**中软国际考题-笔试**

1. 单选题
2. MyBatis 中,不是Annotation注解的是（C）。

A. Result

B. Options

C. NumberFormat

D. Many

1. MyBatis 中如果SqlSession执行下面哪个操作并提交数据库不会情况一级缓存（D ）

A.insert

B.delete

C.update

D.select

1. MyBatis 中，使用Many注解必须指定什么属性，表示已映射的SQL语句的完全限定名(D)

A．insert

B. update

C. deleteProveider

D. select

4、下列关于Mybatis的使用场景描述错误的是（C）

A：当希望对象的持久化对应用程序完全透明时不适合使用Mybatis

B：当数据库有移植需求或需要支持多种数据库时不适合使用Mybatis

C：当不需要太多复杂功能，且对性能要求很高是不适合使用Mybatis

D：当应用程序需要完全动态的sql不适合使用Mybatis

5、MyBatis指定配置文件的根元素使用的是（B）。

A．<sqlMapConfig>

B．<configuration>

C．<setting>

D．<environments>

6、#{}和${}的区别是什么？（B）

A. 没有区别一样用 B. #{}是预编译处理，${}是字符串替换。

C. #{}是字符串替换 D. ${}是预编译处理

7、请选出Mybatis框架的优点 （C）

A. sql依赖于数据库，导致数据库移植性差。

B. 编写动态sql时,不方便调试，尤其逻辑复杂时。

C. sql写在xml里，便于统一管理和优化

D. 不支持级联更新、级联删除

8、下面哪个类是Mybatis sql处理的核心类？（C）

A．SqlFactory；B．SqlCreator；C．SqlSession；D．SessionCreator；

9、在MyBatis的Mapper文件中，以下选项配置错误的是（D）

1. <select id=”getById” resultType=”map” parameterType=”int”>

select \* from student

</select>

1. <delete id=”delById” parameterType=”int”>

delete from student where id=#{id}

</delete>

1. <update id=”update” parameterType=”com.pojo.Student”>

update student set name=#{name},age=#{age} where id=#{id}

</update>

1. <insert id=”addStudent” resultType=”com.pojo.Student”>

insert into student(name,age) values(#{name},#{age})

</insert>

10、在一个MyBatis的应用中，关于SlqessionFactoyfBuilder，SqlsessionFactory以及Sqlsession的应用范围和生命周期，以下说法错误的是（D）

A. SqlSessionFactoryBuilder的最佳范围为方法范围,即可以定义为本地方法变量

B. SqlSessionFactory的最佳范围是应用范围

C. SqlSession的最佳范围是方法范围或请求范围

D. SqlSession执行CRUD操作,不同线程可以共享使用

11. 以下关于MyBatis的使用场景描述错误的是（C）

A. 当希望对象持久化对应用程序完全透明时不适合使用mybatis

B. 当数据库有移植需求或需要支持多种数据库时不适合使用mybatis

C. 当不需要太多复杂功能,且对性能要求很高时不适合使用mybatis

D. 当应用程序需要完全动态的sql时不适合使用mybatis

12. 在MaBatis中，以下关于缓存说法错误的选项是（C）

A. mybatis的缓存分为一级缓存和二级缓存

B. 二级缓存可以被所有的SqlSession共享

C. 全局的二级缓存开关在mybatis配置文件的settings标签中配置,默认为false

D. 在映射文件中默认是没有开启二级缓存的,开启二级缓存需要使用cache标签

13.在Mybatis中，下列关于ResultType说法错误的是：（C）

A.resultType表示返回值类型为:完整类名或别名,mybatis也允许使用基本的数据类型,包括String,int类型

B.resultType和resultMap的数据结构是一样的,都是Map结构

C.如果POJO的属性名与SQL语句查询出来的字段名不一致的话,也可使用resultType来进行结果的自动映射

D.resultType和resultMap不能同时使用

14、mysql默认接受sql的大小是（A ）

A）1M B）2M C）500字符 D）以上说法都不对

15、下面哪个标签不是mapper.xml中的（D）

A）Select B）foreach C）if D）switch

16、下面标签写法正确的是（A）

A）<if test=”a != null” B）<if test=”a is not null”

C）<if test=”a <> null”

15、MyBatis框架适用场合是什么（D）

A.MyBatis不专注于SQL本身，是一个足够灵活的DAO层解决方案。

B.对性能的要求不是很高，或者需求变化不多的项目。

C. SQL语句的编写工作量较大。

D. 对性能的要求很高，或者需求变化较多的项目，如互联网项目。

16．MyBatis对结果集是如何处理的（A）。

A. Mybatis自动将sql执行结果映射至java对象

B. 和JDBC处理结果集的方法一样。

C. 编写代码将数据库记录封装成pojo对象。

17、在MyBatis的Mapper文件中，以下选项配置错误的是（D）

A： <select id="queryUserById" resultType="hashmap" parameterType ="int">

<J[CDATA[SELECT \* FROM user u WHERE u.id=#{id}]]>

〈/select〉

B:〈delete id="deleteUserById" parameterType="int">

<J[CDATA[DELETE FROM user WHERE id=#{id}]]>

〈/delete〉

C： <update id="updateUser" parameterType="com.proj ect.model.User">

<J[CDATA[UPDATE user SET username=#{username}, passwd=#{passwd} WHERE id=#{id}]]>

〈/update〉

D:〈insert id="addUser" resultType="com.project.model.User">

<J[CDATA[INSERT INTO user(username, passwd, id)VALUES (#{username}, #{passwd}, #{id})]]>

