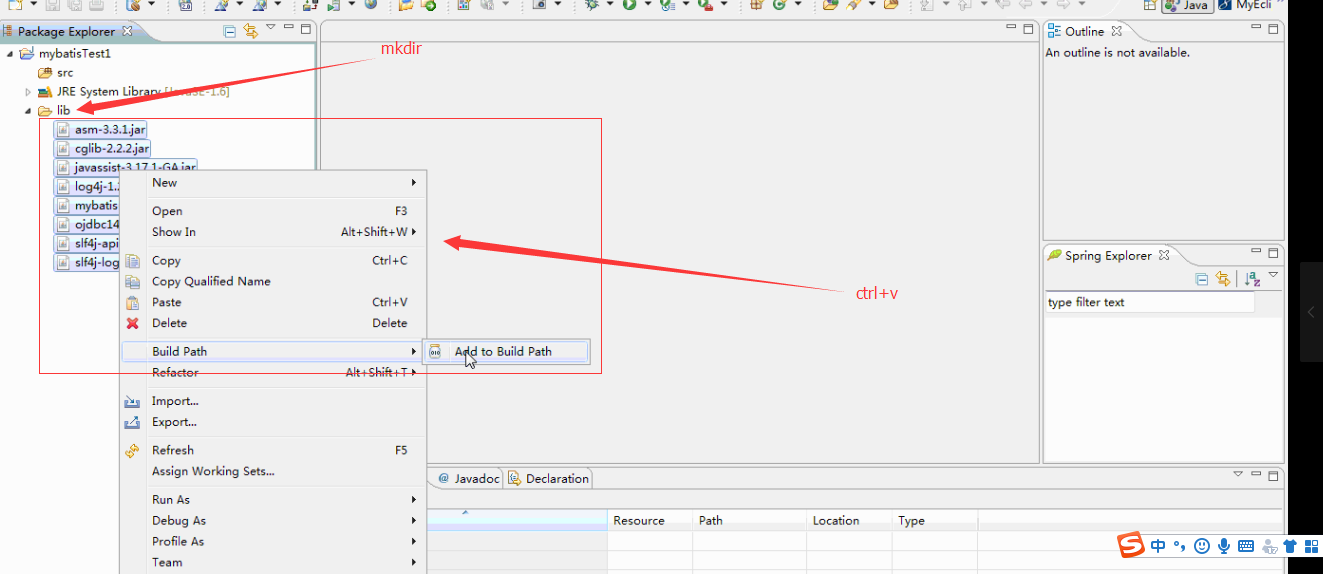
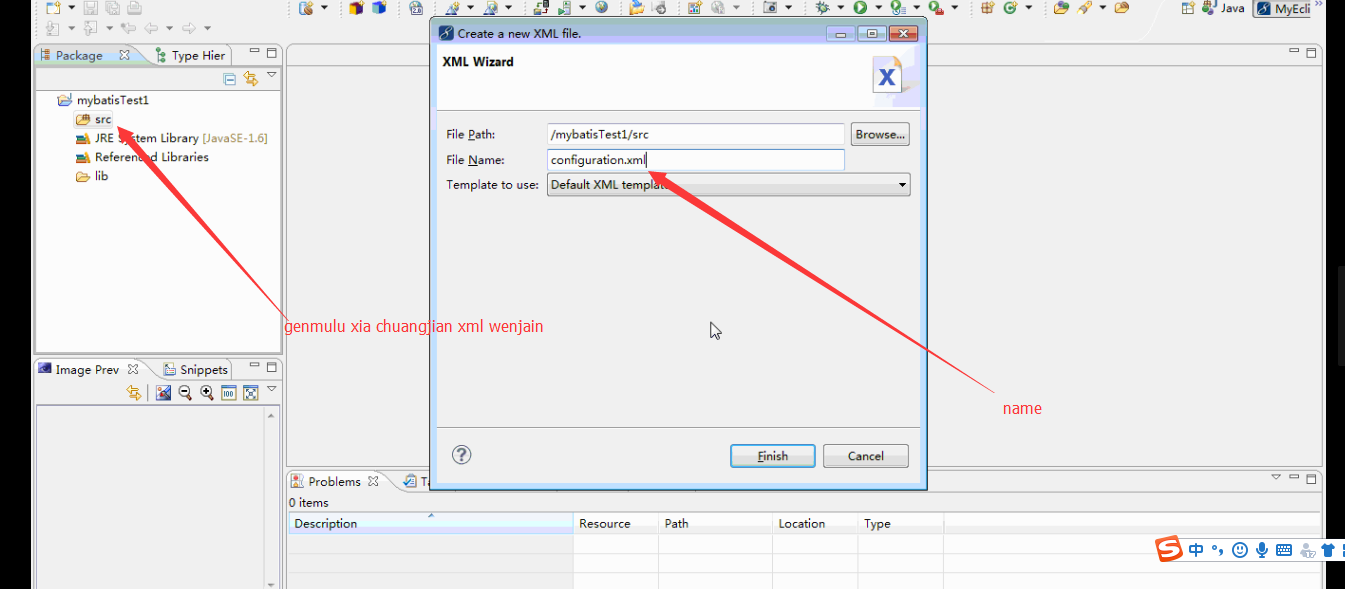
java项目的mybatis搭建

