**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。

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,Mybatis的优点**

1、简单易学。mybatis本身就很小且简单。没有任何第三方依赖，最简单安装只要两个jar加配置几个sql映射文件，易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现；

2、灵活。mybatis不会对应用程序或者数据库的现有设计强加任何影响。 sql写在xml里，便于统一管理和优化。通过sql基本上可以实现我们不使用数据访问框架可以实现的所有功能，或许更多；

3、解除sql与程序代码的耦合。通过提供DAO层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试。sql和代码的分离，提高了可维护性；

4、提供映射标签，支持对象与数据库的orm字段关系映射；

5、提供对象关系映射标签，支持对象关系组建维护；

6、提供xml标签，支持编写动态sql。

**3,Mybatis的缺点**

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。

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,#{}和${}的区别是什么？**

1. #将传入的数据都当成一个字符串，会对自动传入的数据加一个双引号。如：order by #user\_id#，如果传入的值是111,那么解析成sql时的值为order by "111", 如果传入的值是id，则解析成的sql为order by "id".

$将传入的数据直接显示生成在sql中。如：order by $user\_id$，如果传入的值是111,那么解析成sql时的值为order by 111, 如果传入的值是id，则解析成的sql为order by id.

2. #方式能够很大程度防止sql注入。

$方式无法防止Sql注入。

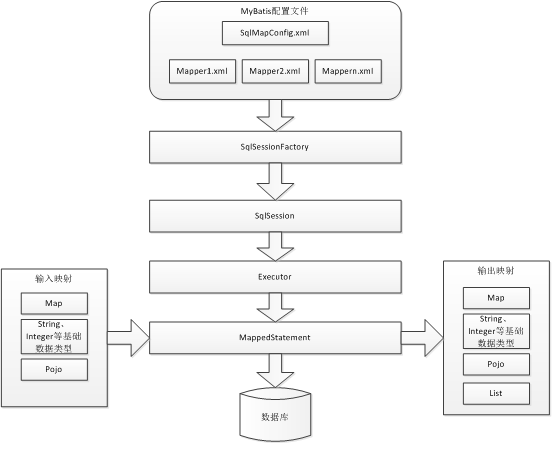
Mybatis在处理#{}时，会将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；Mybatis在处理${}时，就是把${}替换成变量的值。

select \* from emp where empno = 7360 or 1=1 -- ${}

select \* from emp where empno = '7369' -- #{}

没有特殊需求情况下，都要使用#{}

**6,Mybatis执行流程**



1、 mybatis配置

SqlMapConfig.xml，此文件作为mybatis的全局配置文件，配置了mybatis的运行环境等信息。

mapper.xml文件即sql映射文件，文件中配置了操作数据库的sql语句。此文件需要在SqlMapConfig.xml中加载。

2、 通过mybatis环境等配置信息构造SqlSessionFactory即会话工厂

3、 由会话工厂创建sqlSession即会话，操作数据库需要通过sqlSession进行。

4、 mybatis底层自定义了Executor执行器接口操作数据库，Executor接口有两个实现，一个是基本执行器、一个是缓存执行器。

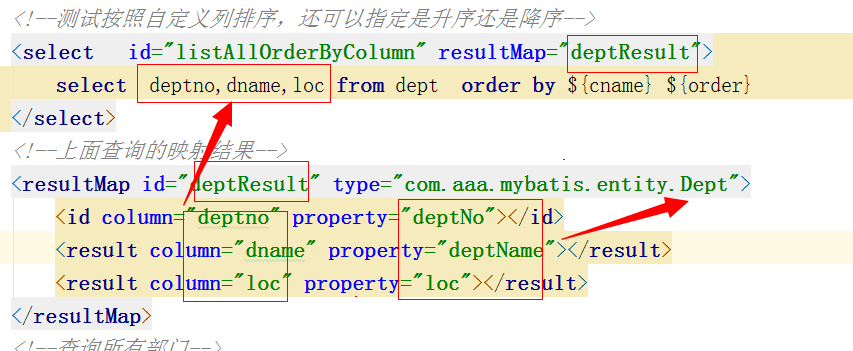
5、 MappedStatement也是mybatis一个底层封装对象，它包装了mybatis配置信息及sql映射信息等。mapper.xml文件中一个sql对应一个Mapped Statement对象，sql的id即是Mapped statement的id。

6、 MappedStatement对sql执行输入参数进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql前将输入的java对象映射至sql中，输入参数映射就是jdbc编程中对preparedStatement设置参数。