</insert>

18、 关于MyBatis的使用打桩错误的是（ C ）。

A. 当希望对象的持久化对应用程序完全透明时不适合使用MyBatis

B. 当数据库有移植需求或需要支持多种数据库时不适合使用MyBatis

C. 当不需要太多复杂功能、且对性能要求很高时不适合使用MyBatis

D. 当应用程序需要完全动态的SQL时不适合使用MyBatis

19、 MyBatis的全局配置文件中，settings元素设置项的默认值为false的是（D）

A. cacheEnabled

B. lazyLoadingEnabled

C. autoMappingBehavior

D. useGeneratedKeys

1. 多选
2. MyBaits提供了一个SQL工具类org.apache.ibatis.jdbc.SQL此类常用方法有（ABC）

A. T SELECT (String columns)

B. T FROM(Sring table)

C. T INSERT\_INTO(String tableName)

D. T REMOVE\_\_INTO(String tableName)

2、在Mybatis中，关于<resultmap>和<collection>元素说法错误的是（B,C）

A. <collection>通过property属性指定实体类中集合属性的名字

B. <collection>通过javaType属性指定集合中每个元素的类型

C. <collection>通过javaType属性指定集合的类型

D. <collection>通过type属性指定集合的类型

3、在MyBatis中，ExecutorType的值包括（ABD）。

A．ExecutorType.SIMPLE

B．ExecutorType.BATCH

C．ExecutorType.EXECUTE

D．ExecutorType.REUSE

4、MyBatis实现一对一有几种方式？（A、B）

A. 联合查询 B. 嵌套查询

C. 组合查询

D. 独立查询 E.继承查询

5、关于Mybatis接口绑定，下列说法正确的是（B C）

1. 接口本身可以加注解绑定
2. 通过注解方式，在接口方法上加上@Select @Update等
3. Xml中配置SQL来绑定；
4. Xml中namespace为接口名即可，无需加包名;

6、在Mybatis中，关于<resultmap>和<collection>元素说法错误的是（B,D）

1. <collection>通过property属性指定实体类中集合属性的名字
2. <collection>通过javaType属性指定集合中每个元素的类型
3. <collection>通过javaType属性指定集合的类型
4. <collection>通过type属性指定集合的类型

7、Mybatis内置的日志工厂提供日志功能，具体的日志实现有以下几种方式：（ABCD ）（选择两项）

A） SJF4J

B） Apache Commons Logging

C） log4j

D） log4j2

8、mybatis中级联删除的做法正确的是（ AC )

A）传入集合用foreach删除

B）删除标签上加上cascading属性

C）利用数据库自带约束删除

D）只能用代码控制先删从表再删主表

9、{}和${}的区别是什么?(ABCD)

A. {}解析传递进来的参数数据

B. ${}对传递进来的参数原样拼接在SQL中

C.{}是预编译处理，${}是字符串替换。

D.用#{}可以有效的防止SQL注入，提高系统安全性

10. 什么是ORM?(ABCD)

A.ORM(Object/Relational Mapping)中文翻译为对象/关系型数据映射

B.它也可以理解为一种数据持久化技术。

C.其主要是把对象模型关系型数据库的表建立对应关系,并且提供了一个通过JavaBean对象去操作数据库表的机制。

D. 直接操作JavaBean对象就可以实现数据的存储,查询,更改和删除等操作

11. MyBatis XML配置都有哪些，从下面选项中选出?(ABCD)

A. properties

B.Settings

C. typeAliases

D. typeHandlers

12、在Mybatis中，关于<resultmap>和<collection>元素说法错误的是（BD）【选两项】

A. <collection>通过property属性指定实体类中集合属性的名字

B. <collection>通过javaType属性指定集合中每个元素的类型

C. <collection>通过javaType属性指定集合中的类型

D. <collection>通过type属性指定集合中的类型

1. 填空题
2. MyBatis的一级缓存是\_\_\_\_SqlSession\_\_\_\_\_\_\_级别的缓存。在操作数据库时需要构造 \_\_\_\_\_SqlSession\_\_\_\_\_\_对象，在对象中有一个\_\_\_\_HashMap\_\_\_\_\_\_\_用于存储缓存数据
3. Mybatis的缓存分为：\_\_\_\_\_\_\_\_\_\_ 和\_\_\_\_\_\_\_\_\_。 （ 一级缓存、二级缓存 ）
4. Mybatis中使用\_\_\_\_\_\_\_\_\_\_类执行批处理。( BatchExecutor )
5. MyBatis注解绑定就是在接口的方法上面\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_等注解里面包含Sql语句来绑定(@Select@Update)
6. MyBatis里面的动态Sql一般是通过 节点来实现,通过OGNL语法来实现,但是如果要写的完整,必须配合 ,trim节点, （if，where）
7. 在三层结构中，数据访问层承担的责任是\_\_\_\_\_\_\_\_\_\_ 。(数据的增删改查操作)
8. mybaitsMybatis中支持三种形式数据源的配置，分别为：：\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_。(UNPOOLED、POOLED和JNDI)
9. 在mybatis中设置默认类型转换需要实现\_\_\_\_\_\_\_\_\_\_接口（typehandle）
10. 在mybatis中如果不支持生成主键的数据库时，需要用到\_\_\_\_\_\_\_\_\_\_标签处理（SelectKey）
11. 在支持主键生成的数据库中，使用\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 属性设置主键并获取主键id。（useGeneratedKeys keyProperty）
12. Mybatis使用RowBounds对象进行分页。
13. Dao接口里的方法，是不能重载的，因为是全限名+方法名的保存和寻找策略。
14. 在映射文件中，开启二级缓存需要使用cache标签
15. 判断题
16. MyBatis是一个支持普通SQL查询、存储过程和高级映射的优秀持久岑框架（T）
17. 在一个MyBatis的应用中，关于SlqessionFactoyfBuilder，SqlsessionFactory以及Sqlsession的应用范围和生命周期，SqlSession执行CRUD操作，不同线程可以共享使用。（ T ）
18. Mybatis执行批量插入，能返回数据库主键列表。（ T ）
19. MyBatis一级缓存放在session里面,默认就有,二级缓存放在它的命名空间里,默认是打开的,（ T ）
20. Mybatis缓存分为一级缓存和二级缓存，一级缓存默认开启，二级缓存放在命名空间里（ T ）
21. Mybatis可以使用RowBounds对象进行分页（ T ）
22. 3、Mybatis仅可以编写针对ParameterHandler、ResultSetHandler、StatementHandler、Executor这4种接口的插件（T）
23. Mybatis是半自动化的对象关系映射。（ T）
24. Mybits中的Mapper接口是通过cglib动态代理的。（ F ）
25. Mybits中使用的是使用JAXP解析mapper.xml的.（F）
26. mybatis-plus中使用MybatisSqlSessionFactoryBean类替换sqlSessionFactory的。（T）
27. SqlSessionFactory本身就是单例模式，所以不需要再次封装单例模式获取SqlSessionFactory对象（F）
28. Mybatis因为自带连接池，所以可以不用在配置其他连接池（T）
29. MyBatis不需要程序员自己编写Sql语句。（ F ）
30. MyBatis半自动ORM映射工具。（T）
31. resultType和resultMap的数据结构是一样的。（ √ ）

