

**学 生 实 验 报 告**

**（理工类）**



课程名称： 企业级java框架编程 专业班级： 15软件工程（Z）

学生学号： 1512001066 学生姓名： 吴跟强

所属院部： 软件工程学院 指导教师： 王勇/桂文明

**20 17 ——20 18 学年 第 2 学期**

金陵科技学院教务处制

**实验报告书写要求**

实验报告原则上要求学生手写，要求书写工整。若因课程特点需打印的，要遵照以下字体、字号、间距等的具体要求。纸张一律采用A4的纸张。

**实验报告书写说明**

实验报告中一至四项内容为必填项，包括实验目的和要求；实验仪器和设备；实验内容与过程；实验结果与分析。各院部可根据学科特点和实验具体要求增加项目。

**填写注意事项**

（1）细致观察，及时、准确、如实记录。

（2）准确说明，层次清晰。

（3）尽量采用专用术语来说明事物。

（4）外文、符号、公式要准确，应使用统一规定的名词和符号。

（5）应独立完成实验报告的书写，严禁抄袭、复印，一经发现，以零分论处。

**实验报告批改说明**

实验报告的批改要及时、认真、仔细，一律用红色笔批改。实验报告的批改成绩采用百分制，具体评分标准由各院部自行制定。

**实验报告装订要求**

实验批改完毕后，任课老师将每门课程的每个实验项目的实验报告以自然班为单位、按学号升序排列，装订成册，并附上一份该门课程的实验大纲。

实验项目名称： Spring 实验学时： 4

同组学生姓名： 实验地点： 北辅311

实验日期： 2018.04.17 实验成绩：

批改教师： 批改时间：

1. 实验目的

1.1 掌握Spring框架的建立步骤；

1.2 掌握Spring的Ioc注入方式：传值注入和构造注入；

1.3 掌握Spring的静态工厂注入方式；

1.4 掌握Spring的实例工厂注入方式。

1. 实验要求

2.1 根据实验步骤中提出的要求，使用Eclipse编写相应的Java程序；

2.2 根据实验步骤中提出的要求，使用Eclipse编译、运行自己所编写的Java程序；

2.3 根据编译与运行过程中所获得的错误信息修改程序直至获得正确的结果；

2.4 记录实验中遇到的各类问题并以及解决办法。

1. 实验环境、设备

3.1 开发平台：Eclipse Neon3

3.2 操作系统：Window 7及以上

3.3 数据库：MySQL5.1

1. 实验代码
2. UserDao.java

public interface UserDao {

public boolean logon(String username, String password);

}

1. UserDaoImpl.java

@Component

public class UserDaoImpl implements UserDao{

private boolean flag = false;

@Override

public boolean logon(String username, String password) {

if(username.equalsIgnoreCase("erwin")&&password.equals("123")){

this.flag = true;

}

return flag;

}

}

1. UserService.java

@Component("us12")

public class UserService {

protected final transient Log log = LogFactory.getLog(UserService.class);

@Resource

private UserDao userDao;

public boolean check(String u, String p){

log.info("check user invoke!");

return this.userDao.logon(u, p);

}

}

1. UserServiceTest.java

public class UserServiceTest {

private ApplicationContext context;

private UserService us;

/\*\*

\* @throws java.lang.Exception

\*/

@Before

public void setUp() throws Exception {

}

/\*\*

\* @throws java.lang.Exception

\*/

@After

public void tearDown() throws Exception {

}

@Test

public void testShowUsers() {

this.context =

new ClassPathXmlApplicationContext("com\\spring\\demo012\\beans.xml");

us = (UserService) context.getBean("us12");

System.out.println("jiaoyan: "+us.check("erwin", "123"));

}

}

1. Bean.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.3.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-4.3.xsd">