7、 MappedStatement对sql执行输出结果进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql后将输出结果映射至java对象中，输出结果映射过程相当于jdbc编程中对结果的解析处理过程。

**7,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，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。

(数据库中部门对象列为dname，实体为deptName select deptno,dname deptName,loc from dept )

**8,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,多对一，collection指的就是一对多，多对多查询。在Mybatis配置文件中，可以配置是否启用延迟加载lazyLoadingEnabled=true|false。

**延迟加载代码实现（一对多collection 查询部门时，查出该部门的员工）：**

**实体：Dept Emp**

需要在Dept中加入private List<Emp> empList; 和getter setter

**dao：**DeptDao

/\*\*

\* 查询所有部门

\* @return

\*/

List<Dept> listAll();

EmpDao:

/\*\*

\* 根据部门编号获取员工

\* @param deptNo

\* @return

\*/

List<Emp> listEmpByDeptNo(int deptNo);

**配置文件:DeptMapper.xml :**

<!--查询所有部门-->

<select id="listAll" resultMap="deptEmp">

select \* from dept

</select>

<!--映射-->

<resultMap id="deptEmp" type="com.aaa.mybatis.entity.Dept">

<id property="deptNo" column="deptno"/>

<result property="deptName" column="deptname"/>

<result property="loc" column="loc"/>

<collection property="empList" column="deptno" ofType="com.aaa.mybatis.entity.Emp"

select="com.aaa.mybatis.dao.EmpDao.listEmpByDeptNo">

</collection>

</resultMap>

EmpMapper.xml:

<!--根据部门编号获取员工-->

<select id="listEmpByDeptNo" resultType="com.aaa.mybatis.entity.Emp">

select empno,empname,salary from emp where deptno=#{deptNo}

</select>

**主配置文件：mybatis-config.xml 中的 settings**

<!--开启全局懒加载-->

<setting name="lazyLoadingEnabled" value="true"></setting>

**测试：**

/\*\*

\* 测试延时加载

\*/

@Test

public void testListAll(){

SqlSession sqlSession = null;

try {

sqlSession = SqlSessionFacotryUtil.getSqlSession();

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

List<Dept> depts = deptDao.listAll();

if(depts!=null&&depts.size()>0){

for (Dept dept : depts) {

dept.getEmpList();

}

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(sqlSession!=null)

sqlSession.close();

}

}

如果下面代码注释掉，开启懒加载，底层查询数据库时，只执行了部门查询，如果不开启懒加载，部门和该部门的员工都查询了

if(depts!=null&&depts.size()>0){

for (Dept dept : depts) {

dept.getEmpList();

}

}

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intercept()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。（ctrl+n ProxyFactory 实现类：JavassistProxyFactory 的invoke和CglibProxyFactory的 intercept）

<https://my.oschina.net/wenjinglian/blog/1857581?from=singlemessage>

（Javaassist 就是一个用来 处理 Java 字节码的类库。）

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。

**9,模糊查询like语句该怎么写?**

**在Java代码中添加sql通配符。**

List<Dept> depts = deptDao.listByLike("%dev%");

mapper中：

<!--模糊查询 like-->

<select id="listByLike" parameterType="string" resultType="com.aaa.mybatis.entity.Dept">

select deptno,dname as deptName,loc from dept where dname like #{param}

</select>

**在sql语句中拼接通配符**

select deptno,dname as deptName,loc from dept where dname like '%${param}%'

'%#{param}%' 写法错误的

select deptno,dname as deptName,loc from dept where dname like concat('%',# {param},'%')

oracle : '%'||# {param}||'%' concat(concat('%',#{param}),'%')

mysql: concat('%',# {param},'%')

mysql 或者oracle '%${param}%'

**10,请问MyBatis中的动态SQL是什么意思？**

对于一些复杂的查询，我们可能会指定多个查询条件，但是这些条件可能存在也可能不存在，需要根据用户指定的条件动态生成SQL语句。如果不使用持久层框架我们可能需要自己拼装SQL语句，还好MyBatis提供了动态SQL的功能来解决这个问题。MyBatis中用于实现动态SQL的元素主要有：

