# Mybatis 环境配置与使用

## 环境配置与简单使用

1. 在maven pom.xml文件中移入引入 mybatis log4j mysql驱动
2. 创建mybatis的全局配置文件 mybatis-config.xml 写入数据库连接配置

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

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

<configuration>

<environments default=*"development"*>

<environment id=*"development"*>

<transactionManager type=*"JDBC"*/>

<dataSource type=*"POOLED"*>

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

<property name=*"url"* value=*"jdbc:mysql://localhost:3306/mybatis?useUnicode=true&amp;characterEncoding=UTF-8"*/>

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

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

</dataSource>

</environment>

</environments>

<!-- 注册mapper -->

<mappers>

<mapper resource=*"EmpolyeeMapper.xml"*/>

</mappers>

</configuration>

1. 创建log4j.xml文件配置需要输出的日志信息
2. 创建实体类对应数据表
3. 创建与实体类对应的\*mapper.xml文件 \*与实体类的名称一致

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

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

<mapper namespace=*"com.wanggang.mybatis.EmpolyeeMapper"*>

<select id=*"selectEmpolyee"* resultType=*"model.Empolyee"*>

select

id,

last\_name lastName,

email,

gender

from tbl\_empolyee where id = #{id}

</select>

</mapper>

1. 在全局配置文件中注册\*mapper.xml文件

<!-- 注册mapper -->

<mappers>

<mapper resource=*"EmpolyeeMapper.xml"*/>

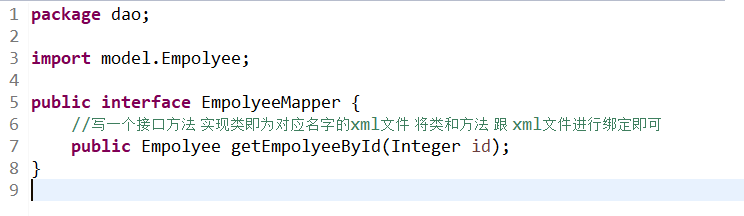
</mappers>

8.

1. 得到全局配置文件的IO流,并使用该IO流创建SqlSessionFactory对象sqlSessionFactory
2. 使用sqlSessionFactory.openSession()打开SqlSession,并获得该对象 openSession
3. 使用openSession.select(“\*mapper.xml中的配置方法”,”传入参数”)执行\*mapper.xml文件中写的方法得到指定所需返回值的对象
4. 关闭openSession

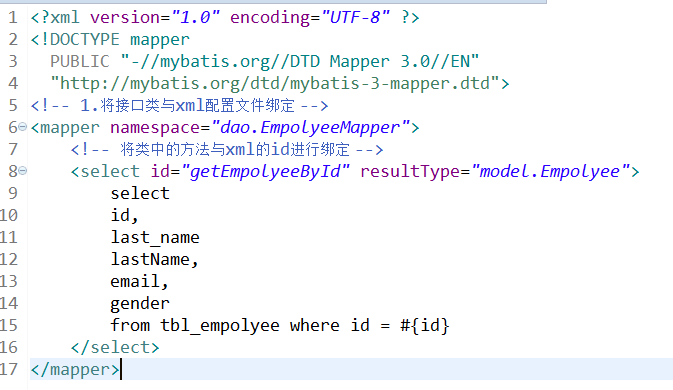
## 使用接口方式(开发中使用)

1. 创建接口文件 \*Mapper.java 并写入需要的方法



1. 创建与接口文件对应的\*Mapper.xml 文件 并注册相关的方法与值对应

此时 \*Mapper.xml的nameSpace为接口文件的全类名,配置的相关查询方法的ID为接口里面的方法名字 返回值为java全类名



1. 使用中通过接口openSession对象拿到mapper对象并调用接口中的对应方法



## 使用注解的方式代替\*mapper.xml(非常简单的方法用这种)

1. 创建\*MapperAnnotation的接口文件
2. 指定方法并在方法上注解 增删改查的对应注解
3. 并在注解上写入要执行的sql语句
4. 在全局配置文件中注册该接口