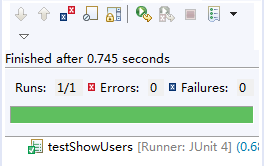
<context:annotation-config/>

<context:component-scan base-package="com.spring.demo012"/>

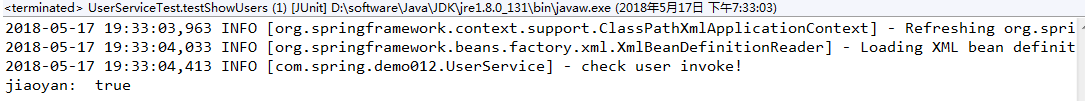
</beans>

1. 实验结果与分析

Junit测试结果：



运行结果：



1. 实验心得

使用maven学习spring技术，让我懂得了很多，明白了小细节对于项目的成败的重要性，关键性文件Bean.xml，了解spring项目，最主要的就是了解这个配置文件，这个文件决定着项目的success。发现自己由于基础不牢，有些知识点还需要反复记忆，我还要更加努力。

实验项目名称： Spring-MVC 实验学时： 4

同组学生姓名： 实验地点： 北辅311

实验日期： 2018.04.20 实验成绩：

批改教师： 批改时间：

1. 实验目的

1.1 理解Spring MVC模型

1.2 掌握Spring MVC的表单数据传递与接收方法；

* 1. 掌握Spring MVC的表单校验方法；

1. 实验要求

2.1 根据实验步骤中提出的要求，使用Eclipse编写相应的Java程序；

2.2 根据实验步骤中提出的要求，使用Eclipse编译、运行自己所编写的Java程序；

2.3 根据编译与运行过程中所获得的错误信息修改程序直至获得正确的结果；

记录实验中遇到的各类问题并以及解决办法。

1. 实验环境、设备

开发平台：Eclipse Neon3

操作系统：Window 7及以上

数据库：MySQL5.1

1. 实验代码

1.HelloSpringController.java

@Controller

public class HelloSpringController {

String message = "Welcome to Spring MVC!";

@RequestMapping("/hello")

public ModelAndView showMessage(@RequestParam(value = "name", required = false, defaultValue = "Spring") String name) {

ModelAndView mv = new ModelAndView("hellospring");//指定视图

//向视图中添加所要展示或使用的内容，将在页面中使用

mv.addObject("message", message);

mv.addObject("name", name);

return mv;

}

}

2.SpringContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<!-- 搜索spring控件 -->

<context:component-scan base-package="com.wgq"></context:component-scan>

<!-- 视图页面配置 -->

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix">

<value>/WEB-INF/jsp/</value>

</property>

<property name="suffix">

<value>.jsp</value>

</property>

</bean>

</beans>

3.HelloSpring.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Spring 4 MVC -HelloWorld</title>

</head>

<body>

<center>

<h2>Hello World</h2>

<h2>

${message} ${name} </h2>

</center>

</body>

</html>

4.index.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme() + "://" + request.getServerName() + ":" + request.getServerPort()

+ path + "/";

%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Spring 4 MVC - HelloWorld Index Page</title>

</head>

<body>

<center>

<h2>Hello World</h2>

<h3>

<a href="hello?name=吴跟强">点击跳转</a> <br />

</h3>

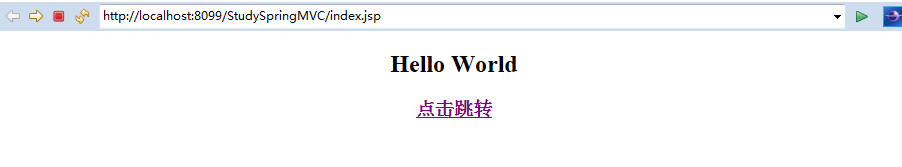
</center>

</body>

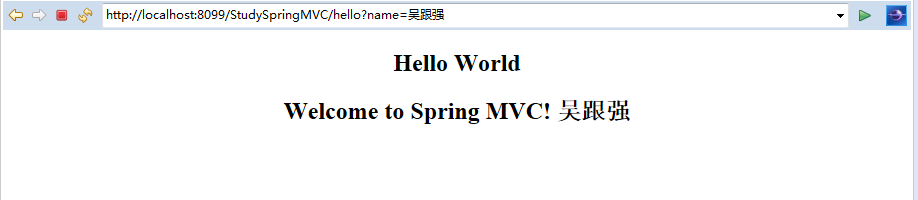
</html>

1. 实验结果与分析

Index进入页面：



跳转页面：



1. 实验心得

通过这次的实验，理解了Spring MVC模型，掌握了Spring MVC的表单数据传递与接收方法；掌握了Spring MVC的表单校验方法；将课堂上的知识和实验结合在一起，对SpringMVC知识更加的理解透彻，同时也学会了使用mysql数据库。