- if

- choose / when / otherwise

- trim

- where

- set

- foreach

**11,请说明一下MyBatis中命名空间（namespace）的作用是什么？**

在大型项目中，可能存在大量的SQL语句，这时候为每个SQL语句起一个唯一的标识（ID）就变得并不容易了。为了解决这个问题，在MyBatis中，可以为每个映射文件起一个唯一的命名空间，这样定义在这个映射文件中的每个SQL语句就成了定义在这个命名空间中的一个ID。只要我们能够保证每个命名空间中这个ID是唯一的，即使在不同映射文件中的语句ID相同，也不会再产生冲突了。

1，隔离语句

2，绑定接口

**12,如何执行批量插入?**

**SQL 基础：**

-- 批量插入

-- mysql

insert into dept values (18,'dt1','1floor'),(19,'dt1','1floor'),(20,'dt1','1floor');

-- oracle

insert into dept

select 18,'dt1','1floor' from dual UNION

select 19,'dt1','1floor' from dual UNION

select 20,'dt1','1floor' from dual

**dao:**

/\*\*

\* 批量插入

\* @param depts

\* @return

\*/

int batchAdd(List<Dept> depts);

**mapper:**

<!--批量插入-->

<insert id="batchAdd">

insert into dept values

<foreach collection="list" item="dept" index="i" separator="," >

(null,#{dept.deptName},#{dept.loc})

</foreach>

</insert>

**测试：**

List<Dept> deptList = new ArrayList<Dept>(5);

Dept dept = new Dept();

dept.setDeptName("team33");

dept.setLoc("33floor");

deptList.add(dept);

Dept dept1 = new Dept();

dept1.setDeptName("team44");

dept1.setLoc("44floor");

deptList.add(dept1);

Dept dept2 = new Dept();

dept2.setDeptName("team55");

dept2.setLoc("55floor");

deptList.add(dept2);

Dept dept3 = new Dept();

dept3.setDeptName("team66");

dept3.setLoc("66floor");

deptList.add(dept3);

Dept dept4 = new Dept();

dept4.setDeptName("team77");

dept4.setLoc("77floor");

deptList.add(dept4);

int i = deptDao.batchAdd(deptList);

sqlSession.commit();

**13,如何获取自动生成的(主)键值?**

MySQL：Mapper文件insert语句设置    useGeneratedKeys="true" keyProperty="id"

<!--添加部门并返回自增ID-->

<insert id="add" useGeneratedKeys="true" keyProperty="deptNo">

insert into dept values(null,#{deptName},#{loc})

</insert>

自增生成之后的值赋值实体中的属性deptNo

Oracle：Mapper文件insert语句增加

<selectKey keyProperty="id" order="BEFORE" resultType="Integer">

    select xxx\_SEQ.nextval from dual

</selectKey>

**14,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？**

**hql:**" from Emp" Emp 实体类

session.save(emp);

session.saveOrUpdate(emp);

session.update(emp);

session.deleteByke(empno);

session.delete(emp);

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（hibernate可以操作实体直接查询，不需要手写语句）。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。

**15,在mapper中如何传递多个参数?**

（1）第一种：#{0}或者是#{arg0} 看是否是3.4.2及之前#{0} 或者之后#{arg0}

**dao:**

/\*\*

\* 添加，传递多个参数

\* @param dname

\* @param loc

\* @return

\*/

int addA(String dname,String loc);

**mapper:**

<!-- 添加，多个参数 3.4.2版本之前#{0} 之后#{arg0} https://www.cnblogs.com/zhangmingcheng/p/9922236.html-->

<insert id="addA">

insert into dept values(null,#{arg0},#{arg1})

</insert>

（2）第二种： 使用 @param 注解:

**dao:**

/\*\*

\* 添加，@Param传递多个参数

\* @param dname

\* @param loc

\* @return

\*/

int addB(@Pa <!--添加，@Param传递多个参数-->

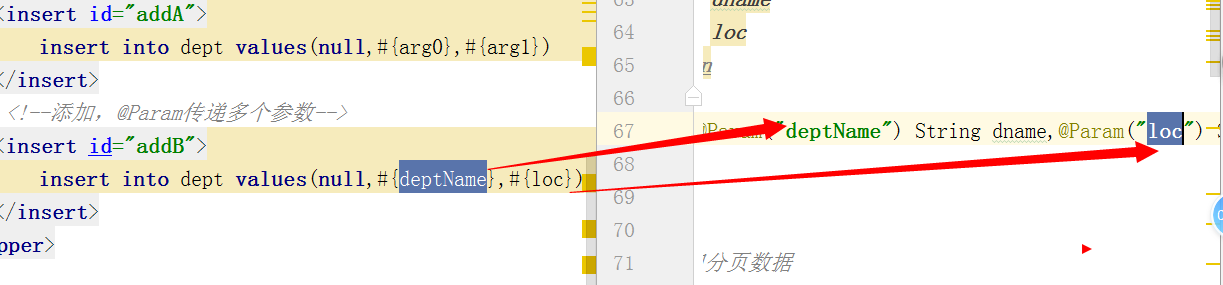
**mapper:**

<insert id="addB">

insert into dept values(null,#{deptName},#{loc})

