## 1．#{}和${}的区别是什么？

在Mybatis中，有两种占位符

#{}解析传递进来的参数数据

${}对传递进来的参数原样拼接在SQL中

#{}是预编译处理，${}是字符串替换。

使用#{}可以有效的防止SQL注入，提高系统安全性。

## 2．当实体类中的属性名和表中的字段名不一样 ，怎么办 ？

第1种： 通过在查询的sql语句中定义字段名的别名，让字段名的别名和实体类的属性名一致

|  |
| --- |
| <select id=”selectorder”  parametertype=”int” resultetype=”me.gacl.domain.order”>  select order\_id id, order\_no orderno ,order\_price price form orders where order\_id=#{id};  </select> |

第2种： 通过来映射字段名和实体类属性名的一一对应的关系（建议采用第二种）

|  |
| --- |
| <select id="getOrder"  parameterType="int" resultMap="orderresultmap">  select \* from orders where order\_id=#{id}  </select>  <resultMap  type=”me.gacl.domain.order” id=”orderresultmap”>  <!–用id属性来映射主键字段–>  <id property=”id” column=”order\_id”>  <!–用result属性来映射非主键字段，property为实体类属性名，column为数据表中的属性–>  <result property = “orderno” column =”order\_no”/>  <result property=”price” column=”order\_price” />  </reslutMap> |

## 3．如何获取自动生成的(主)键值?

如果我们一般插入数据的话，如果我们想要知道刚刚插入的数据的主键是多少，我们可以通过以下的方式来获取

需求：

user对象插入到数据库后，新记录的主键要通过user对象返回，通过user获取主键值。

解决思路：

通过LAST\_INSERT\_ID()获取刚插入记录的自增主键值，在insert语句执行后，执行select LAST\_INSERT\_ID()就可以获取自增主键。

mysql:

|  |
| --- |
| <insert id="insertUser" parameterType="cn.itcast.mybatis.po.User">  <selectKey keyProperty="id" order="AFTER" resultType="int">  select LAST\_INSERT\_ID()  </selectKey>  INSERT INTO USER(username,birthday,sex,address) VALUES(#{username},#{birthday},#{sex},#{address})  </insert> |

oracle:

实现思路：

先查询序列得到主键，将主键设置到user对象中，将user对象插入数据库。

|  |
| --- |
| <!-- oracle  在执行insert之前执行select 序列.nextval() from dual取出序列最大值，将值设置到user对象 的id属性  -->  <insert id="insertUser" parameterType="cn.itcast.mybatis.po.User">  <selectKey keyProperty="id" order="BEFORE" resultType="int">  select 序列.nextval() from dual  </selectKey>  INSERT INTO USER(id,username,birthday,sex,address) VALUES( 序列.nextval(),#{username},#{birthday},#{sex},#{address})  </insert> |

## 4．在mapper中如何传递多个参数?

第一种：使用占位符的思想

在映射文件中使用#{0},#{1}代表传递进来的第几个参数

\*\*使用@param注解:来命名参数 \*\*

{0},#{1}方式

//对应的xml,#{0}代表接收的是dao层中的第一个参数，#{1}代表dao层中第二参数，更多参数一致往后加即可。

|  |
| --- |
| <select id="selectUser"resultMap="BaseResultMap">  select \* fromuser\_user\_t whereuser\_name = #{0} anduser\_area=#{1}  </select>  @param注解方式  public interface usermapper {  user selectuser(@param(“username”) string username,  @param(“hashedpassword”) string hashedpassword);  }  <select id=”selectuser” resulttype=”user”>  select id, username, hashedpassword  from some\_table  where username = #{username}  and hashedpassword = #{hashedpassword}  </select> |