<!-- 1.注册 xml 该xml中与类绑定 -->

<mapper resource=*"mapper/EmpolyeeMapper.xml"* />

<!-- 2.注册注解过的接口文件-->

<mapper class=*"dao.EmpolyeeMapperAnnotation"*/>

</mappers>

1. 不用写\*Mapper.xml
2. 使用方法同 接口方式

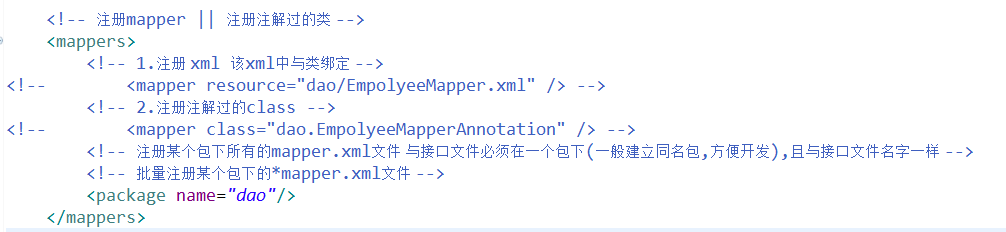
## 重要配置

Mapper的注册三种形式

1. 注册\*Mapper.xml
2. 注册注解过的接口文件
3. 注册整个包下的所有文件

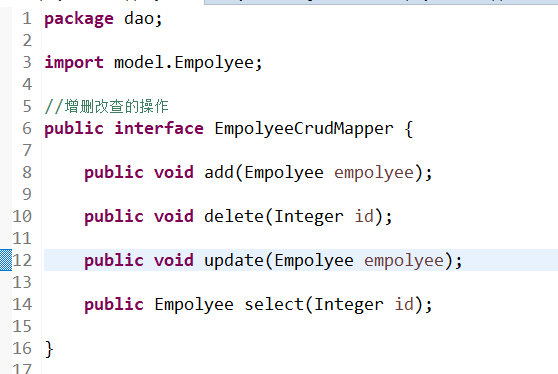
\*\*\*接口文件必须与\*Mapper.xml文件同名

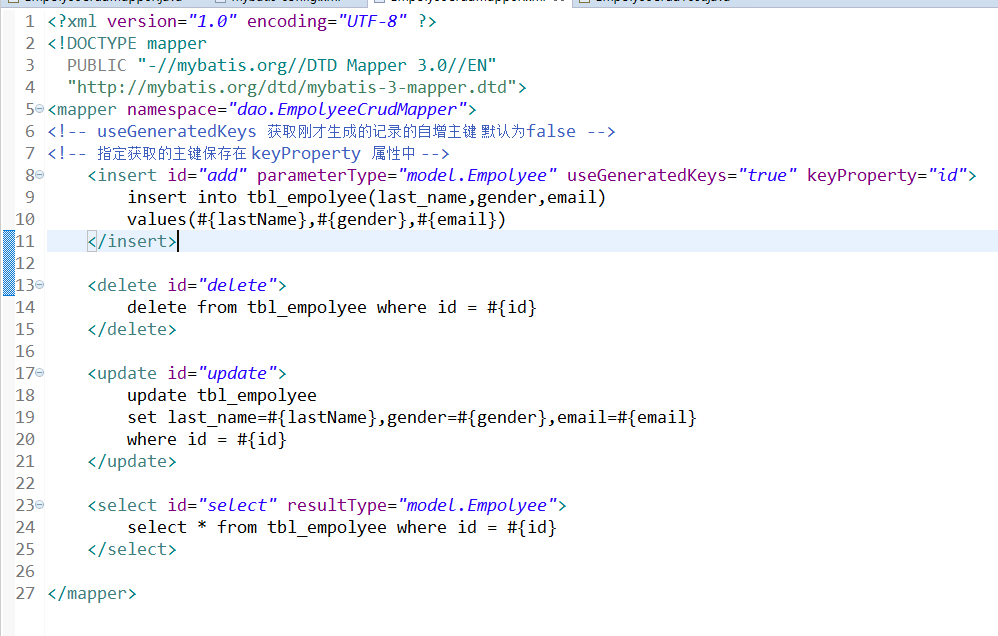
\*\*\*两个文件必须在一个包下(通过创建不同的源代码文件 和相同的包名在编译器中分开,输出其实还是在一起来实现便于开发)



## 增删查改

1. 创建接口文件 并写入相关的方法
2. 创建与接口文件同名的\*Mapper.xml文件并指定nameSpqce为接口文件全类名,方法的ID闻接口文件对应的方法名,returnType为接口文件方法的返回值类型
3. 在全局配置文件中注册整个包下的\*Mapper.xml文件
4. 使用同上