一、导入jar包



二、配置xml文件

1、创建xml文件(核心配置文件)



2、添加配置标签

（1）第二行添加以下字符串

<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

</configuration>

这个configuration是根标签，别写错

（2）配置数据源（必选）

说明default=*"myOracle"*和id=*"myOracle"*必须一样

<environments default=*"myOracle"*>

<environment id=*"myOracle"*>

<transactionManager type=*"jdbc"*/>

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"url"* value=*"jdbc:mysql://localhost:3306/test"*/>

<property name=*"username"* value=*"wl"*/>

<property name=*"password"* value=*"123456"*/>

</dataSource>

</environment>

</environments>

配置数据源结束，注意2个value的值

-----------------------------------------------------------

（3）configuration中声明一个mapper标签，

<!-- 配置一个另xml文件，这个文件和java中的一实体类对应，和数据库中的一张表对应 -->

<!-- 这个xml文件我们称作ORM（对象关系映射） -->

<!-- 这里只需要告诉mybatis框架这些xml文件的位置 -->

<mappers>

<mapper resource="com/bts/bean/EmpMapper.xml" />

</mappers>

（4）建立一个实体类对应的xml文件（映射文件）

com/bts/bean的路径下名称为EmpMapper.xml，与configuration

<mappers>标签的名称对应。

第二行粘贴次字符串

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

（5）功能实现

<mapper namespace="empmapper">

<!-- 编写一个查询功能 -->

<!-- id相当于方法名 resultType相当于返回值类型，注意需要写全类名-->

<select id="getEmps" resultType="com.bts.bean.Emp" >

<!-- 运行这个功能的sql语句写在这里 -->

select \* from emp

</select>

</mapper>

（5）注意事项

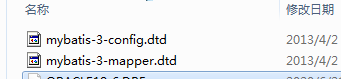
建立emp实体类时，属性名与表的字段名一致。

mybatis重要特点是把sql语句卸载xml里。

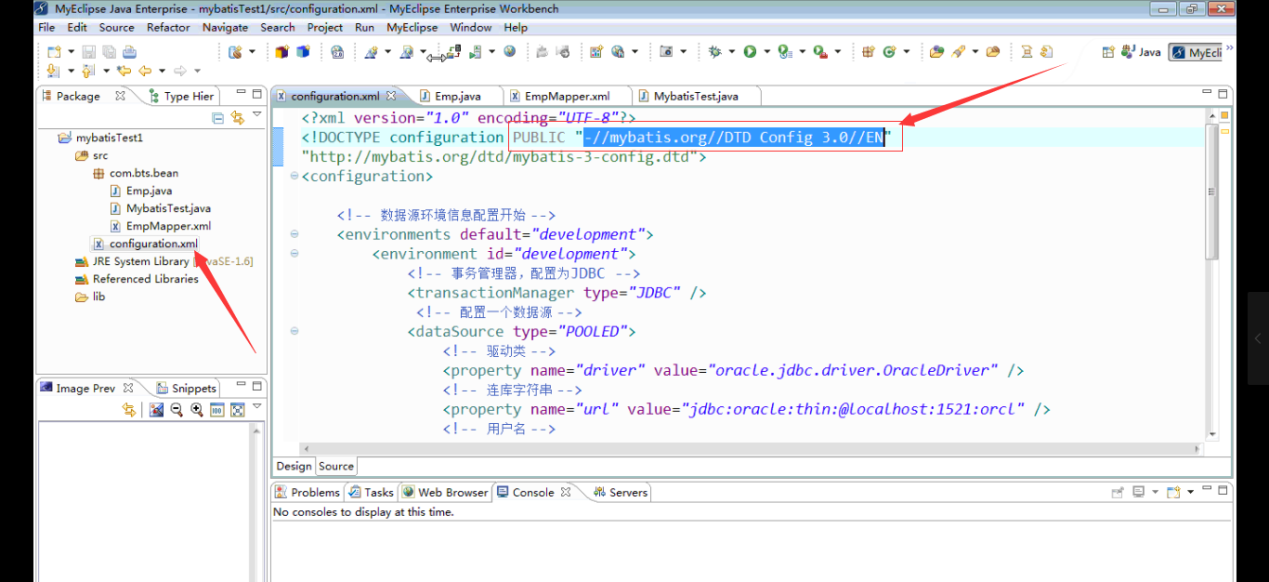
sql结尾不要写“；”