实验项目名称： MyBatis 实验学时： 4

同组学生姓名： 实验地点： 北辅311

实验日期： 2018.04.23 实验成绩：

批改教师： 批改时间：

1. 实验目的
   1. 掌握Mybatis框架的建立步骤；
   2. 掌握MyBatis的使用技巧。
2. 实验要求

2.1 根据实验步骤中提出的要求，使用Eclipse编写相应的Java程序；

2.2 根据实验步骤中提出的要求，使用Eclipse编译、运行自己所编写的Java程序；

* 1. 根据编译与运行过程中所获得的错误信息修改程序直至获得正确的结果；

2.4记录实验中遇到的各类问题并以及解决办法。

1. 实验环境、设备

3.1 开发平台：Eclipse Neon3

3.2 操作系统：Window 7及以上

3.3 数据库：MySQL5.1

1. 实验代码

1.UserInfoMapper.java

. public interface UserInfoMapper {

/\*\*

\*

\* @param userId

\* @return

\*/

int deleteByPrimaryKey(Integer userId);

/\*\*

\*

\* @param record

\* @return

\*/

int insert(UserInfo record);

/\*\*

\*

\* @param record

\* @return

\*/

int insertSelective(UserInfo record);

UserInfo selectByPrimaryKey(Integer userId);

int updateByPrimaryKeySelective(final UserInfo record);

int updateByPrimaryKey(UserInfo record);

/\*\*

\*

\* @return

\*/

List<UserInfo> queryAllUsers();

int insert(String string, UserInfo user);

}

2.UserInfoMapper.xml

<?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.mybatis.demo001.mapper.UserInfoMapper">

<resultMap id="BaseResultMap" type="com.mybatis.demo001.pojo.UserInfo">

<id column="user\_id" jdbcType="INTEGER" property="userId" />

<result column="user\_name" jdbcType="VARCHAR" property="userName" />

<result column="user\_pass" jdbcType="VARCHAR" property="userPass" />

<result column="user\_email" jdbcType="VARCHAR" property="userEmail" />

<result column="user\_position\_id" jdbcType="INTEGER" property="userPositionId" />

<result column="user\_role\_id" jdbcType="INTEGER" property="userRoleId" />

<result column="user\_phone" jdbcType="VARCHAR" property="userPhone" />

<result column="user\_telephone" jdbcType="VARCHAR" property="userTelephone" />

<result column="user\_comment" jdbcType="VARCHAR" property="userComment" />

<result column="user\_department\_id" jdbcType="INTEGER" property="userDepartmentId" />

<result column="user\_created\_datetime" jdbcType="TIMESTAMP" property="userCreatedDatetime" />

<result column="user\_created\_name" jdbcType="INTEGER" property="userCreatedName" />

<result column="user\_updated\_datetime" jdbcType="TIMESTAMP" property="userUpdatedDatetime" />

<result column="user\_updated\_name" jdbcType="INTEGER" property="userUpdatedName" />

<result column="user\_isvalid" jdbcType="BIT" property="userIsValid" />

<result column="user\_default1" jdbcType="VARCHAR" property="userDefault1" />

<result column="user\_default2" jdbcType="VARCHAR" property="userDefault2" />

</resultMap>

<sql id="base\_column\_userinfo">

user\_id, user\_name, user\_pass, user\_email, user\_position\_id, user\_role\_id, user\_phone,

user\_telephone, user\_comment, user\_department\_id, user\_created\_datetime, user\_created\_name,

user\_updated\_datetime, user\_updated\_name, user\_default1, user\_default2, user\_isvalid