五、简答题

# 1、Mybatis#{}和${}的区别是什么？

答案：${}是Properties文件中的变量占位符，它可以用于标签属性值和sql内部，属于静态文本替换，比如${driver}会被静态替换为com.mysql.jdbc.Driver。#{}是sql的参数占位符，Mybatis会将sql中的#{}替换为?号，在sql执行前会使用PreparedStatement的参数设置方法，按序给sql的?号占位符设置参数值，比如ps.setInt(0, parameterValue)，#{item.name}的取值方式为使用反射从参数对象中获取item对象的name属性值，相当于param.getItem().getName()。

2、#{}和${}的区别是什么？（5分）

答案：#{}是预编译处理，${}是字符串替换。

Mybatis在处理${}时，就是把${}替换成变量？号，调用PreparedStatement的set方法来赋值；Mybatis在处理${}时，就是把${}替换成变量的值。使用#{}可以有效的防止SQL注入，提高系统安全性。

3、Mybatis是如何将sql执行结果封装为目标对象并返回的？都有哪些映射形式？（5分）

答案：第一种是使用<resultMap>标签，逐一定义列名和对象属性名之间的映射关系。第二种是使用sql列的别名功能，将列名书写为对象属性名，比如T\_NAME AS NAME对象属性名一般是name，小写，但是列名不区分大小写，Mybatis会忽略大小写，智能找到与之对应对象属性名，你甚至可以写成T\_NAME AS NaMe，Mybatis一样可以正常工作。

有了列名与属性名的映射关系后，Mybatis通过反射创建对象，同时使用反射给对象的属性逐一赋值并返回，那些找不到映射关系的属性，是无法完成赋值的。

4、MyBatis分页插件的基础原理是什么？（5分）

答案：Mybatis中Statement语句是通过RoutingStatementHandler对象的prepare方法生成的。所以利用拦截器实现Mybatis分页的一个思路就是拦截StatementHandler接口的prepare方法，然后在拦截器方法中把Sql语句改成对应的分页查询Sql语句，之后再调用StatementHandler对象的prepare方法，即调用invocation.proceed()。

5、Xml映射文件中，除了常见的select|insert|updae|delete标签之外，还有哪些标签？

答：还有很多其他的标签，<resultMap>、<parameterMap>、<sql>、<include>、<selectKey>，加上动态sql的9个标签，trim|where|set|foreach|if|choose|when|otherwise|bind等，其中<sql>为sql片段标签，通过<include>标签引入sql片段，<selectKey>为不支持自增的主键生成策略标签。

6、最佳实践中，通常一个Xml映射文件，都会写一个Dao接口与之对应，请问，这个Dao接口的工作原理是什么？Dao接口里的方法，参数不同时，方法能重载吗？

答：Dao接口，就是人们常说的Mapper接口，接口的全限名，就是映射文件中的namespace的值，接口的方法名，就是映射文件中MappedStatement的id值，接口方法内的参数，就是传递给sql的参数。Mapper接口是没有实现类的，当调用接口方法时，接口全限名+方法名拼接字符串作为key值，可唯一定位一个MappedStatement，举例：com.mybatis3.mappers.StudentDao.findStudentById，可以唯一找到namespace为com.mybatis3.mappers.StudentDao下面id = findStudentById的MappedStatement。在Mybatis中，每一个<select>、<insert>、<update>、<delete>标签，都会被解析为一个MappedStatement对象。

Dao接口里的方法，是不能重载的，因为是全限名+方法名的保存和寻找策略。

Dao接口的工作原理是JDK动态代理，Mybatis运行时会使用JDK动态代理为Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回。

7、简述Mybatis的插件运行原理，以及如何编写一个插件。

答：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，具体就是InvocationHandler的invoke()方法，当然，只会拦截那些你指定需要拦截的方法。

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，记住，别忘了在配置文件中配置你编写的插件。

8、Mybatis动态sql是做什么的？都有哪些动态sql？能简述一下动态sql的执行原理不？

答：Mybatis动态sql可以让我们在Xml映射文件内，以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trim|where|set|foreach|if|choose|when|otherwise|bind。

