#### 1、什么是Mybatis？

（1）半ORM框架，它内部封装了JDBC。用 XML或注解来将 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。

（2）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

#### 2、 #{}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${}替换成变量的值。使用#{}可以有效的防止SQL注入，提高系统安全性，即#是将传入的值当做字符串的形式，先替换为?号，然后调用PreparedStatement的set方法来赋值，而$是将传入的数据直接显示生成sql语句

#### **3、Xml和Mapper接口对应，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？**

**当调用一个接口方法时，通过接口全限定名+方法名得到一个方法id，对应着XML中的namespace+方法id**

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

#### 4、MyBatis中用到了哪些设计模式

动态代理：动态生成Mapper实现类执行xml中的sql语句

#### 5、MyBatis有什么优势；

#### 6、MyBatis如何做事务管理；

#### **7、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。

#### ****8、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。

#### ****9、如何执行批量插入?****

#### ****10、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。

<insert id=”insertname” usegeneratedkeys=”true” keyproperty=”id”>

insert into names (name) values (#{name})

</insert>

#### ****11、在mapper中如何传递多个参数?****

一个参数，啥也不用写；

两三个参数@param 注解，mybatis会自动将其封装为Map，@param的值就是Map的Key；不写注解#{0},#{1}别用

多个用Map或者DTO

多个DTO的话：(@Param (” user ”) SysUser user, @Param (” role”) SysRole role) ===》 #{user.id} #{role.Id}

#### ****12、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。

#### ****13、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？****

<resultMap>、<parameterMap>、<sql>、<include>、<selectKey>，加上动态sql的9个标签，其中<sql>为sql片段标签，通过<include>标签引入sql片段，<selectKey>为不支持自增的主键生成策略标签

#### ****14、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。

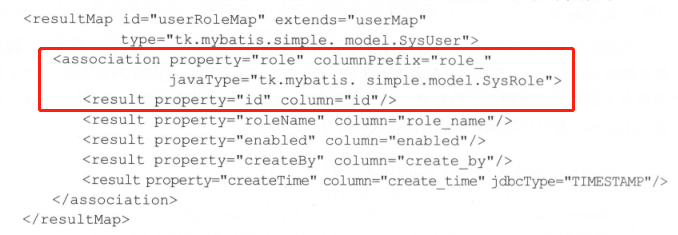
但是，在以前的Mybatis版本的namespace是可选的，不过新版本的namespace已经是必须的了。

#### ****15、 一对一、一对多的关联查询 ？****

##### 一对一映射：

a自动映射： SysUser 里头有一个sysRole字段，一对一，select u.user\_name username,r.role\_name “role.roleName” from sys\_user u join sys\_role r

b <resultMap>一对一映射：不方便

c <association>标签一对一

##### 一对多映射

SysUser 里有一个List<SysRole> roleList;角色集合

Collection集合的嵌套结果映射:



#### ****16、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。

#### ****17、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。

#### ****18、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。

#### ****19、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<cache/> ；

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 掉并重新更新，如果开启了二级缓存，则只根据配置判断是否刷新。

#### ****20、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。

#### 21、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。

#### 22、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

<mappers>

    <mapper resource="mapper.xml文件的地址" />

    <mapper resource="mapper.xml文件的地址" />

</mappers>

（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

<bean id=" " class="mapper接口的实现">

    <property name="sqlSessionFactory" ref="sqlSessionFactory"></property>

</bean>

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

<mappers>

    <mapper resource="mapper.xml文件的地址" />

    <mapper resource="mapper.xml文件的地址" />

</mappers>

（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

<bean id="" class="org.mybatis.spring.mapper.MapperFactoryBean">

    <property name="mapperInterface"   value="mapper接口地址" />

    <property name="sqlSessionFactory" ref="sqlSessionFactory" />

</bean>

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：

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

    <property name="basePackage" value="mapper接口包地址"></property>

    <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>

</bean>

（4）使用扫描器后从spring容器中获取mapper的实现对象。

#### 23、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。

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。

#### 24、<where>标签

如果有返回值，就插入一个where；where紧挨着的字符串是以and和or开头，就去掉

#### 25、<foreach>

<foreach collection =” list ” open =” (” close =” )” separator =” , ” item =” id ” index =” i ” >

#{id}

</foreach>

一个或者多个参数，推荐使用@Param注解指定名字

<foreach>实现批量插入

int insertList(List<SysUser> userList);