一、引入DTD文件

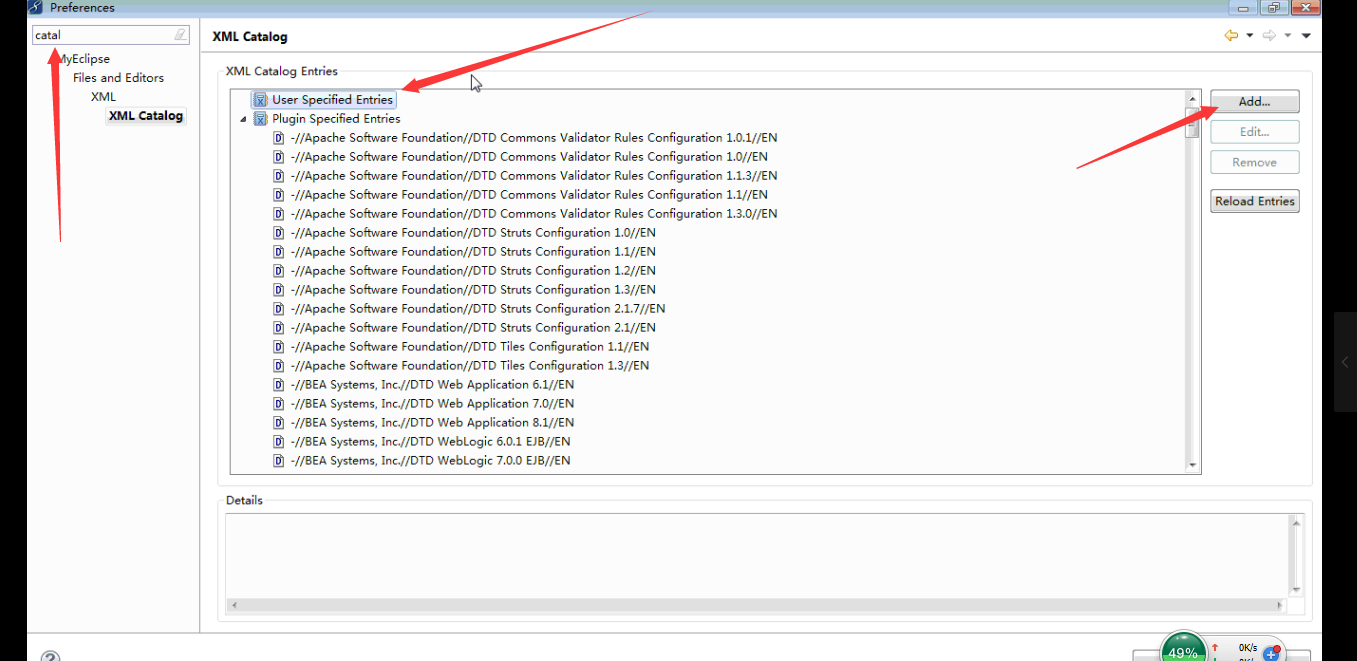
1.先准备好两个DTD文件

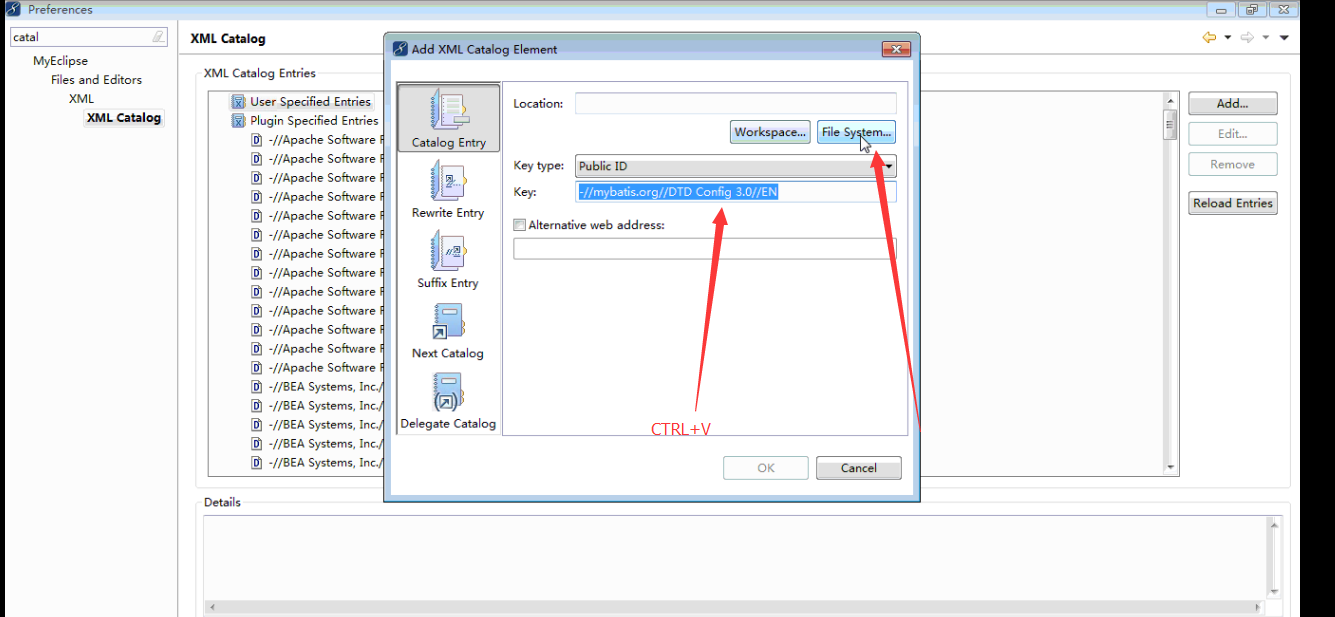


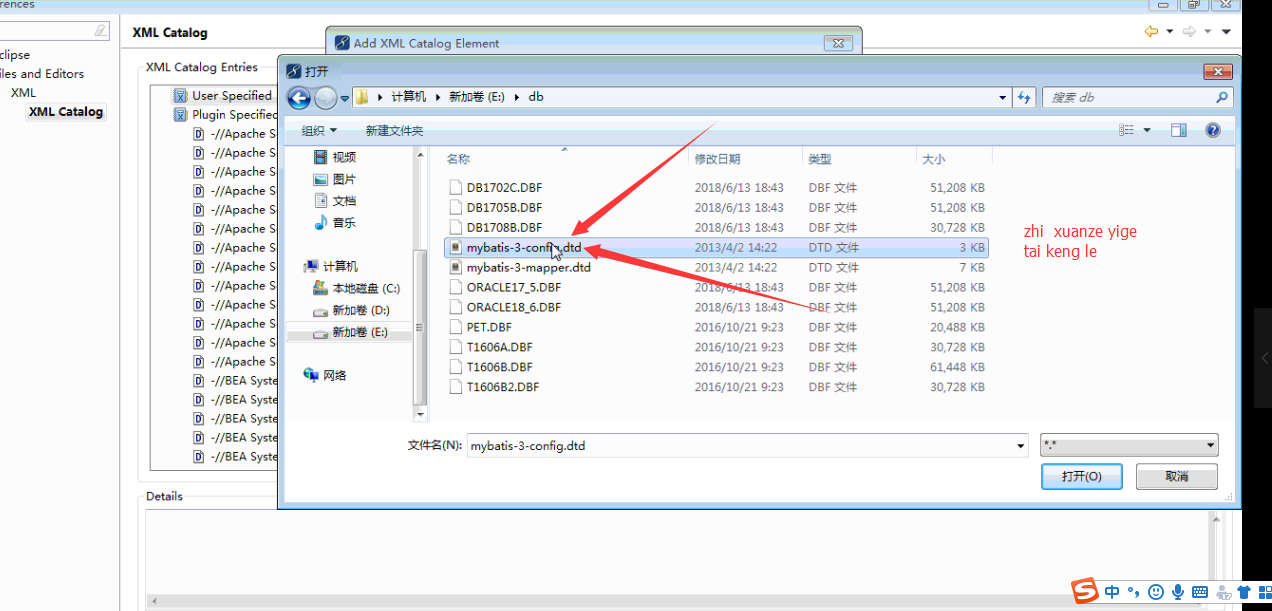
2.粘贴语句

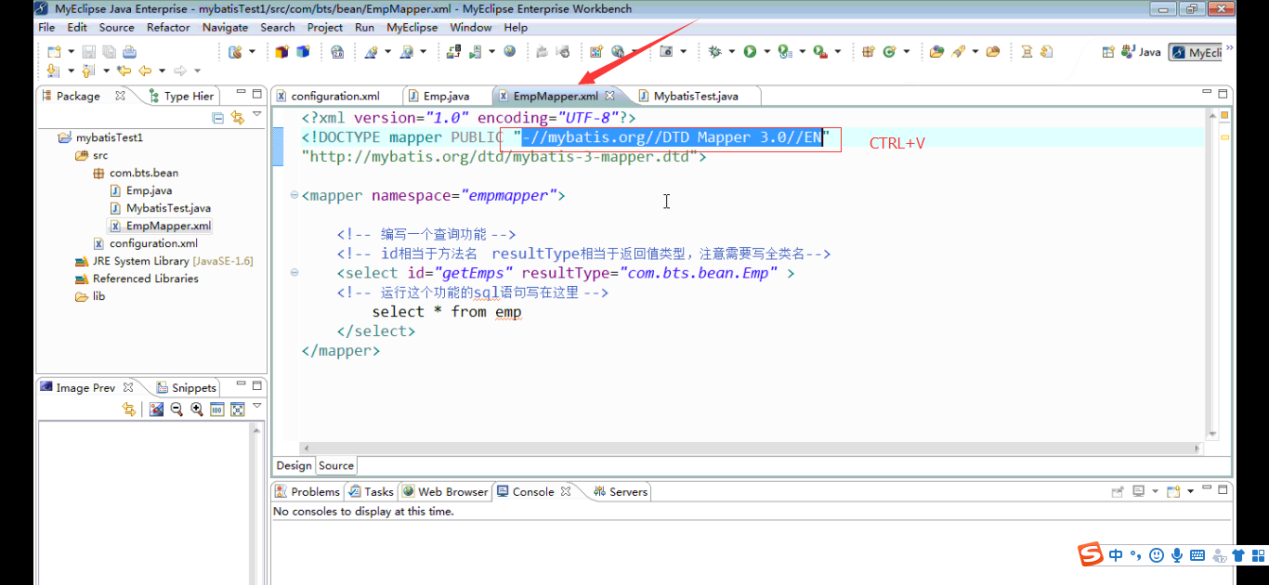


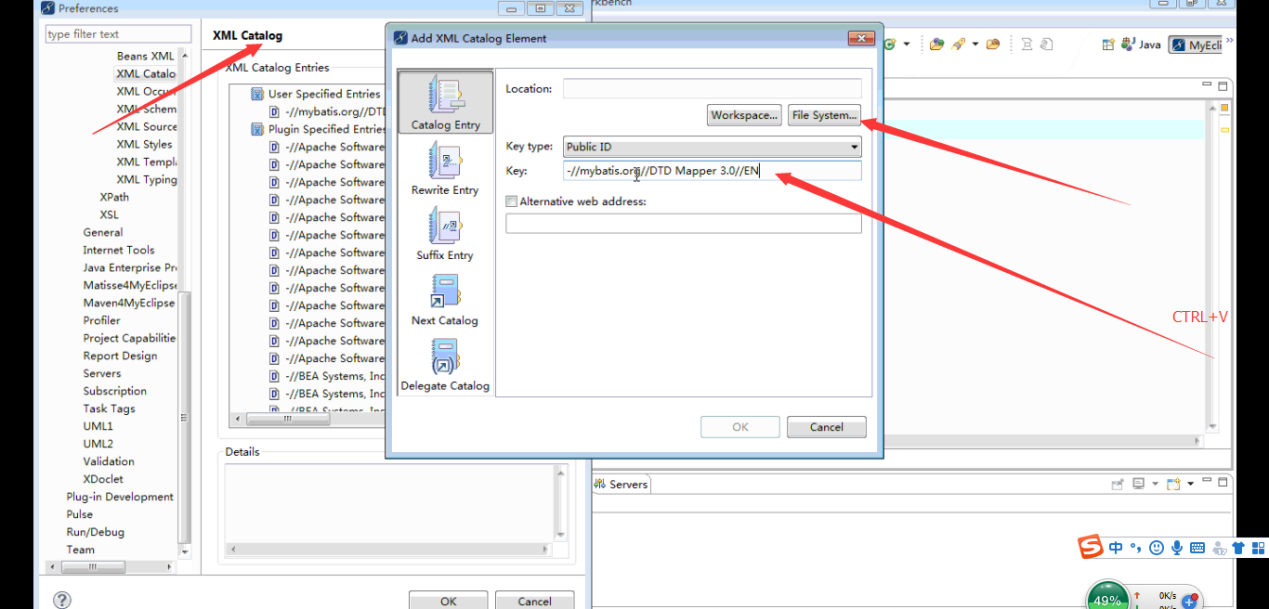
3.添加DTD文件，点击windows-preferences-catal











创建测试类

**public** **static** **void** main(String[] args) **throws** IOException {

/\*1.读取配置文件

2.创建sqlsessionfactory工厂

3.使用工厂生产sqlsession对象

4.调用查询方法

5.释放资源

\*/

//------------------------------------

//获得配置文件的流

Reader reader=Resources.*getResourceAsReader*("configurationl.xml");

//将这个流发送给mybatis框架让其解析

//创建SqlSessionFactory工厂（建设者模式）

SqlSessionFactoryBuilder builder= **new** SqlSessionFactoryBuilder();

//创建SqlSessionFactory工厂（工厂模式）

SqlSessionFactory factory=builder.build(reader);

//通过ssf对象制作一个sqlSession对象

//使用工厂生产SqlSession对象

SqlSession session= factory.openSession();

List<Emp> emps=session.selectList("getEmps");

**for** (Emp emp : emps) {

System.*out*.println(emp.getEmpid()+"--"+emp.getEmpname()+"--"+emp.getEmpname()+"--"+emp.getGender());

}

session.close();

reader.close();

