**实验二： MyBatis框架应用**

## 实验内容：

现有一张员工表如下。利用本章所学知识完成员工管理系统。实现如下功能：根据id查询员工信息、新增员工信息、根据id修改员工信息、根据id删除员工信息。



## 实验代码

**建表SQL语句**

create table employee  
(  
 id int auto\_increment primary key,  
 name varchar(100) null,  
 age int null,  
 position varchar(100) null,  
 constraint employee\_id\_uindex unique (id)  
);

**Employee.java**

@Data  
@AllArgsConstructor  
@NoArgsConstructor  
public class Employee *{* private Integer id;  
 private String name;  
 private Integer age;  
 private String position;  
*}*

MyBatisUtils

MyBatisUtils.java

package com.shf.utils;  
  
import org.apache.ibatis.io.Resources;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.ibatis.session.SqlSessionFactory;  
import org.apache.ibatis.session.SqlSessionFactoryBuilder;  
  
import java.io.IOException;  
import java.io.InputStream;  
  
public class MyBatisUtils *{* private static SqlSessionFactory *sqlSessionFactory*;  
  
 static *{* try *{* String resource = "mybatis-config.xml";  
 InputStream inputStream = Resources.*getResourceAsStream(*resource*)*;  
 *sqlSessionFactory* = new SqlSessionFactoryBuilder*()*.build*(*inputStream*)*;  
 *}* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
 }* public static SqlSession getSqlSession*() {  
// 自动提交事物* return *sqlSessionFactory*.openSession*(*true*)*;  
 *}  
}*

EmployeeMapper.xml

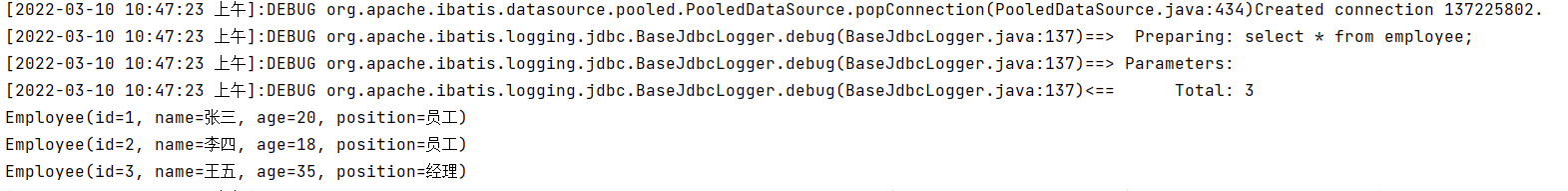
*<?*xml version="1.0" encoding="UTF-8" *?>  
<!DOCTYPE* mapper  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>  
  
<*mapper namespace="com.shf.mapper.EmployeeMapper"*>  
 <*select id="selectEmployeeList" resultType="employee"*>* select \*  
 from employee;  
 *</*select*>  
  
 <*select id="selectEmployeeById" resultType="employee" parameterType="int"*>* select \*  
 from employee  
 where id = #{id};  
 *</*select*>  
  
 <!-- 添加用户-->  
 <*sql id="addKey"*>  
 <*trim suffixOverrides=","*>  
 <*if test="id!=0"*>* id,  
 *</*if*>  
 <*if test="name!=null"*>* name,  
 *</*if*>  
 <*if test="age!=0"*>* age,  
 *</*if*>  
 <*if test="position!=null"*>* position,  
 *</*if*>  
 </*trim*>  
 </*sql*>  
 <*sql id="addValue"*>  
 <*trim suffixOverrides=","*>  
 <*if test="id!=0"*>* #{id},  
 *</*if*>  
 <*if test="name!=null"*>* #{name},  
 *</*if*>  
 <*if test="age!=0"*>* #{age},  
 *</*if*>  
 <*if test="position!=null"*>* #{position},  
 *</*if*>  
 </*trim*>  
 </*sql*>  
 <*insert id="addEmployee" parameterType="employee"*>* insert into employee (*<*include refid="addKey"*/>)* values *(<*include refid="addValue"*/>)*;  
 *</*insert*>  
  
