https://blog.csdn.net/a745233700/article/details/80977133

1、什么是 Mybatis？

（1）Mybatis 是一个半 ORM（对象关系映射）框架，它内部封装了 JDBC，开发时只需要关注 SQL 语句本身，不需要花费精力去处理加载驱动、创建连接、创建 statement 等繁杂的过程。程序员直接编写原生态 sql，可以严格控制 sql 执行性能，灵活度高。

（2）MyBatis 可以使用 XML 或注解来配置和映射原生信息，将 POJO 映射成数据库中的记录，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

（3）通过 xml 文件或注解的方式将要执行的各种 statement 配置起来，并通过 java 对象和 statement 中 sql 的动态参数进行映射生成最终执行的 sql 语句，最后由 mybatis 框架执行 sql 并将结果映射为 java 对象并返回。（从执行 sql 到返回 result 的过程）。

2、Mybaits 的优点：

（1）基于 SQL 语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL 写在 XML 里，解除 sql 与程序代码的耦合，便于统一管理；提供 XML 标签，支持编写动态 SQL 语句，并可重用。

（2）与 JDBC 相比，减少了 50%以上的代码量，消除了 JDBC 大量冗余的代码，不需要手动开关连接；

（3）很好的与各种数据库兼容（因为 MyBatis 使用 JDBC 来连接数据库，所以只要 JDBC 支持的数据库 MyBatis 都支持）。

（4）能够与 Spring 很好的集成；

（5）提供映射标签，支持对象与数据库的 ORM 字段关系映射；提供对象关系映射标签，支持对象关系组件维护。

3、MyBatis 框架的缺点：

（1）SQL 语句的编写工作量较大，尤其当字段多、关联表多时，对开发人员编写 SQL 语句的功底有一定要求。

（2）SQL 语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

4、MyBatis 框架适用场合：

（1）MyBatis 专注于 SQL 本身，是一个足够灵活的 DAO 层解决方案。

（2）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，MyBatis 将是不错的

选择。

5、MyBatis 与 Hibernate 有哪些不同？

（1）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。

（2）Mybatis 直接编写原生态 sql，可以严格控制 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，因为这类软件需求变化频繁，一但需求变化要求迅速输出成果。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。

（3）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，如果用 hibernate 开发可以节省很多代码，提高效率。

6、#{}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。

7、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>
```

8、 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>
```

9、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 接口即 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 Mapper 的 Statement 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 MapperStatement。在 Mybatis 中，每一个<select>、<insert>、<update>、<delete>标签，都会被解析为一个 MapperStatement 对象。

举例：com.mybatis3.mappers.StudentDao.findStudentById，可以唯一找到 namespace 为 com.mybatis3.mappers.StudentDao 下面 id 为 findStudentById 的 MapperStatement。

Mapper 接口里的方法，是不能重载的，因为是使用 全限名+方法名 的保存和寻找策略。Mapper 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 Mapper 接口生成代理对象 proxy，代理对象会拦截接口方法，转而执行 MapperStatement 所代表的 sql，然后将 sql 执行结果返回。

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。

11、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？

第一种是使用<resultMap>标签，逐一定义数据库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，将列的别名书写为对象属性名。

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。

12、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();
    }catch(Exception e){
      e.printStackTrace();
      sqlSession.rollback();
      throw e;
  }
    finally {
        sqlsession.close();
    }
```

13、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。

如果采用自增长策略，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());
```

14、在 mapper 中如何传递多个参数?

（1）第一种：

//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>
```

（2）第二种： 使用 @param 注解:

```
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):