}

----------------------------------------------------------------------

以上功能完成了一个简单的单表查询

在核心配置文件（configuration.xml文件中），引入以下标签

<typeAliases>

<!--设置默认包，前面按照实体类的包名写清楚,同时实体类中的mapper.xml应写类名 -->

<package name=*"com.bts.bean"*/>

</typeAliases>

在configuration.xml文件中加入<properties resource="dbsource.properties"></properties>

在src下建立资源文件dbsource. properties,并如下写明连接数据库的参数;

driver=oracle.jdbc.driver.OracleDriver

url=jdbc:oracle:thin:@localhost:1521:orcl

username=wl

pwd=123456

CRUD操作

参数书写语法为：#{属性名}

参数可以是int,如果有data数据类型，传入字符串需要转换to\_date（#{birthdate),'yyyy-MM-dd')}）

orcale自增用seq\_myseq.nextval;

模糊查询操作,动态查询

<select id=*"conSelectEmp"* parameterType=*"ConditionEmp"* resultType=*"Emp"*>

//注意参数，可以是string

<!-- 运行这个功能的sql语句写在这里 -->

select \* from emp where 1=1

<if test=*'gender!=null and gender!=""'*>

and gender=#{gender}//第一行加and

</if>

<if test=*'minsalary!=null and minsalary!=""'*>

and salary>=#{minsalary}

</if>

<if test=*'maxsalary!=null and minsalary!=""'*>

and salary&lt;=#{maxsalary}//看好转义符

</if>

</select>

动态更新：感觉没啥用，必须得改一个

批量删除，参数是数组，返回值是删除的条数

<delete id=*"deleteMoneyEmps"*>

delete emp where empid in

<foreach collection=*"array"* index=*"index"* item=*"item"* open=*"("* separator=*","* close=*")"*>

#{item}

</foreach>

</delete>

参数是map,以后不用建立没有必要的对象了，id,name 为map的键

<select id=*"loginEmp"* parameterType=*"map"* resultType=*"Emp"*>

select \* from emp where empid=#{id} and empname=#{name}

</select>

------------------------------------------------------------------------------`

**多表联查：（select标签中,resultMap属性值与resultMap标签id属性值必须相同，association 中property属性值是关联类的属性名 属性值javaType是相对应的类名，column是表的字段名，property是类的属性名）**

**多对一的映射**

<resultMap type=*"Emp"* id=*"empWithDept"*>

<id column=*"empid"* property=*"empid"*/>

<result column=*"empname"* property=*"empname"*/>

<result column=*"gender"* property=*"gender"*/>

<result column=*"job"* property=*"job"*/>

<result column=*"birthday"* property=*"birthday"*/>

<result column=*"salary"* property=*"salary"*/>

<association property=*"dept"* javaType=*"Dept"*>

<id column=*"deptid"* property=*"deptid"*/>

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

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

</association>

</resultMap>

<select id=*"selectEmpWithDept"* resultMap=*"empWithDept"*>

select \* from emp a inner join dept b on a.deptid=b.deptid

</select>

**一对多的映射**

<resultMap type=*"Dept"* id=*"DeptWithEmp"*>

<id column=*"deptid"* property=*"deptid"*/>

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

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

//optype属性值是泛型类型

<collection property=*"emps"* ofType=*"Emp"*>

<id column=*"empid"* property=*"empid"*/>

<result column=*"empname"* property=*"empname"*/>

<result column=*"gender"* property=*"gender"*/>

<result column=*"job"* property=*"job"*/>

<result column=*"birthday"* property=*"birthday"*/>

<result column=*"salary"* property=*"salary"*/>

</collection >

</resultMap>

<select id=*"getDeptWithEmp"* resultMap=*"DeptWithEmp"*>

select \* from dept a inner join emp b on a.deptid=b.deptid order by a.deptname

</select>

关系表只有“两列”，每列都是“外键”；  
<select id="getStudentWithSubject" resultMap="studentWithSubject">

select \* from student s inner join guanxi g on s.studentno=g.studentno

inner join subject sub on sub.subjectid=g.subjectid

</select>

<resultMap type="Student" id="studentWithSubject">

<id column="studentno" property="studentno"/>

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

<collection property="subjects" ofType="Subject">

<id column="subjectid" property="subjectid"/>

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

</collection>

</resultMap>