</insert>

@Param("deptName") String dname,@Param("loc") String loc);



（3）第三种：多个参数封装成map或者是实体

**dao方法：**

/\*\*

\* 添加

\* @param dept

\* @return

\*/

int add(Dept dept);

**mapper：**

<!--添加部门并返回自增ID-->

<insert id="add" useGeneratedKeys="true" keyProperty="deptNo">

insert into dept values(null,#{deptName},#{loc})

</insert>

**15, 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。

**一对一、一对多的关联查询 ？**

参考以前上课时的例子（参考刚才讲的懒加载的例子也行）

**16, Mybatis的一级、二级缓存**

参考项目（mybatis\_interview\_cache）

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

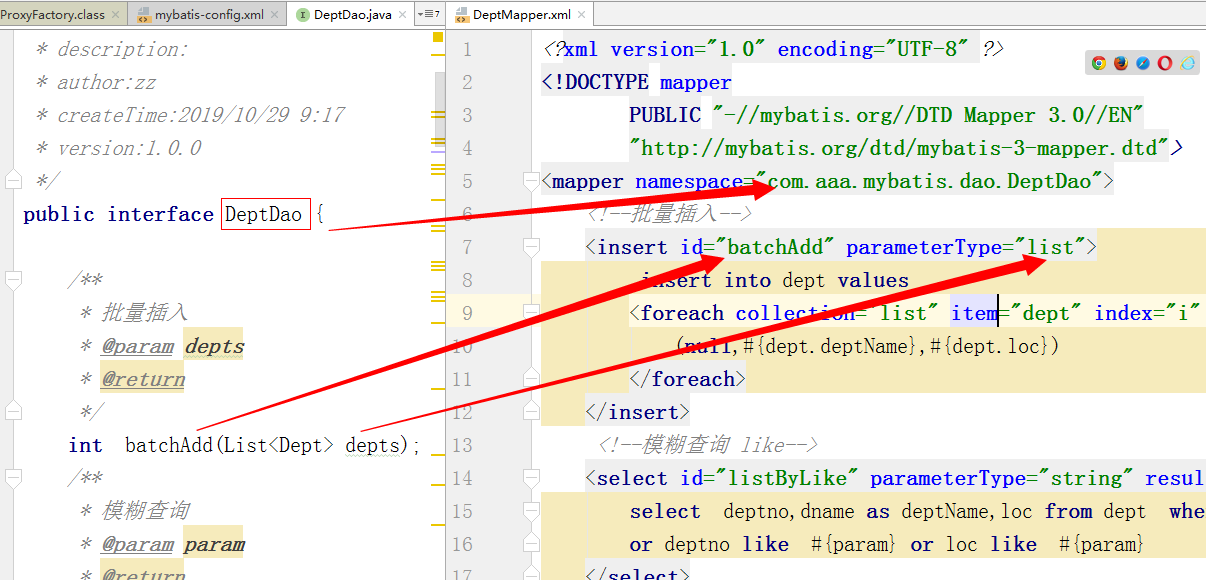
**17,使用MyBatis的mapper接口调用时有哪些要求？**

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(配置文件中可以省略，如果写，必须相同)；

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（通过resultMap也可以，更新方法没有返回值配置的）；