</sql>

<select id="selectByPrimaryKey" parameterType="java.lang.Integer" resultMap="BaseResultMap">

select

<include refid="base\_column\_userinfo" />

from user

where user\_id = #{userId,jdbcType=INTEGER}

</select>

<select id="queryAllUsers" resultMap="BaseResultMap">

select

<include refid="base\_column\_userinfo" />

from user

</select>

<delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">

delete from user

where user\_id = #{userId,jdbcType=INTEGER}

</delete>

<insert id="insert" parameterType="com.mybatis.demo001.pojo.UserInfo">

insert into user (user\_id, user\_name, user\_pass,

user\_email, user\_position\_id, user\_role\_id,

user\_phone, user\_telephone, user\_comment,

user\_department\_id, user\_created\_datetime,

user\_created\_name, user\_updated\_datetime,

user\_updated\_name, user\_default1,

user\_default2, user\_isvalid)

values (#{userId,jdbcType=INTEGER}, #{userName,jdbcType=VARCHAR}, #{userPass,jdbcType=VARCHAR},

#{userEmail,jdbcType=VARCHAR}, #{userPositionId,jdbcType=INTEGER}, #{userRoleId,jdbcType=INTEGER},

#{userPhone,jdbcType=VARCHAR}, #{userTelephone,jdbcType=VARCHAR}, #{userComment,jdbcType=VARCHAR},

#{userDepartmentId,jdbcType=INTEGER}, #{userCreatedDatetime,jdbcType=TIMESTAMP},

#{userCreatedName,jdbcType=INTEGER}, #{userUpdatedDatetime,jdbcType=TIMESTAMP},

#{userUpdatedName,jdbcType=INTEGER}, #{userDefault1,jdbcType=VARCHAR},

#{userDefault2,jdbcType=VARCHAR}, #{userIsValid,jdbcType=BIT})

</insert>

<insert id="insertSelective" parameterType="com.mybatis.demo001.pojo.UserInfo">

insert into user

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="userId != null">

user\_id,

</if>

<if test="userName != null">

user\_name,

</if>

<if test="userPass != null">

user\_pass,

</if>

<if test="userEmail != null">

user\_email,

</if>

<if test="userPositionId != null">

user\_position\_id,

</if>

<if test="userRoleId != null">

user\_role\_id,

</if>

<if test="userPhone != null">

user\_phone,

</if>

<if test="userTelephone != null">

user\_telephone,

</if>

<if test="userComment != null">

user\_comment,

</if>

<if test="userDepartmentId != null">

user\_department\_id,

</if>

<if test="userCreatedDatetime != null">

user\_created\_datetime,

</if>

<if test="userCreatedName != null">

user\_created\_name,

</if>

<if test="userUpdatedDatetime != null">

user\_updated\_datetime,

</if>

<if test="userUpdatedName != null">

user\_updated\_name,

</if>

<if test="userDefault1 != null">

user\_default1,

</if>

<if test="userDefault2 != null">

user\_default2,

</if>

<if test="userIsValid != null">

user\_isvalid,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="userId != null">

#{userId,jdbcType=INTEGER},

</if>

<if test="userName != null">

#{userName,jdbcType=VARCHAR},

</if>

<if test="userPass != null">

#{userPass,jdbcType=VARCHAR},

</if>

<if test="userEmail != null">

#{userEmail,jdbcType=VARCHAR},

</if>

<if test="userPositionId != null">

#{userPositionId,jdbcType=INTEGER},

</if>

<if test="userRoleId != null">

#{userRoleId,jdbcType=INTEGER},

</if>

<if test="userPhone != null">

#{userPhone,jdbcType=VARCHAR},

</if>

<if test="userTelephone != null">

#{userTelephone,jdbcType=VARCHAR},

</if>

<if test="userComment != null">

#{userComment,jdbcType=VARCHAR},

</if>

<if test="userDepartmentId != null">

#{userDepartmentId,jdbcType=INTEGER},

</if>

<if test="userCreatedDatetime != null">

#{userCreatedDatetime,jdbcType=TIMESTAMP},

</if>

<if test="userCreatedName != null">

#{userCreatedName,jdbcType=INTEGER},

</if>

<if test="userUpdatedDatetime != null">

#{userUpdatedDatetime,jdbcType=TIMESTAMP},

</if>

<if test="userUpdatedName != null">

#{userUpdatedName,jdbcType=INTEGER},

</if>

<if test="userDefault1 != null">

#{userDefault1,jdbcType=VARCHAR},

</if>

<if test="userDefault2 != null">

#{userDefault2,jdbcType=VARCHAR},

</if>

<if test="userIsValid != null">

#{userIsValid,jdbcType=BIT},

</if>

</trim>

</insert>

<update id="updateByPrimaryKeySelective" parameterType="com.mybatis.demo001.pojo.UserInfo">