其执行原理为，使用OGNL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。

9、Mybatis是否支持延迟加载？如果支持，它的实现原理是什么？

答：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。

当然了，不光是Mybatis，几乎所有的包括Hibernate，支持延迟加载的原理都是一样的。

10、Mybatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复？

答：不同的Xml映射文件，如果配置了namespace，那么id可以重复；如果没有配置namespace，那么id不能重复；毕竟namespace不是必须的，只是最佳实践而已。

原因就是namespace+id是作为Map<String, MappedStatement>的key使用的，如果没有namespace，就剩下id，那么，id重复会导致数据互相覆盖。有了namespace，自然id就可以重复，namespace不同，namespace+id自然也就不同。

11、Mybatis都有哪些Executor执行器？它们之间的区别是什么？

答：Mybatis有三种基本的Executor执行器，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<String, Statement>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。

12、Mybatis中如何指定使用哪一种Executor执行器？

答：在Mybatis配置文件中，可以指定默认的ExecutorType执行器类型，也可以手动给DefaultSqlSessionFactory的创建SqlSession的方法传递ExecutorType类型参数。

13、**MyBatis有什么功能？它和JDBC有什么区别？**

答案：

MyBatis是一个数据持久层(ORM)框架。把实体类和SQL语句之间建立了映射关系，是一种半自动化的ORM实现，MyBatis的优点：

基于SQL语法，简单易学

能了解底层组装过程

SQL语句封装在配置文件中，便于统一管理与维护，降低了程序的耦合度

程序调试方便

MyBatis和JDBC的区别：

减少了大量的代码量

最简单的持久化框架

架构级性能增强

SQL代码从程序代码中彻底分离，可重用

增强了项目中的分工

增强了移植性

**14. MyBatis核心配置文件包含什么内容？**

答案：

configuration.xml（文件名可以自定义）是系统的核心配置文件，包含数据源和事务管理器等设置和属性信息，XML文档结构如下：

configuration 配置

properties 可以配置在Java 属性配置文件中

settings 修改 MyBatis 在运行时的行为方式

typeAliases 为 Java 类型命名一个短的名字

typeHandlers 类型处理器

objectFactory 对象工厂

plugins 插件

environments 环境（MyBatis的基础配置内容，包含了JDBC数据源相关参数等）

environment 环境变量

transactionManager 事务管理器

dataSource 数据源

mappers映射器

15、resultMap的作用是什么？

答案：

resultMap元素是MyBatis映射中最重要最强大的元素。它在一些情形下允许完成一些JDBC不支持的功能，如果不提供resultMap，则语句执行结果可以自动被转换为基本数据类型、Map和对应的实体类对象（约定的映射依据是结果集中的列名映射为实体类中的属性名）

ResultMap的设计目标就是为复杂语句通过配置文件描述数据库表结果和Java实体类的关系（OR映射）

16、（2）最佳实践中，通常一个Xml映射文件，都会写一个Dao接口与之对应，请问，这个Dao接口的工作原理是什么？Dao接口里的方法，参数不同时，方法能重载吗？

Dao接口，就是人们常说的Mapper接口，接口的全限名，就是映射文件中的namespace的值，接口的方法名，就是映射文件中MappedStatement的id值，接口方法内的参数，就是传递给sql的参数。Mapper接口是没有实现类的，当调用接口方法时，接口全限名+方法名拼接字符串作为key值，可唯一定位一个MappedStatement，举例：com.mybatis3.mappers.StudentDao.findStudentById，可以唯一找到namespace为com.mybatis3.mappers.StudentDao下面id = findStudentById的MappedStatement。在Mybatis中，每一个<select>、<insert>、<update>、<delete>标签，都会被解析为一个MappedStatement对象。

Dao接口里的方法，是不能重载的，因为是全限名+方法名的保存和寻找策略。

Dao接口的工作原理是JDK动态代理，Mybatis运行时会使用JDK动态代理为Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回。

17、简述Mybatis的插件运行原理，ParameterHandler、ResultSetHandler、StatementHandler、Executor这4种接口的插件运行方式。

Mybatis仅可以编写针对，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，具体就是InvocationHandler的invoke()方法，当然，只会拦截那些你指定需要拦截的方法。

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，记住，别忘了在配置文件中配置你编写的插件。

4）fetchSize的作用

设置获取结果集的数量。

18、请写出Mybatis框架的优缺点?。(分别写出2条即可)（5分）

答案：

优点：

1. 易于上手和掌握。

2. sql写在xml里，便于统一管理和优化。

3. 解除sql与程序代码的耦合。

4. 提供映射标签，支持对象与数据库的orm字段关系映射

5. 提供对象关系映射标签，支持对象关系组建维护

6. 提供xml标签，支持编写动态sql。

缺点：

1. sql工作量很大，尤其是字段多、关联表多时，更是如此。

2. sql依赖于数据库，导致数据库移植性差。

3. 由于xml里标签id必须唯一，导致DAO中方法不支持方法重载。

4. 字段映射标签和对象关系映射标签仅仅是对映射关系的描述，具体实现仍然依赖于sql。（比如配置了一对多Collection标签，如果sql里没有join子表或查询子表的话，查询后返回的对象是不具备对象关系的，即Collection的对象为null）

5. DAO层过于简单，对象组装的工作量较大。

6.  不支持级联更新、级联删除。

7. 编写动态sql时,不方便调试，尤其逻辑复杂时。

8 提供的写动态sql的xml标签功能简单（连struts都比不上），编写动态sql仍然受限，且可读性低。

9. 使用不当，容易导致N+1的sql性能问题。

10. 使用不当，关联查询时容易产生分页bug。

11. 若不查询主键字段，容易造成查询出的对象有“覆盖”现象。

12. 参数的数据类型支持不完善。（如参数为Date类型时，容易报没有get、set方法，需在参数上加@param）

13. 多参数时，使用不方便，功能不够强大。（目前支持的方法有map、对象、注解@param）

14. 缓存使用不当，容易产生脏数据。

**19、Mybatis动态sql是做什么的？都有哪些动态sql？能简述一下动态sql的执行原理不？**

Mybatis动态sql可以让我们在Xml映射文件内，以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：trim|where|set|foreach|if|choose|when|otherwise|bind。

其执行原理为，使用OGNL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

20、MyBatis里面的动态Sql是怎么设定的?用什么语法?