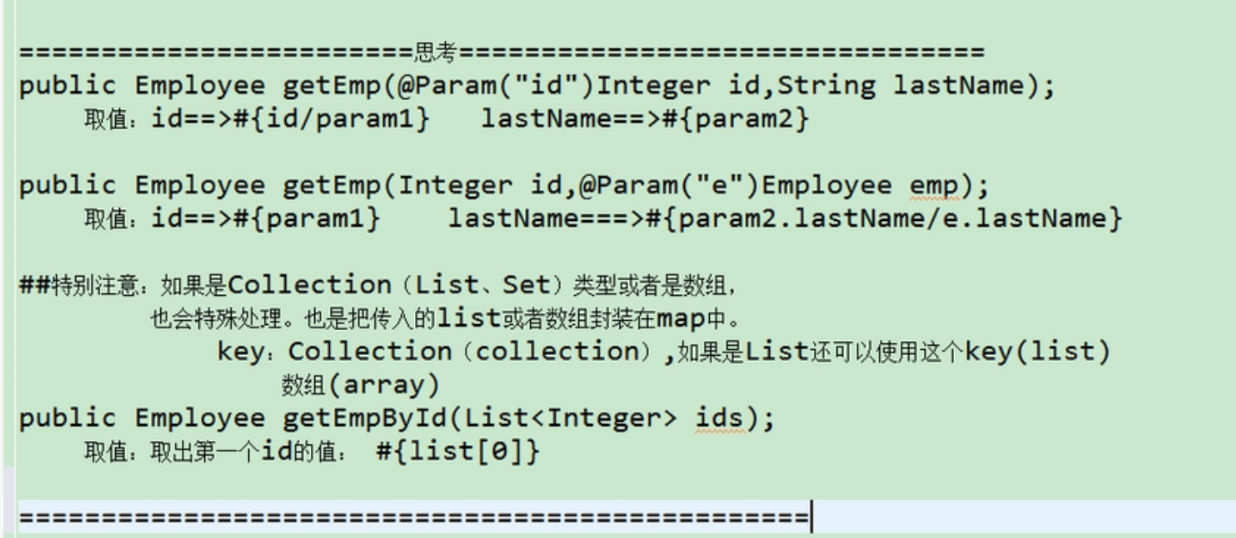
## Mybatis对于传入参数的处理

1. 一个参数:不做任何处理 直接使用#{参数名}进行取值
2. 多个参数:参数会被 封装成 一个Map
3. 取值使用 #{param1} #{param2}
4. 如果不想让mybatis封装为Map,则在接口方法中写@Param注解 指定传入参数的名字

public Empolyee manyParams(@Param("id") Integer id,@Param("lastName")String lastName);

1. 如果参数非常多,传入一个POJO类对象,直接使用#{名字}取值,支持级联对象
2. 也可以自己封装一个Map进行传入 ,则方法的参数也为对应Map map,自己new Map进行put键值再传入
3. 一般经常使用的一些参数可以封装为一个POJO,不是经常使用的则临时new Map再传入

## Mybatis 对于不同的方式传入参数(思考)



## #{} 和 ${}

#{}用于取值 类似PreparedStatement的设置预编译

${}类似字符串拼接

## Mybatis的返回值相关

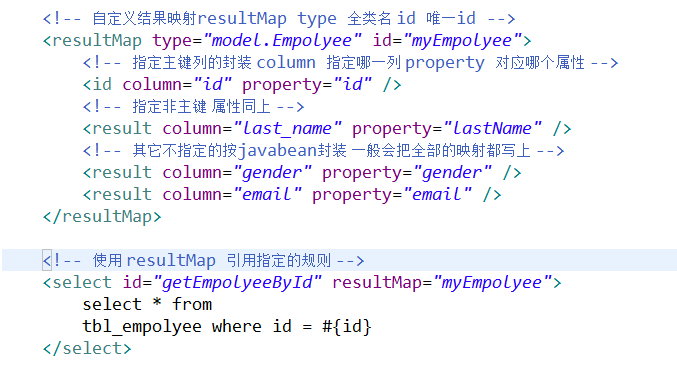
1. ReturnType

\*如果是对象,则为对象的全类名

\*如果是集合类型 ,则为集合里面的装的对象的类型全类名

## 入参和返回结果映射ResultMap

数据库字段与实体类字段进行自定义的映射,并在\*Mapper,xml写方法时进行引用



支持级联属性的ResultMap指定



## 使用association进行级联属性查询



## 使用association进行分步查询



## 动态sql if where choose trim set foreach include

Where :省略sql语句中的 where 条件关键词 避免产生sql语句拼接错误

If 其中 test属性中写判断表达式 注意实体的转换

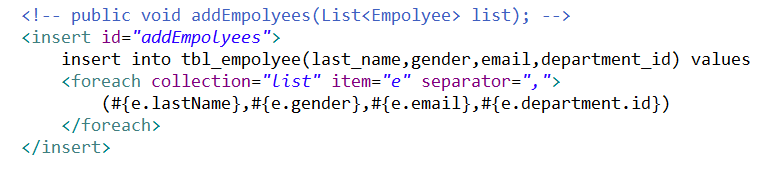
Choose 其中 when子标签中有test属性,写表达式

Trim 过滤或者拼接sql



Set 更新操作中拼接sql的关键字标签 结合 <if test=””>进行简单的sql拼接操作

Foreach 当入参为一个集合时,比如:一个包含id的集合,需要全部查出,则使用foreach进行



Include 引入一段使用sql标签设定好的sql语句