update user

<set>

<if test="userName != null">

user\_name = #{userName,jdbcType=VARCHAR},

</if>

<if test="userPass != null">

user\_pass = #{userPass,jdbcType=VARCHAR},

</if>

<if test="userEmail != null">

user\_email = #{userEmail,jdbcType=VARCHAR},

</if>

<if test="userPositionId != null">

user\_position\_id = #{userPositionId,jdbcType=INTEGER},

</if>

<if test="userRoleId != null">

user\_role\_id = #{userRoleId,jdbcType=INTEGER},

</if>

<if test="userPhone != null">

user\_phone = #{userPhone,jdbcType=VARCHAR},

</if>

<if test="userTelephone != null">

user\_telephone = #{userTelephone,jdbcType=VARCHAR},

</if>

<if test="userComment != null">

user\_comment = #{userComment,jdbcType=VARCHAR},

</if>

<if test="userDepartmentId != null">

user\_department\_id = #{userDepartmentId,jdbcType=INTEGER},

</if>

<if test="userCreatedDatetime != null">

user\_created\_datetime = #{userCreatedDatetime,jdbcType=TIMESTAMP},

</if>

<if test="userCreatedName != null">

user\_created\_name = #{userCreatedName,jdbcType=INTEGER},

</if>

<if test="userUpdatedDatetime != null">

user\_updated\_datetime = #{userUpdatedDatetime,jdbcType=TIMESTAMP},

</if>

<if test="userUpdatedName != null">

user\_updated\_name = #{userUpdatedName,jdbcType=INTEGER},

</if>

<if test="userDefault1 != null">

user\_default1 = #{userDefault1,jdbcType=VARCHAR},

</if>

<if test="userDefault2 != null">

user\_default2 = #{userDefault2,jdbcType=VARCHAR},

</if>

<if test="userIsValid != null">

user\_isvalid = #{userIsValid,jdbcType=BIT},

</if>

</set>

where user\_id = #{userId,jdbcType=INTEGER}

</update>

<update id="updateByPrimaryKey" parameterType="com.mybatis.demo001.pojo.UserInfo">

update user

set user\_name = #{userName,jdbcType=VARCHAR},

user\_pass = #{userPass,jdbcType=VARCHAR},

user\_email = #{userEmail,jdbcType=VARCHAR},

user\_position\_id = #{userPositionId,jdbcType=INTEGER},

user\_role\_id = #{userRoleId,jdbcType=INTEGER},

user\_phone = #{userPhone,jdbcType=VARCHAR},

user\_telephone = #{userTelephone,jdbcType=VARCHAR},

user\_comment = #{userComment,jdbcType=VARCHAR},

user\_department\_id = #{userDepartmentId,jdbcType=INTEGER},

user\_created\_datetime = #{userCreatedDatetime,jdbcType=TIMESTAMP},

user\_created\_name = #{userCreatedName,jdbcType=INTEGER},

user\_updated\_datetime = #{userUpdatedDatetime,jdbcType=TIMESTAMP},

user\_updated\_name = #{userUpdatedName,jdbcType=INTEGER},

user\_default1 = #{userDefault1,jdbcType=VARCHAR},

user\_default2 = #{userDefault2,jdbcType=VARCHAR},

user\_isvalid = #{userIsValid,jdbcType=BIT},

where user\_id = #{userId,jdbcType=INTEGER}

</update>

</mapper>

3.Mybatis.xml

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

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

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

<configuration>

<properties resource="jdbc.properties"></properties>

<environments default="development">

<environment id="development">

<transactionManager type="JDBC" />

<dataSource type="POOLED">

<!-- MySQL Pool -->

<property name="driver" value="${jdbc.driverClassName}" />

<property name="url" value="${jdbc.url}" />

<property name="username" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource="com/mybatis/demo001/mapper/UserInfoMapper.xml"></mapper>

</mappers>

</configuration>

4.UserInfoMapperTest

public class UserInfoMapperTest {

UserInfo user = new UserInfo();

/\*\*

\* @throws java.lang.Exception

\*/

@Before

public void setUp() throws Exception {

}

/\*\*

\* @throws java.lang.Exception

\*/

@After

public void tearDown() throws Exception {

}

SqlSession session = null;

/\*\*

\* Test method for {@link com.mybatis.demo001.mapper.UserInfoMapper#queryAllUsers()}.