  
 <*update id="updateEmployee" parameterType="employee"*>* update employee  
 *<*set*>  
 <*if test="id!=0"*>* id=#{id},  
 *</*if*>  
 <*if test="name!=null"*>* name=#{name},  
 *</*if*>  
 <*if test="age!=0"*>* age=#{age},  
 *</*if*>  
 <*if test="position!=null"*>* position=#{position},  
 *</*if*>  
 </*set*>* where id=#{id};  
 *</*update*>  
  
 <*delete id="deleteEmployeeById" parameterType="int"*>* delete  
 from employee  
 where id = #{id};  
 *</*delete*>  
</*mapper*>*

EmployeeTest.java

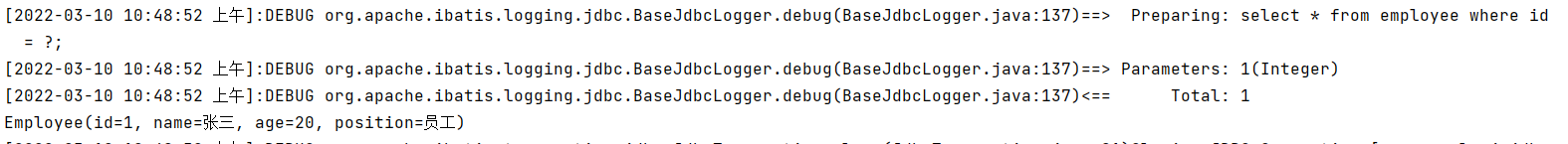
package com.shf.test;  
  
import com.shf.mapper.EmployeeMapper;  
import com.shf.pojo.Employee;  
import com.shf.utils.MyBatisUtils;  
import org.apache.ibatis.session.SqlSession;  
import org.junit.Test;  
  
import java.util.List;  
  
public class EmployeeTest *{* @Test  
 public void selectEmployeeList*() {* SqlSession sqlSession = MyBatisUtils.*getSqlSession()*;  
 EmployeeMapper mapper = sqlSession.getMapper*(*EmployeeMapper.class*)*;  
 List*<*Employee*>* employees = mapper.selectEmployeeList*()*;  
 for *(*Employee employee : employees*) {* System.*out*.println*(*employee*)*;  
 *}* sqlSession.close*()*;  
 *}* @Test  
 public void selectEmployeeById*() {* SqlSession sqlSession = MyBatisUtils.*getSqlSession()*;  
 EmployeeMapper mapper = sqlSession.getMapper*(*EmployeeMapper.class*)*;  
 Employee employee = mapper.selectEmployeeById*(*1*)*;  
 System.*out*.println*(*employee*)*;  
  
 sqlSession.close*()*;  
 *}* @Test  
 public void addEmployee*() {* SqlSession sqlSession = MyBatisUtils.*getSqlSession()*;  
 EmployeeMapper mapper = sqlSession.getMapper*(*EmployeeMapper.class*)*;  
  
 Employee employee = new Employee*(*5, "张帅", 39, "总监"*)*;  
 mapper.addEmployee*(*employee*)*;  
  
 sqlSession.close*()*;  
 *}* @Test  
 public void updateEmployee*() {* SqlSession sqlSession = MyBatisUtils.*getSqlSession()*;  
 EmployeeMapper mapper = sqlSession.getMapper*(*EmployeeMapper.class*)*;  
  
