1、#{}和${}的区别是什么？

#{}是预编译处理，${}是字符串替换。

Mybatis 在处理#{}时，会将 sql 中的#{}替换为?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。

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<resultMap>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>
```

3、 模糊查询 like 语句该怎么写?

第 1 种：在 Java 代码中添加 sql 通配符。

```
string wildcardname = “%smi%”;
list<name> names = mapper.selectlike(wildcardname);

<select id=”selectlike”>
 select * from foo where bar like #{value}
</select>
```

第 2 种：在 sql 语句中拼接通配符，会引起 sql 注入

```
string wildcardname = “smi”;
list<name> names = mapper.selectlike(wildcardname);

<select id=”selectlike”>
 select * from foo where bar like "%"#{value}"%"
</select>
```

4、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。

5、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。

6、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？

答：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，将列别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 会忽略列名大小写，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。

7、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();
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();
}
```

8、如何获取自动生成的(主)键值?

insert 方法总是返回一个 int 值 - 这个值代表的是插入的行数。

而自动生成的键值在 insert 方法执行完后可以被设置到传入的参数对象中。

示例：

```
<insert id=”insertname” usegeneratedkeys=”true” keyproperty=”id”>
 insert into names (name) values (#{name})
</insert>

name name = new name();
name.setname(“fred”);

int rows = mapper.insertname(name);
// 完成后,id已经被设置到对象中
system.out.println(“rows inserted = ” + rows);
system.out.println(“generated key value = ” + name.getid());
```

9、在 mapper 中如何传递多个参数?

第1种：

```
//DAO层的函数

Public UserselectUser(String name,String area);
```

```
//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>
```

第2种： 使用 @param 注解：

```
    import org.apache.ibatis.annotations.param;
        public interface usermapper {
         user selectuser(@param(“username”) string username,
         @param(“hashedpassword”) string hashedpassword);
        }
```

然后,就可以在xml像下面这样使用(推荐封装为一个map,作为单个参数传递给mapper)：

```
    <select id=”selectuser” resulttype=”user”>
         select id, username, hashedpassword
         from some_table
         where username = #{username}
         and hashedpassword = #{hashedpassword}
    </select>
```

10、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。

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

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。

12、 一对一、一对多的关联查询 ？

```xml
<mapper namespace="com.lcb.mapping.userMapper">
    <!--association  一对一关联查询 -->
    <select id="getClass" parameterType="int" resultMap="ClassesResultMap">
        select * from class c,teacher t where c.teacher_id=t.t_id and c.c_id=#{id}
    </select>
    <resultMap type="com.lcb.user.Classes" id="ClassesResultMap">
        <!— 实体类的字段名和数据表的字段名映射 -->
        <id property="id" column="c_id"/>
        <result property="name" column="c_name"/>
        <association property="teacher" javaType="com.lcb.user.Teacher">
            <id property="id" column="t_id"/>
            <result property="name" column="t_name"/>
        </association>
    </resultMap>

    <!--collection  一对多关联查询 -->
    <select id="getClass2" parameterType="int" resultMap="ClassesResultMap2">
        select * from class c,teacher t,student s where c.teacher_id=t.t_id and c.c_id=s.class_id and c.c_id=#{id}
    </select>
    <resultMap type="com.lcb.user.Classes" id="ClassesResultMap2">
        <id property="id" column="c_id"/>
        <result property="name" column="c_name"/>
        <association property="teacher" javaType="com.lcb.user.Teacher">
            <id property="id" column="t_id"/>
            <result property="name" column="t_name"/>
        </association>
        <collection property="student" ofType="com.lcb.user.Student">
            <id property="id" column="s_id"/>
            <result property="name" column="s_name"/>
        </collection>
    </resultMap>

</mapper>
```