答案：MyBatis里面的动态Sql一般是通过if节点来实现,通过OGNL语法来实现,但是如果要写的完整,必须配合where,trim节点,where节点是判断包含节点有内容就插入where,否则不插入,trim节点是用来判断如果动态语句是以and 或or开始,那么会自动把这个and或者or取掉。

六、程序实现题

1、有一张wf\_org表，编写Mapper接口名字为OrgMapper，有方法int selectOrgAllNum，参数为orgFatherId、orgName；要求只只能使用纯注解的方式进行查询操作，查询结果是count(\*)；查询条件是字段orgfatherid等于参数orgfatherid；字段orgdelete等于0；如果参数orgName不为空，根据orgname字段进行模糊查询，否则字段orgname不作为查询条件

|  |
| --- |
| **public** **interface** OrgMapper {  @Select("<script>"  + "SELECT count(\*) FROM wf\_org WHERE orgfatherid=#{orgFatherId}"  + "<if test='orgName!=null and orgName != \"\" '>"  +" AND orgname LIKE CONCAT('%',#{orgName},'%') "  + "</if>"  +"AND orgdelete=0"  + "</script>")  **int** selectOrgAllNum(@Param("orgFatherId") String orgFatherId,@Param("orgName") String orgName);  } |

2、使用SSM技术来实现学生管理系统的以下功能：（10分）

（1）添加学生信息

（2）查询所有学生信息

（3）删除某个学生信息

|  |
| --- |
| 配置文件  <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <!-- 别名 -->  <typeAliases >  <package name="cn.happy.entity"/>  </typeAliases>    <environments default="development">  <environment id="development">  <!-- 使用jdbc的事务 -->  <transactionManager type="JDBC" />  <!-- 使用自带的连接池 -->  <dataSource type="POOLED">  <property name="driver" value="oracle.jdbc.OracleDriver" />  <property name="url" value="jdbc:oracle:thin:@localhost:1521:orcl" />  <property name="username" value="happy" />  <property name="password" value="1" />  </dataSource>  </environment>  </environments>  <mappers>  <!-- 连接小配置 -->  <mapper resource="cn/happy/dao/StudentDAO.xml" />  </mappers>  </configuration>  Student.java  public class Student {  private int stuno;  private String stuname;  private int stuage;  private Date studate;  public String toString() {  return "Student [stuno=" + stuno + ", stuname=" + stuname + ", stuage="  + stuage + ", studate=" + studate + "]";  }  public int getStuno() {  return stuno;  }  public void setStuno(int stuno) {  this.stuno = stuno;  }  public String getStuname() {  return stuname;  }  public void setStuname(String stuname) {  this.stuname = stuname;  }  public int getStuage() {  return stuage;  }  public void setStuage(int stuage) {  this.stuage = stuage;  }  public Date getStudate() {  return studate;  }  public void setStudate(Date studate) {  this.studate = studate;  }    }  IStudentDAO  public interface IStudentDAO {  /\*  \* 添加学生信息  \*/  public int addStu(Student stu) throws IOException;    /\*  \* 查询所有  \*/  public List<Student> findAll() throws IOException;    /\*  \* 删除  \*/  public int delStudent(int id) throws IOException;  }  IStudentDAOImpl  public class IStudentDAOImpl implements IStudentDAO{      SqlSession session;  /\*  \* 添加  \*/  public int addStu(Student stu) throws IOException {  //获取session  session = MybatisUtil.getSession();    //添加insert  int result = session.insert("insertStudent", stu);    //添加事物  session.commit();    //关闭session  session.close();  return result;  }  /\*  \* 查询所有  \*/  public List<Student> findAll() throws IOException {  //获取session  session = MybatisUtil.getSession();  List<Student> list = session.selectList("findAll");  //关闭session  session.close();  return list;  }  /\*  \* 删除  \*/  public int delStudent(int id) throws IOException {  //获取session  session = MybatisUtil.getSession();  int result = session.delete("delStudent", id);  session.commit();  return result;  }  }  StudentDAO.xml  <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="cn.happy.dao">  <!-- 添加 -->  <insert id="insertStudent">  insert into student(stuno,stuname,stuage,studate)  values(ssm.nextval,#{stuname},#{stuage},#{studate})  <selectKey keyProperty="stuno" resultType="int">  select SSM.CURRVAL from dual  </selectKey>  </insert>  <!-- 查询所有 -->  <select id="findAll" resultType="Student">  select \* from student  </select>  <!--删除学生 -->  <delete id="delStudent">  delete from student where stuno=#{id}  </delete>  </mapper>  MybatisUtil.java  public class MybatisUtil {  private static String config="mybatis-config.xml";  static Reader reader;  static{  try {  reader= Resources.getResourceAsReader(config);  } catch (IOException e) {  e.printStackTrace();  }  }  private static SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(reader);  //提供一个可以获取到session的方法  public static SqlSession getSession() throws IOException{    SqlSession session = factory.openSession();  System.out.println("3333");  return session;  }  } |