\*/

@Test

public void testQueryAllUsers() {

SqlSessionFactory sf = null;

try {

sf = MyBatisUtil.getSqlSessionFactory();

session =sf.openSession();

UserInfoMapper mapper =session.getMapper(UserInfoMapper.class);

List<UserInfo> users = mapper.queryAllUsers();

Iterator<UserInfo> its = users.iterator();

while(its.hasNext()){

System.out.println(its.next());

}

session.commit();

} catch (Exception e) {

session.rollback();

}finally{

if(session !=null){

session.close();

}

}

}

@Test

public void testInsert() {

SqlSessionFactory sf = null;

try {

System.out.println("start---");

sf = MyBatisUtil.getSqlSessionFactory();

session =sf.openSession();

UserInfoMapper mapper =session.getMapper(UserInfoMapper.class);

//user.setUserId(4);

user.setUserName("konig1");

user.setUserPass("123");

user.setUserEmail("erwin.wang1@clpsglobal.com");

user.setUserPositionId(1);

user.setUserRoleId(1);

user.setUserPhone("123-12332434");

user.setUserTelephone("1234234535");

user.setUserComment("12342343");

user.setUserDepartmentId(1);

user.setUserCreatedDatetime(new Timestamp(new Date().getTime()));

user.setUserCreatedName(1);

user.setUserUpdatedDatetime(null);

user.setUserUpdatedName(1);

user.setUserDefault1("");

user.setUserDefault2("");

user.setUserIsValid(true);

int flag = mapper.insert(user);

if(flag>0)

System.out.println("ok");

session.commit();

} catch (Exception e) {

session.rollback();

}finally{

if(session !=null){

session.close();

}

}

}

@Test

public void testUpdate() {

SqlSessionFactory sf = null;

try {

sf = MyBatisUtil.getSqlSessionFactory();

session =sf.openSession();

UserInfoMapper mapper =session.getMapper(UserInfoMapper.class);

mapper.updateByPrimaryKey(null);

session.commit();

} catch (Exception e) {

session.rollback();

}finally{

if(session !=null){

session.close();

}

}

}

@Test

public void testDelete() {

SqlSessionFactory sf = null;

try {

sf = MyBatisUtil.getSqlSessionFactory();

session =sf.openSession();

UserInfoMapper mapper =session.getMapper(UserInfoMapper.class);

mapper.deleteByPrimaryKey(3);

session.commit();

} catch (Exception e) {

session.rollback();

}finally{

if(session !=null){

session.close();

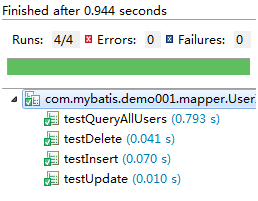
}

}

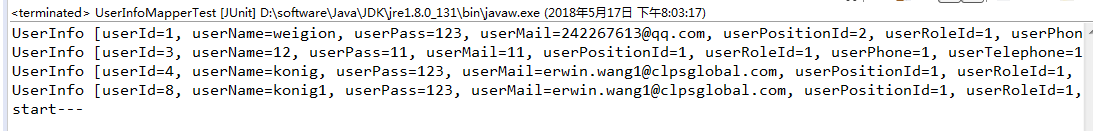
}

}

1. 实验结果与分析



测试结果：



1. 实验心得

经过这次学习，使我对于Mybatis有了更加深入的了解，在过程中也逐渐掌握了Mybatis的使用，MyBatis挺考验程序员对于sql的掌握程度的，但是相对的，编写起来可以更加随心所欲。而且MyBatis的移植性并不算好，它十分依赖数据库。网上其实也有，就是generator，然后你只要按照教程搞一下就可以自动帮你生成sql语句。事务回滚比较人性化一点。相较于Spring的事务回滚(好像默认只有RuntimeException及其子类才会触发)，MyBatis的事务回滚更好一点。这一次的深入学习，我受益匪浅，希望在接下来的实战项目中，可以有更大的进步。