第二种：使用Map集合作为参数来装载

|  |
| --- |
| try{  //映射文件的命名空间.SQL片段的ID，就可以调用对应的映射文件中的SQL  /\*\*  \* 由于我们的参数超过了两个，而方法中只有一个Object参数收集  \* 因此我们使用Map集合来装载我们的参数  \*/  Map<String, Object> map = new HashMap();  map.put("start", start);  map.put("end", end);  return sqlSession.selectList("StudentID.pagination", map);  }catch(Exception e){  e.printStackTrace();  sqlSession.rollback();  throw e;  }finally{  MybatisUtil.closeSqlSession();  }  <!--分页查询-->  <select id="pagination" parameterType="map" resultMap="studentMap">  /\*根据key自动找到对应Map集合的value\*/  select \* from students limit #{start},#{end};  </select> |

## 5．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？

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。

## 6．Mybatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复？

如果配置了namespace那么当然是可以重复的，因为我们的Statement实际上就是namespace+id

如果没有配置namespace的话，那么相同的id就会导致覆盖了。

## 7．为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里？

Hibernate属于全自动ORM映射工具，使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。

而Mybatis在查询关联对象或关联集合对象时，需要手动编写sql来完成，所以，称之为半自动ORM映射工具。

## 8．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。

## 9．接口绑定有几种实现方式,分别是怎么实现的?

接口绑定有两种实现方式：

一种是通过注解绑定,就是在接口的方法上面加上@Select@Update等注解里面包含Sql语句来绑定

另外一种就是通过xml里面写SQL来绑定,在这种情况下,要指定xml映射文件里面的namespace必须为接口的全路径名.

## 10．Mybatis是如何进行分页的？分页插件的原理是什么？

Mybatis使用RowBounds对象进行分页，它是针对ResultSet结果集执行的内存分页，而非物理分页，可以在sql内直接书写带有物理分页的参数来完成物理分页功能，也可以使用分页插件来完成物理分页。

分页插件的基本原理是使用Mybatis提供的插件接口，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，根据dialect方言，添加对应的物理分页语句和物理分页参数。

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

## 11．简述Mybatis的插件运行原理，以及如何编写一个插件

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。

## 12．Mybatis是否支持延迟加载？如果支持，它的实现原理是什么？

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，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。

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

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

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。

## 14．MyBatis与Hibernate有哪些不同？

Mybatis和hibernate不同，它不完全是一个ORM框架，因为MyBatis需要程序员自己编写Sql语句，不过mybatis可以通过XML或注解方式灵活配置要运行的sql语句，并将java对象和sql语句映射生成最终执行的sql，最后将sql执行的结果再映射生成java对象。

Mybatis学习门槛低，简单易学，程序员直接编写原生态sql，可严格控制sql执行性能，灵活度高，非常适合对关系数据模型要求不高的软件开发，例如互联网软件、企业运营类软件等，因为这类软件需求变化频繁，一但需求变化要求成果输出迅速。但是灵活的前提是mybatis无法做到数据库无关性，如果需要实现支持多种数据库的软件则需要自定义多套sql映射文件，工作量大。

Hibernate对象/关系映射能力强，数据库无关性好，对于关系模型要求高的软件（例如需求固定的定制化软件）如果用hibernate开发可以节省很多代码，提高效率。但是Hibernate的缺点是学习门槛高，要精通门槛更高，而且怎么设计O/R映射，在性能和对象模型之间如何权衡，以及怎样用好Hibernate需要具有很强的经验和能力才行。

总之，按照用户的需求在有限的资源环境下只要能做出维护性、扩展性良好的软件架构都是好架构，所以框架只有适合才是最好。

## 15．JDBC 编程有哪些不足？MyBatis 是如何解决的？

（1）数据库的连接创建、释放频繁，从而影响性能，mybatis 通过配置 SqlMapConfig.xml 文件使用连接池的方式解决了数据库连接创建和释放频繁所造成的性能影响。

（2）大量的 sql 存在于代码之中，造成代码的可维护性低。mybatis 只用 xml 文件对 sql 进行统一管理，方便维护。

（3）jdbc 操作中存在参数时，需要准确的定位参数的位置和对应占位符的个数，否则会出错。mybatis 通过提供参数对象的方式解决了该问题。

（4）sql 语句在编写时，如果存在动态条件则不容易处理。mybatis 提供动态 sql 编写机制，时用户可以根据自己传入参数的情况进行 sql 语句的动态编写。