3、根据员工表编写规范的存储过程。完成以下任务：

1)．给指定员工涨工资，其中员工编号和涨的工资金额由输入参数决定

2)．涨工资后,显示所有员工薪水情况

3)．在存储过程中考虑到以下异常：

A．指定的员工不存在的情况

B．涨的工资金额小于等于0的情况

4)．编写调用程序，要求测试各种情况

（10分）

|  |
| --- |
| CREATE OR REPLACE PROCEDURE raise\_salary  (eno NUMBER, *--输入参数，雇员编号*  salary NUMBER, *--输入参数，雇员薪水*  on\_Flag OUT number, *--执行状态*  os\_Msg OUT VARCHAR2 *--提示信息*  ) IS  CURSOR emp\_cur IS  SELECT ename,sal FROM employee;  v\_emp emp\_cur%ROWTYPE;  exp1 EXCEPTION;  exp2 EXCEPTION;  BEGIN  IF salary<=0 THEN  RAISE exp1;  END IF;  UPDATE EMPLOYEE  SET sal=sal+salary  WHERE empno=eno;  IF SQL%NOTFOUND THEN  RAISE exp2;  END IF;  on\_Flag:=1;  os\_Msg:='修改成功';  OPEN emp\_cur;  LOOP  FETCH emp\_cur INTO v\_emp;  EXIT WHEN emp\_cur%NOTFOUND;  dbms\_output.put\_line(v\_emp.ename||v\_emp.sal);  END LOOP;  CLOSE emp\_cur;  EXCEPTION  WHEN exp1 THEN  on\_Flag:=-1;  os\_Msg:='涨的薪水不能小于等于0。';  WHEN exp2 THEN  on\_Flag:=-2;  os\_Msg:='不存在该员工。';  WHEN OTHERS THEN  on\_Flag:=SQLcode;  os\_Msg:=SQLERRM;  END;  *--调用1不存在该员工*  DECLARE  v\_empno employee.empno%TYPE:=1234;  v\_sal employee.sal%TYPE:=50;  on\_Flag number(1); *--执行状态*  os\_Msg VARCHAR2(200); *--提示信息*  BEGIN  raise\_salary(v\_empno,v\_sal,on\_Flag,os\_Msg);  IF on\_flag=1 THEN  dbms\_output.put\_line(os\_msg);  END IF;  IF on\_flag=-1 THEN  dbms\_output.put\_line(os\_msg);  END IF;  IF on\_flag=-2 THEN  dbms\_output.put\_line(os\_msg);  END IF;  END;  *--调用2涨的薪水不能小于等于0*  DECLARE  v\_empno employee.empno%TYPE:=7788;  v\_sal employee.sal%TYPE:=0;  on\_Flag number(1); *--执行状态*  os\_Msg VARCHAR2(200); *--提示信息*  BEGIN  raise\_salary(v\_empno,v\_sal,on\_Flag,os\_Msg);  IF on\_flag=1 THEN  dbms\_output.put\_line(os\_msg);  END IF;  IF on\_flag=-1 THEN  dbms\_output.put\_line(os\_msg);  END IF;  IF on\_flag=-2 THEN  dbms\_output.put\_line(os\_msg);  END IF;  END;  *--调用3修改成功*  DECLARE  v\_empno employee.empno%TYPE:=7788;  v\_sal employee.sal%TYPE:=50;  on\_Flag number(1); *--执行状态*  os\_Msg VARCHAR2(200); *--提示信息*  BEGIN  raise\_salary(v\_empno,v\_sal,on\_Flag,os\_Msg);  IF on\_flag=1 THEN  dbms\_output.put\_line(os\_msg);  END IF;  IF on\_flag=-1 THEN  dbms\_output.put\_line(os\_msg);  END IF;  IF on\_flag=-2 THEN  dbms\_output.put\_line(os\_msg);  END IF;  END; |