 Employee employee = new Employee*(*5, "张帅new", 93, "总监new"*)*;  
 mapper.updateEmployee*(*employee*)*;  
  
 sqlSession.close*()*;  
 *}* @Test  
 public void deleteEmployeeById*() {* SqlSession sqlSession = MyBatisUtils.*getSqlSession()*;  
 EmployeeMapper mapper = sqlSession.getMapper*(*EmployeeMapper.class*)*;  
  
 mapper.deleteEmployeeById*(*5*)*;  
  
 sqlSession.close*()*;  
 *}  
}*

## 实验截图：

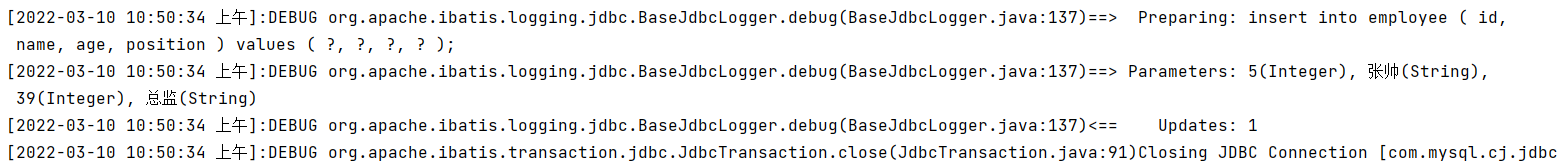
查询所有员工

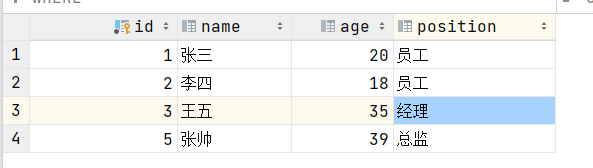


根据id查询员工信息

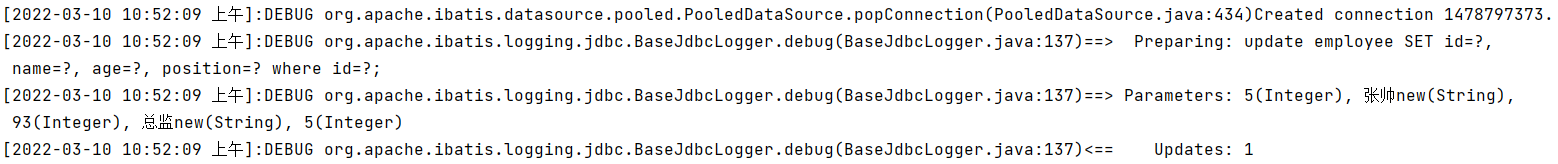


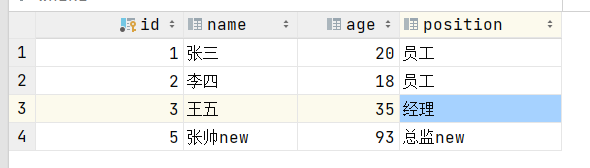
添加员工



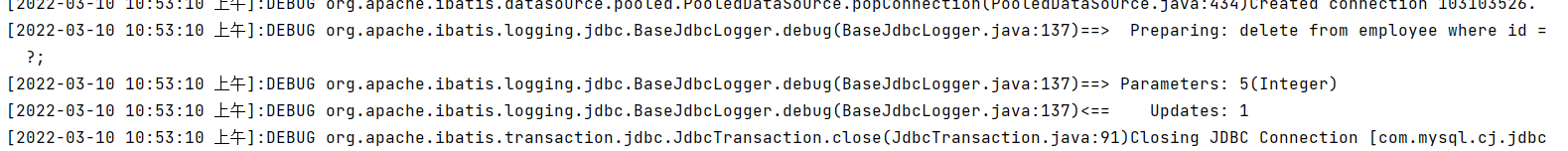


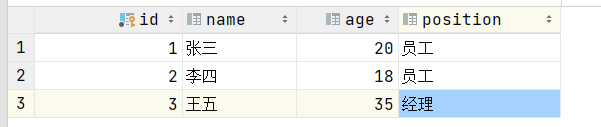
根据Id修改员工信息





根据id删除员工信息





## 实验小结：

通过本次实验我学会使用mybatis进行简单的CRUD操作,并把SqlSessionFactory封装成工具类，简化代码开发，使用log4j把日志输出到控制台，观察mybatis给我们拼接的sql语句，并把日志存储到本地磁盘上。