（5）对查询时返回的结果集处理是一件比较麻烦的事情，而随着查询结果的改变，处理结果集的代码也随之改变。mybatis 通过把结果集映射成对应的 java 对象，解决了结果集处理麻烦的问题。

## 16．MyBatis 编程步骤

（1）创建 SqlSessionFactory

（2）通过 SqlSessionFactory 获取 SqlSession

（3）通过 SqlSession 执行数据库操作

（4）提交事务

（5）关闭会话

## 17．xml 映射文件中有哪些标签

（1）常见操作标签：select、insert、update、delete

（2）动态 sql 标签：trim、where、set、foreach、if、choose、when、otherwise、bind

（3）其他标签：(sql 片段标签)、(引入 sql 片段标签)

## 18．mybatis 如何确定唯一的操作 sql（MappedStatement）

mybatis 通过 namespace 和 MappedStatement 的 id 值确定唯一的 MappedStatement。

## 19．mybatis 的 mapper 接口中的接口方法可以通过不同参数的方式对方法进行重载吗？

不可以。在 mybatis 在执行方法前是通过 namespace+id 进行 MappedStatement 的确定，跟参数没有关系，所以不能实现重载。

## 20．Mapper 接口的工作原理

mybatis 在运行时使用 JDK 动态代理生成代理对象，在调用接口方法时，代理对象会拦截接口方法，转而运行 MappedStatement 的 sql 语句，并生成结果返回。

## 21．MyBatis 在批量插入时能返回数据库主键列表吗？

可以。在 JDBC 执行批量插入时可以返回主键列表，而 MyBatis 是基于 JDBC 的实现，同样可以返回主键列表。

## 22．MyBatis 是否支持延迟加载？如果支持，他的实现原理是什么？

mybatis 支持延迟加载，但有限制。他支持一对一、一对多的延迟加载。

要使 mybatis 支持延迟加载，需要在配置文件中配置 lazyLoadingEnabled 值为 true。mybatis 通过 CGLIB 创建目标对象的代理对象，在调用目标对象的 get 方法时，进行拦截并检查关联对象是否为空，如果为空则调用 sql 进行相应对象的查询并通过 set 方法进行数据填充，最后在返回对应关联对象。

## 23．在 MyBatis 的 xml 映射文件中，不同的 xml 文件 id 是否可以重复？

在 mybatis 实现中要获取一个 MappedStatement 是通过 namespace+id 进行唯一确定的，而 namespace 是最佳实践，但并不是必须的，如果没有 namespace 的存在，重复的 id 会造成 MappedStatement 被覆盖。所以结合最佳实践而言，不同的 xml 文件 id 建议别重复使用。

## 24．MyBatis 中最终执行数据库操作的 Executor 执行器，在 mybatis 中有那些基本的执行器？他们的区别是什么？

在 mybatis 中有三种基本的执行器，他们分别是：SimpleExecutor、ReuseExecutor、BatchExecutor。他们区别如下：

（1）SimpleExecutor：每执行一次 update 或 select 都开启一个 Statement 对象，执行完后关闭该对象。

（2）ReuseExecutor: 执行 update 或 select 时，以 sql 作为 key 进行缓存查找 Statement，如果找到直接执行，如果没找到则创建并缓存，执行完操作后不关闭该对象，以备下次执行时使用。

（3）BatchExecutor: 在执行 update 时，所有的 sql 都将添加到批处理器中进行统一处理。他缓存了多个 Statement 对象，所有的对象都等待着屁处理器统一执行。

## 25．MyBatis 是否支持枚举类的映射？

支持。mybatis 可以通过自定义 TypeHandler, 并实现setParameter()和getResult()方法进行类型转换。

## 26．简述mybatis缓存机制

mybatis的查询缓存分为一级缓存和二级缓存，一级缓存是SqlSession级别的缓存，二级缓存时mapper级别的缓存，二级缓存是多个SqlSession共享的。mybatis通过缓存机制减轻数据压力，提高数据库性能。