4、mybatis实现分页查询的功能

（10分）

|  |
| --- |
| public class Pagination<T> {    /\*\*  \* 总条数  \*/  private int totalCount;  /\*\*  \* 页面  \*/  private int pageNo;  /\*\*  \* 每页条数  \*/  private int pageSize;  /\*\*  \* 总页数  \*/  private int totalPage;    /\*\*  \* 列表  \*/  private List<T> list;    /\*\*  \* 默认每页条数  \*/  private final static int DEFAULT\_PAGESIZE = 20;    /\*\*  \*  \*/  public Pagination() {    }    /\*\*  \* 分页  \*  \* @param pageNo  \* @param pageSize  \* @param totalCount  \*/  public Pagination(int pageNo, int pageSize, int totalCount) {  if (pageNo <= 0) {  this.pageNo = 1;  } else {  this.pageNo = pageNo;  }    if (pageSize <= 0) {  this.pageSize = DEFAULT\_PAGESIZE;  } else {  this.pageSize = pageSize;  }    if (totalCount < 0) {  this.totalCount = 0;  } else {  this.totalCount = totalCount;  }    totalPage = (this.totalCount % this.pageSize == 0) ?  this.totalCount / this.pageSize :  this.totalCount / this.pageSize + 1;  }    /\*\*  \* 分页  \*  \* @param pageNo  \* @param pageSize  \* @param totalCount  \* @param list  \*/  public Pagination(int pageNo, int pageSize, int totalCount, List<T> list) {  this(pageNo, pageSize, totalCount);  this.list = list;  }    /\*\*  \* get totalCount  \*  \* @return  \*/  public int getTotalCount() {  return totalCount;  }    /\*\*  \* set totalCount  \*  \* @param totalCount  \*/  public void setTotalCount(int totalCount) {  this.totalCount = totalCount;  }    /\*\*  \* get pageNo  \*  \* @return  \*/  public int getPageNo() {  return pageNo;  }    /\*\*  \* set pageNo  \*  \* @param pageNo  \*/  public void setPageNo(int pageNo) {  this.pageNo = pageNo;  }    /\*\*  \* get pageSize  \*  \* @return  \*/  public int getPageSize() {  return pageSize;  }    /\*\*  \* set pageSize  \*  \* @param pageSize  \*/  public void setPageSize(int pageSize) {  this.pageSize = pageSize;  }    /\*\*  \* get totalPage  \*  \* @return  \*/  public int getTotalPage() {  return totalPage;  }    /\*\*  \* set totalPage  \*  \* @param totalPage  \*/  public void setTotalPage(int totalPage) {  this.totalPage = totalPage;  }    /\*\*  \* get list  \*  \* @return  \*/  public List<T> getList() {  return list;  }    /\*\*  \* set list  \*  \* @param list  \*/  public void setList(List<T> list) {  this.list = list;  }    /\*\*  \* 获取前一页  \*  \* @return  \*/  public int getPrevPage() {  int prevPage = 0;  if (pageNo <= 1) {  prevPage = 1;  } else {  prevPage = pageNo - 1;  }    return prevPage;  }    /\*\*  \* 获取下一页  \*  \* @return  \*/  public int getNextPage() {  int nextPage = 0;  if (pageNo < totalPage) {  nextPage = pageNo + 1;  } else {  nextPage = totalPage;  }  return nextPage;  }    /\*\*  \* 判断是否有下一页  \*  \* @return  \*/  public boolean hasNextPage() {  return pageNo < totalPage;  }    /\*\*  \* 列表是否为空  \*  \* @return  \*/  public boolean isEmptyForList() {  return list == null || list.isEmpty();  }  public class Request {    /\*\*  \* 分页大小,默认值20  \*/  private int pageSize = 20;    /\*\*  \* 分页起始，默认值 1  \*/  private int pageNo = 1;    }  //查询总数  <operation name="queryTableInfoCount" multiplicity="one">  <extraparams>  <param name="shopId" javatype="java.lang.String"/>  </extraparams>  <sql>  <![CDATA[  select count(\*) from desk  ]]>  </sql>  <sqlmap>  <![CDATA[  select count(\*) from desk  where shop\_id = #shopId#  ]]>  </sqlmap>  </operation>  //分页查询  <operation name="queryTableInfoListByPage" multiplicity="many">  <extraparams>  <param name="shopId" javatype="java.lang.String"/>  <param name="startRow" javatype="java.lang.Integer"/>  <param name="endRow" javatype="java.lang.Integer"/>  </extraparams>  <sql>  select id,  table\_name,  from desk  </sql>  <sqlmap>  <![CDATA[  select id,  table\_name  from kbdesk\_info  where shop\_id=#shopId#  limit #startRow#,#endRow#  ]]>  </sqlmap>  </operation>  分页查询  public Pagination<Table> queryTableListByShopId(  String shopId,int pageNo,int pageSize) {  int startRow = (pageNo - 1) \* pageSize;  int count = (int)queryTableInfoCountByShopId(shopId);  List<Table> tableInfoList = new ArrayList<>();  if (count > 0) {  List<TableInfo> tableInfoModelList =queryTableInfoListByPage(shopId, startRow, pageSize);  if (!CollectionUtils.isEmpty(tableInfoModelList)) {  tableInfoList = tableInfoModelList.stream().map((s) -> {  return convert(s);  }).collect(Collectors.toList());  }  }  Pagination<Table> answerPagination = new Pagination<>(pageNo, pageSize,  count, tableInfoList);    return answerPagination;  } |

5、用代码演示如何执行批量插入?

|  |
| --- |
| 首先,创建一个简单的insert语句:  <insert id="insertName">   insert into names (name) values (#{value})  </insert>  然后在Java代码中像下面这样执行批处理插入:  List<String> names = new ArrayList<String>();  names.add("Fred");  names.add("Barney");  names.add("Betty");  names.add("Wilma");    // 注意这里 ExecutorType.BATCH  SqlSession sqlSession = sqlSessionFactory.openSession(ExecutorType.BATCH);  try {  NameMapper mapper = sqlSession.getMapper(NameMapper.class);  for (String name : names) {  mapper.insertName(name);  }  sqlSession.commit();  } finally {  sqlSession.close();  } |

