Mybatis常见面试题

#{}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。

${}是Properties文件中的变量占位符，它可以用于标签属性值和sql内部，属于静态文本替换，比如${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()

当实体类中的属性名和表中的字段名不一样 ， 怎么办 ？

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**

```
 <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>
```

我认为第二种方式会好一点。

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>
复制代码
```

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>
```

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>
复制代码
```

- @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>
```

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。**

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。

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。

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。

详情可参考：

- www.cnblogs.com/soundcode/p...

Mybatis比IBatis比较大的几个改进是什么

- a.**有接口绑定,包括注解绑定sql和xml绑定Sql**，
- b.动态sql由原来的节点配置变成OGNL表达式,
- c. 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

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.

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`

分页插件使用参考资料：

- www.cnblogs.com/kangoroo/p/...
- blog.csdn.net/yuchao2015/...
- www.cnblogs.com/ljdblog/p/6...

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。**

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。

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<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。

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。