### 导包

这里用Maven的方式

<!-- 配置mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.1</version>

</dependency>

<!-- 配置mysql驱动 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.37</version>

</dependency>

<!-- 配置log4j日志文件，该包需要一个配置文件，复制即可 -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

Log4j.xml配置文件在类路径下

### 全局配置文件

文件中描述了数据库连接的基本信息，和映射sql配置文件等信息

<?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">

<!-- 全局配置文件 -->

<configuration>

<!-- 这里为数据库连接的基本信息 -->

<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:///mybatis"* />

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

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

</dataSource>

</environment>

</environments>

<!-- **这里为sql语句映射文件的url，如果在工程路径或其文件夹下，直接写文件名，如果在包下则包名 / 文件名** -->

<mappers>

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

</mappers>

</configuration>

### Sql语句配置文件

Mybatis规定，必须把所有的sql语句和sql规则全部放在了配置文件中，写上唯一标识id，用时直接获取即可

<?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=*"mybatis.EmployeeMapper"*>

<!-- **namespace：命名空间 ，指定为接口的全类名，可以自动绑定到接口**

**id：sql语句的唯一标识，一般为接口对应方法名**

**resultType：返回值类型**

**parameterType：参数类型**

**#{id}：从传递过来的参数中取出id值**

-->

<select id=*"selectEmpl"* parameterType=*"int"* resultType=*"entities.Employee"*>

select id id,last\_name lastName,gender gender,email email from tbl\_employee where id = #{id}

</select>

</mapper>

### 执行操作

**// 1. 通过全局配置文件获取SqlSessionFactory工厂**

//全局配置文件的url，如果在工程路径或其文件夹下，直接写文件名，如果在包下则包名 / 文件名

String resource = "mybatis-config.xml";

//用流的方式获取全局配置文件

Reader reader = Resources.*getResourceAsReader*(resource);

//获取SqlSessionFactory工厂

SqlSessionFactory sqlMapper = **new** SqlSessionFactoryBuilder().build(reader);

//2. **通过SqlSessionFactory工厂获取SqlSession对象，可以直接执行已经映射过的sql语句**

// 一个SqlSession对象就是数据库中的一次会话，用完关闭

SqlSession openSession = sqlMapper.openSession();

**try** {

**// 3. 执行sql**

//查询一个对象，第一个参数为sql语句在映射文件中的唯一标示，为了防止冲突，写成命名空间 . id

Employee employee = openSession.selectOne("mybatis.EmployeeMapper.selectEmpl", 1);

System.***out***.println(employee);

} **finally** {

**//关流**

openSession.close();

}

### 改进版本

用接口的方法进行自动绑定，并且解耦

#### 定义一个接口

**public** **interface** EmployeeMapper {

Employee getEmployee(Integer id);

}

#### 把sql映射文件和接口关联

①在sql映射文件中，把namespace命名空间写为接口全类名

<mapper namespace=*"dao.EmployeeMapper"*>

②方法对应的sql语句映射的id为方法名

<select id=*"getEmployee"* parameterType=*"int"* resultType=

*"entities.Employee"*>

select id id,last\_name lastName,gender gender,email email from tbl\_employee where id = #{id}

</select>

#### 测试

**// 1. 读取配置文件，获取SqlSessionFactory工厂**

String resource = "mybatis-config.xml";

Reader resourceAsReader = Resources.*getResourceAsReader*(resource);

SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(resourceAsReader);

**// 2. 获取SqlSession对象**

SqlSession openSession = sqlSessionFactory.openSession();

**// 3. 调用getMapper方法，传入接口的Class对象，使用动态代理生成一个接口的实现对象**

EmployeeMapper mapper = openSession.getMapper(EmployeeMapper.**class**);

**// 4. 调用实现对象方法，实现操作**

Employee employee = mapper.getEmployee(1);

System.***out***.println(employee);