6、使用Maven,junit和Mybatis框架以及mysql实现人员表的person(字段有:id,name,age,birthday)的增删改查

|  |
| --- |
| pom.xml:  <dependencies>  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.2.3</version>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.6</version>  </dependency>  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  </dependency>  <dependency>  <groupId>com.github.pagehelper</groupId>  <artifactId>pagehelper</artifactId>  <version>5.0.0</version>  </dependency>  </dependencies> |
| mybatis.xml:  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <plugins>  <!-- com.github.pagehelper为PageHelper类所在包名 -->  <plugin interceptor=*"com.github.pagehelper.PageInterceptor"*>  <!-- 什么都不配，使用默认的配置 -->  </plugin>  </plugins>  <environments default=*"development"*>  <environment id=*"development"*>  <!-- 使用jdbc事务管理，事务控制由mybatis-->  <transactionManager type=*"JDBC"* />  <!-- 数据库连接池，由mybatis管理 -->  <dataSource type=*"POOLED"*>  <property name=*"driver"* value=*"com.mysql.jdbc.Driver"* />  <property name=*"url"* value=*"jdbc:mysql:///dd"* />  <property name=*"username"* value=*"root"* />  <property name=*"password"* value=*"123456"* />  </dataSource>  </environment>  </environments>  <mappers>  <mapper resource=*"com/po/Person.xml"*/>  </mappers>  </configuration> |
| Person 实体类:  **package** com.po;  **public** **class** Person {  **private** **int** id;  **private** String name;  **private** **int** age;  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** **int** getAge() {  **return** age;  }  **public** **void** setAge(**int** age) {  **this**.age = age;  }  **public** Person() {  **super**();  // **TODO** Auto-generated constructor stub  }  **public** Person(**int** id, String name, **int** age) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.age = age;  }  **public** Person(String name, **int** age) {  **super**();  **this**.name = name;  **this**.age = age;  }  @Override  **public** String toString() {  **return** "Person [id=" + id + ", name=" + name + ", age=" + age + "]";  }  } |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace=*"test"*>  <insert id=*"savePerson"* parameterType=*"com.po.Person"*>  insert into person(name,age) values(#{name},#{age})  </insert>  <select id=*"select"* parameterType=*"int"* resultType=*"com.po.Person"*>  select \* from person where id=#{id}  </select>  <select id=*"selectList"* resultType=*"com.po.Person"*>  select \* from person  </select>  <update id=*"update"* parameterType=*"com.po.Person"*>  update person set name=#{name},age=#{age} where id=#{id}  </update>  <delete id=*"delete"* parameterType=*"int"*>  delete from person where id=#{id}  </delete>  </mapper> |
| 单元测试类实现增删改查:  package com.test;  import java.io.IOException;  import java.util.HashMap;  import java.util.List;  import org.apache.ibatis.io.Resources;  import org.apache.ibatis.session.SqlSession;  import org.apache.ibatis.session.SqlSessionFactory;  import org.apache.ibatis.session.SqlSessionFactoryBuilder;  import org.junit.Test;  import com.github.pagehelper.PageHelper;  import com.github.pagehelper.PageInfo;  import com.po.Person;  public class PersonTest {  SqlSessionFactory factory=null;  {  try {  factory = new SqlSessionFactoryBuilder().build(Resources.getResourceAsReader("mybatis.xml"));  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  @Test  public void testAdd() {  SqlSession session = factory.openSession();  Person person = new Person("pjl",20);  session.insert("test.savePerson",person);  session.commit();  session.close();  }  @Test  public void testUpdate() {  SqlSession session = factory.openSession();  Person person = new Person(1,"pjl1",30);  session.update("test.update",person);  session.commit();  session.close();  }  @Test  public void testDel() {  SqlSession session = factory.openSession();  session.delete("test.delete",1);  session.commit();  session.close();  }  @Test  public void testGet() {  SqlSession session = factory.openSession();  Person person = session.selectOne("test.select", 2);  System.out.println(person);  session.commit();  session.close();  }    @Test  public void testList() {  SqlSession session = factory.openSession();  List<Person> list = session.selectList("test.selectList");  for (Person person : list) {  System.out.println(person);  }  session.close();  }    } |

7、请使用association节点实现根据用户id查询用户信息以及对应角色信息（Role实体类中有一个对象User,只写SQL映射文件）？

<!-- 01. 根据角色id 查询出角色信息-->  
<select id="selectRoleById" resultMap="roleMap">  
select id,name,userid from role where id=#{xxx}  
</select>  
  
 <!-- 02. 根据角色表中查询结果中的userid 查询用户信息  
 xxx就是resultmap中传递来的 userid-->  
<select id="selectUserByUserId" resultType="User">  
select userid,name from user where userid=#{xxx}  
</select>

已知部分代码，请补充。

<!--对应的roleMap 这种方式 推荐使用 因为使用延迟加载-->  
<resultMap id="rolrMap" type="Role">  
<id property="id" column="id"/>  
<result property="name" column="name"/>

<!-- Role有一个属性的类型是 user 域属性  
javaType:域属性对应的类型 -->  
<association property="user" javaType="User" select="selectUserByUserId"  
 column="userid"/>  
</resultMap>

8请使用collection节点实现获取指定用户的相关信息和地址列表（User实体类中有一个复杂类型的Address集合，只写SQL映射文件）？

<!-- 01. 根据用户id 查询出用户信息-->  
<select id="selectUserById" resultMap="userMap">  
select id,name from user where id=#{xxx}  
</select>  
  
<!-- 02. 根据用户表中查询结果中的userid 查询地址信息xxx就是resultmap中传递来的 userid-->  
<select id="selectAddressByUserId" resultType="Address">  
select id,name,userid from address where userid=#{xxx}  
</select>  
已知部分代码，请补充。

<!--对应的userMap 这种方式 推荐使用 因为使用延迟加载-->  
<resultMap id="userMap" type="User">  
<id property="id" column="id"/>  
<result property="name" column="name"/>  
<!-- User有一个属性的类型是 list   
javaType:域属性对应的类型 -->  
<collection property="addresses" javaType="Address" select="selectAddressByUserId"  
 column="userid"/>  
</resultMap>

9、请使用association节点实现根据用户id查询用户信息以及对应角色信息（Role实体类中有一个对象User,只写SQL映射文件）。

|  |
| --- |
| <!-- 01. 根据角色id 查询出角色信息-->  <select id="selectRoleById" resultMap="roleMap">  select id,name,userid from role where id=#{xxx}  </select>  <!-- 02. 根据角色表中查询结果中的userid 查询用户信息xxx就是resultmap中传递来的 userid-->  <select id="selectUserByUserId" resultType="User">  select userid,name from user where userid=#{xxx}  </select>  <!--对应的roleMap 这种方式 推荐使用 因为使用延迟加载-->  <resultMap id="rolrMap" type="Role">  <id property="id" column="id"/>  <result property="name" column="name"/>  <!-- Role有一个属性的类型是 user 域属性javaType:域属性对应的类型 -->  <association property="user" javaType="User" select="selectUserByUserId" column="userid"/>  </resultMap> |