④  Mapper.xml文件中的namespace即是mapper接口的类路径。



**18, 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？**

**19,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。

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

<https://www.cnblogs.com/hopeofthevillage/p/11384848.html>

**20 Mybatis都有哪些Executor执行器？它们之间的区别是什么？**

答：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。

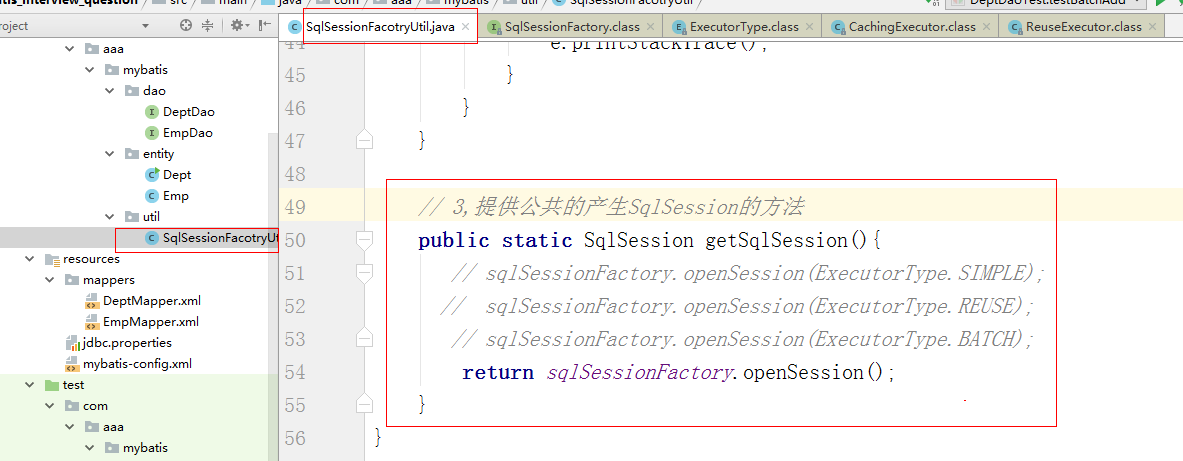
**21 Mybatis中如何指定使用哪一种Executor执行器？**

答：在Mybatis配置文件中，可以指定默认的ExecutorType执行器类型，也可以手动给DefaultSqlSessionFactory的创建SqlSession的方法传递ExecutorType类型参数。

// sqlSessionFactory.openSession(ExecutorType.SIMPLE);

// sqlSessionFactory.openSession(ExecutorType.REUSE);

// sqlSessionFactory.openSession(ExecutorType.BATCH);



**22 Mybatis是如何进行分页的？分页插件的原理是什么？**

开始值 = (第几页-1)\*每页显示数量

结束值= 第几页\*每页显示数量+1

mysql limit 开始值,每页显示数量

select \* from t where ... order by limit 开始值,每页显示数量

oracle rownum

方法1

select \* from

(select rownum rn,t.\* from t ) a

where a.rn>开始值 and a.rn<结束值；

方法2

select \* from

(select rownum rn,t.\* from t where rownmu<结束值） a

where a.rn>开始值 ；

        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。

mybatis分页插件（非官网）

1，添加jar包

2，添加springboot配置

https://pagehelper.github.io/

https://github.com/pagehelper/Mybatis-PageHelper/blob/master/README\_zh.md

#分页插件

#helperDialect属性来指定分页插件使用哪种方言

pagehelper.helper-dialect=oracle

#当该参数设置为 true 时，pageNum<=0 时会查询第一页， pageNum>pages（超过总数时），会查询最后一页。

pagehelper.reasonable=true

#支持通过 Mapper 接口参数来传递分页参数，默认值false，分页插件会从查询方法的参数值中，自动根据上面 params 配置的字段中取值，查找到合适的值时就会自动分页。

pagehelper.support-methods-arguments=true

#增加了该参数来配置参数映射，用于从对象中根据属性名取值

pagehelper.params=count=countSql

3，具体使用

与以前controller区别

//设置当前第几页和每页显示数量 PageHelper.startPage(Integer.valueOf(map.get("pageNo")+""),Integer.valueOf(map.get("pageSize")+""));

//用PageInfo对结果进行包装

PageInfo<Map> pageInfo =new PageInfo<Map>(newsService.getList());

/\*\*

\* 分页部门查询

\* @param map

\* @return

\*/

@ResponseBody

@RequestMapping("page")

public Object page(@RequestParam Map map){

int pageNo = Integer.valueOf(map.get("pageNo")+"");

int pageSize = Integer.valueOf(map.get("pageSize")+"");

//初始化配置

PageHelper.offsetPage(pageNo,pageSize);

PageInfo<Map> pageInfo = new PageInfo<Map>(deptService.getList());

//如果使用easyui可以这样封装,其他框架，自己根据pageInfo解析

Map tmap = new HashMap();

tmap.put("total",pageInfo.getTotal());

tmap.put("rows",pageInfo.getList());

return tmap;

}