（1）一级缓存：

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，不同的SqlSession之间缓存数据区域（HashMap）是互相不影响的。

一级缓存的作用域是SqlSession范围的，当在同一个SqlSession中执行两次相同的sql语句时，第一次执行完毕会将数据库中查询的数据写到缓存（内存）中，第二次查询时会从缓存中获取数据，不再去底层进行数据库查询，从而提高了查询效率。需要注意的是：如果SqlSession执行了DML操作（insert、update、delete），并执行commit（）操作，mybatis则会清空SqlSession中的一级缓存，这样做的目的是为了保证缓存数据中存储的是最新的信息，避免出现脏读现象。

当一个SqlSession结束后该SqlSession中的一级缓存也就不存在了，Mybatis默认开启一级缓存，不需要进行任何配置。

注意：Mybatis的缓存机制是基于id进行缓存，也就是说Mybatis在使用HashMap缓存数据时，是使用对象的id作为key，而对象作为value保存

（2）二级缓存：

二级缓存是mapper级别的缓存，使用二级缓存时，多个SqlSession使用同一个Mapper的sql语句去操作数据库，得到的数据会存在二级缓存区域，它同样是使用HashMapper进行数据存储，相比一级缓存SqlSession，二级缓存的范围更大，多个SqlSession可以共用二级缓存，二级缓存是跨SqlSession的。

二级缓存是多个SqlSession共享的，其作用域是mapper的同一个namespace，不同的SqlSession两次执行相同的namespace下的sql语句，且向sql中传递的参数也相同，即最终执行相同的sql语句，则第一次执行完毕会将数据库中查询的数据写到缓存（内存），第二次查询时会从缓存中获取数据，不再去底层数据库查询，从而提高查询效率。

Mybatis默认没有开启二级缓存，需要在setting全局参数中配置开启二级缓存。

|  |
| --- |
| 在mybatis-config.xml中配置：  [html] view plain copy  <settings>  <setting name="cacheEnabled" value="true"/>  </settings>  cacheEnabled的value为true表示在此配置文件下开启二级缓存，该属性默认为false。  在EmployeeMapper.xml中配置：  [html] view plain copy  <!-- 开启当前mapper的namespace下的二级缓存 -->  <cache eviction="LRU" flushInterval="60000" size="512" readOnly="true"/> |

以上配置创建了一个LRU缓存，并每隔60秒刷新，最大存储512个对象，而且返回的对象被认为是只读。

## 27．什么是MyBatis的接口绑定,有什么好处

接口映射就是在IBatis中任意定义接口,然后把接口里面的方法和SQL语句绑定, 我们直接调用接口方法就可以,这样比起原来了SqlSession提供的方法我们可以有更加灵活的选择和设置。

## 28．接口绑定有几种实现方式,分别是怎么实现的?

接口绑定有两种实现方式,

（1）一种是通过注解绑定,就是在接口的方法上面加上 @Select@Update等注解里面包含Sql语句来绑定。

（2）另外一种就是通过xml里面写SQL来绑定, 在这种情况下,要指定xml映射文件里面的namespace必须为接口的全路径名。

## 29．什么情况下用注解绑定,什么情况下用xml绑定 当Sql语句比较简单时候,用注解绑定

当SQL语句比较复杂时候,用xml绑定,一般用xml绑定的比较多

## 30．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

## 31．MyBatis实现一对多有几种方式,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collection节点配置一对多的类就可以完成。

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,也是通过配置collection,但另外一个表的 查询通过select节点配置

## 32．Mybatis是否支持延迟加载？如果支持，它的实现原理是什么？

（1）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。

（2）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。

## 33．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，<parameterMap>标签会被解析为ParameterMap对象，其每个子元素会被解析为ParameterMapping对象。<resultMap>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。