进了association, 在一对多的时候引入了 collection 节点,不过都是在 resultMap 里面配置