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。

```
1     <select id=”selectorder” parametertype=”int” resultetype=”me.gacl.domain.order”>
2        select order_id id, order_no orderno ,order_price price form orders where order_id=#{id};
3     </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。

```
 1  <select id="getOrder" parameterType="int" resultMap="orderresultmap">
 2         select * from orders where order_id=#{id}
 3     </select>
 4
 5    <resultMap type=”me.gacl.domain.order” id=”orderresultmap”>
 6         <!-用id属性来映射主键字段->
 7         <id property=”id” column=”order_id”>
 8
 9         <!-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->
10         <result property = “orderno” column =”order_no”/>
11         <result property=”price” column=”order_price” />
12     </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。

```
1     string wildcardname = “%smi%”;
2     list<name> names = mapper.selectlike(wildcardname);
3
4     <select id=”selectlike”>
5      select * from foo where bar like #{value}
6     </select>
```

第2种：在sql语句中拼接通配符，会引起sql注入

```
1     string wildcardname = “smi”;
2     list<name> names = mapper.selectlike(wildcardname);
3
4     <select id=”selectlike”>
5          select * from foo where bar like "%"#{value}"%"
6     </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 ▼

```
1    <insert id="insertname">
2         insert into names (name) values (#{value})
3    </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):
<select id=”selectuser” resulttype=”user”>
         select id, username, hashedpassword
         from some_table
         where username = #{username}
         and hashedpassword = #{hashedpassword}
</select>

 (3) 第三种: 多个参数封装成map
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();
 }
```

15、Mybatis动态sql有什么用？执行原理？有哪些动态sql？

Mybatis动态sql可以在Xml映射文件内，以标签的形式编写动态sql，执行原理是根据表达式的值 完成逻辑判断并动态拼接sql的功能。

Mybatis提供了9种动态sql标签：trim | where | set | foreach | if | choose | when | otherwise | bind。

16、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：、、、、，加上动态sql的9个标签，其中为sql片段标签，通过标签引入sql片段，为不支持自增的主键生成策略标签。

17、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；

原因就是namespace+id是作为Map<String, MapperStatement>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。

18、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。

19、 一对一、一对多的关联查询 ？

```
<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>
```

20、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。

21、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里面的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。

22、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。

23、Mybatis的一级、二级缓存:

1）一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当 Session flush 或 close 之后，该 Session 中的所有 Cache 就将清空，默认打开一级缓存。

2）二级缓存与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap 存储，不同在于其存储作用域为 Mapper(Namespace)，并且可自定义存储源，如 Ehcache。默认不打开二级缓存，要开启二级缓存，使用二级缓存属性类需要实现Serializable序列化接口(可用来保存对象的状态),可在它的映射文件中配置；

3）对于缓存数据更新机制，当某一个作用域(一级缓存 Session/二级缓存Namespaces)的进行了C/U/D 操作后，默认该作用域下所有 select 中的缓存将被 clear。

24、什么是MyBatis的接口绑定？有哪些实现方式?

接口绑定，就是在MyBatis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。

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。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。

25、使用MyBatis的mapper接口调用时有哪些要求?

① Mapper接口方法名和mapper.xml中定义的每个sql的id相同； ② Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同； ③ Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同； ④ Mapper.xml文件中的namespace即是mapper接口的类路径。

26、Mapper编写有哪几种方式?

> ***第一种：接口实现类继承SqlSessionDaoSupport：***使用此种方法需要编写mapper接口，mapper接口实现类、mapper.xml文件。 （1）在sqlMapConfig.xml中配置mapper.xml的位置 （2）定义mapper接口 （3）实现类集成SqlSessionDaoSupport mapper方法中可以this.getSqlSession()进行数据增删改查。 （4）spring 配置
>
> ***第二种：使用org.mybatis.spring.mapper.MapperFactoryBean：*** （1）在sqlMapConfig.xml中配置mapper.xml的位置，如果mapper.xml和mappre接口的名称相同且在同一个目录，这里可以不用配置 （2）定义mapper接口： ①mapper.xml中的namespace为mapper接口的地址 ②mapper接口中的方法名和mapper.xml中的定义的statement的id保持一致 ③Spring中定义
>
> ***第三种：使用mapper扫描器：*** （1）mapper.xml文件编写： mapper.xml中的namespace为mapper接口的地址； mapper接口中的方法名和mapper.xml中的定义的statement的id保持一致； 如果将mapper.xml和mapper接口的名称保持一致则不用在sqlMapConfig.xml中进行配置。 （2）定义mapper接口： 注意mapper.xml的文件名和mapper的接口名称保持一致，且放在同一个目录 （3）配置mapper扫描器： （4）使用扫描器后从spring容器中获取mapper的实现对象。

27、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。

编写插件：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。