



JavaEE 实验六补充实验

实验报告

姓　　名　　 Mcrivers

班　　级　　 软件工程2102班

学　　号

提交日期　　 2023.12.24

目录

[一、基础实验——Spring 与 Hibernate 的整合 1](#_Toc154354165)

[1. 运行结果截图 1](#_Toc154354166)

[2.总结 DataSource、SessionFactory、CustomerDAO、UserService 对象之间的依赖关系 1](#_Toc154354167)

[3. 总结 Spring 配置文件中对 DataSource、 SessionFactory、CustomerDAO、UserService 的配置方法，以及属性注入的方式 2](#_Toc154354168)

[4.碰到的问题及解决方案或思考 2](#_Toc154354169)

[5.实验收获及总结 4](#_Toc154354170)

[二、提高实验——Spring、Struts 与 Hibernate 的整合 5](#_Toc154354171)

[1. 运行结果截图 5](#_Toc154354172)

[2. 总结 Spring 整合 Struts2 框架的关键步骤 5](#_Toc154354173)

[3. 总结本实验中的 UserAction 与以往实验中的写法关键区别 6](#_Toc154354174)

[4. 总结配置文件 applicationContext.xml 中 bean 元素的 prototype 属性及其取值的含义 6](#_Toc154354175)

[11. 总结 web.xml 文件中添加监听器的目的 6](#_Toc154354176)

[6. 碰到的问题及解决方案或思考 7](#_Toc154354177)

[7.实验收获及总结 7](#_Toc154354178)

[三、扩展实验——Spring AOP 实现事务管理 8](#_Toc154354179)

[1.运行结果截图 8](#_Toc154354180)

[2. 总结 Spring 配置文件中与实现声明式事务 管理相关的主要元素及其属性的作用 8](#_Toc154354181)

[3. 总结 Spring 事务管理中的七种事务传播行为（propagation） 所起到的不同作用 9](#_Toc154354182)

[4. 碰到的问题及解决方案或思考 10](#_Toc154354183)

[5. 实验收获及总结 10](#_Toc154354184)

# 一、基础实验——Spring 与 Hibernate 的整合

## 1. 运行结果截图

观察到，程序成功运行，控制台打印输出插入数据成功的信息（该信息是在saveUser()函数中额外增加的）；查看数据库，发现数据条目确实已经被添加到数据库中。

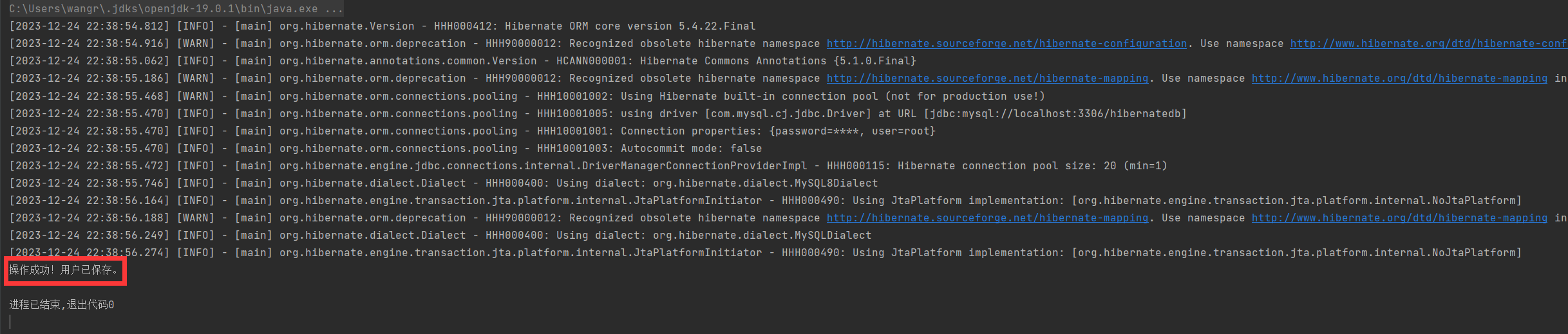


图1.1 程序运行成功时控制台输出的信息



图1.2 程序运行成功后数据库的状态

## 2.总结 DataSource、SessionFactory、CustomerDAO、UserService 对象之间的依赖关系

**Ⅰ.DataSource:**

用途：DataSource是用于获取数据库连接的对象，它提供了与数据库的连接池管理和连接获取相关的功能。

依赖关系：SessionFactory对象依赖于DataSource来获取数据库连接。

**Ⅱ.SessionFactory:**

用途：SessionFactory是用于创建Session对象的工厂，Session用于进行Hibernate持久化操作。

依赖关系：SessionFactory对象依赖于DataSource，它使用DataSource来获取数据库连接。

**Ⅲ.CustomerDAO:**

用途：CustomerDAO是一个数据访问对象，用于执行与Customer实体类相关的数据库操作。

依赖关系：CustomerDAO对象依赖于Session，它需要一个Hibernate Session来执行数据库操作。

**Ⅳ.UserService:**

用途：UserService是一个服务类，用于保存用户信息到数据库。

依赖关系：

UserService依赖于CustomerDAO，它通过CustomerDAO来执行数据库操作。

UserService依赖于SessionFactory，它使用SessionFactory来创建Hibernate Session。

## 3. 总结 Spring 配置文件中对 DataSource、 SessionFactory、CustomerDAO、UserService 的配置方法，以及属性注入的方式

**ⅠDataSource的配置方法：**

①配置数据源的bean，通常使用Spring提供的内置数据源，如BasicDataSource或使用第三方库（例如，C3P0、HikariCP）。

②配置数据源的连接属性，如URL、用户名和密码。

**ⅡSessionFactory的配置方法：**

①配置SessionFactory的bean，引用数据源bean，并指定Hibernate的配置文件。

②配置实体类的扫描路径。

**ⅢCustomerDAO的配置方法：**

①配置CustomerDAO的bean，引用SessionFactory bean，并指定要使用的实体类。

**ⅣUserService的配置方法：**

①配置UserService的bean，引用CustomerDAO bean，并注入其他依赖。

②使用属性注入方式将CustomerDAO注入到UserService。

属性注入的方式通常使用<property>元素来实现，将属性名与对应的bean引用关联。

## 4.碰到的问题及解决方案或思考

碰到的问题：

①程序运行时出现如下报错：



图1.3程序运行时出现的报错信息

分析：发现是SpringEnvTest.java类中调用的方法和UserService.java中的不一致

解决方法：统一方法名，将SpringEnvTest.java中的register改为saveUser，如下图

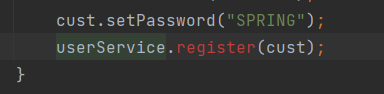


图1.4修改方法名

②程序运行时出现如下报错：

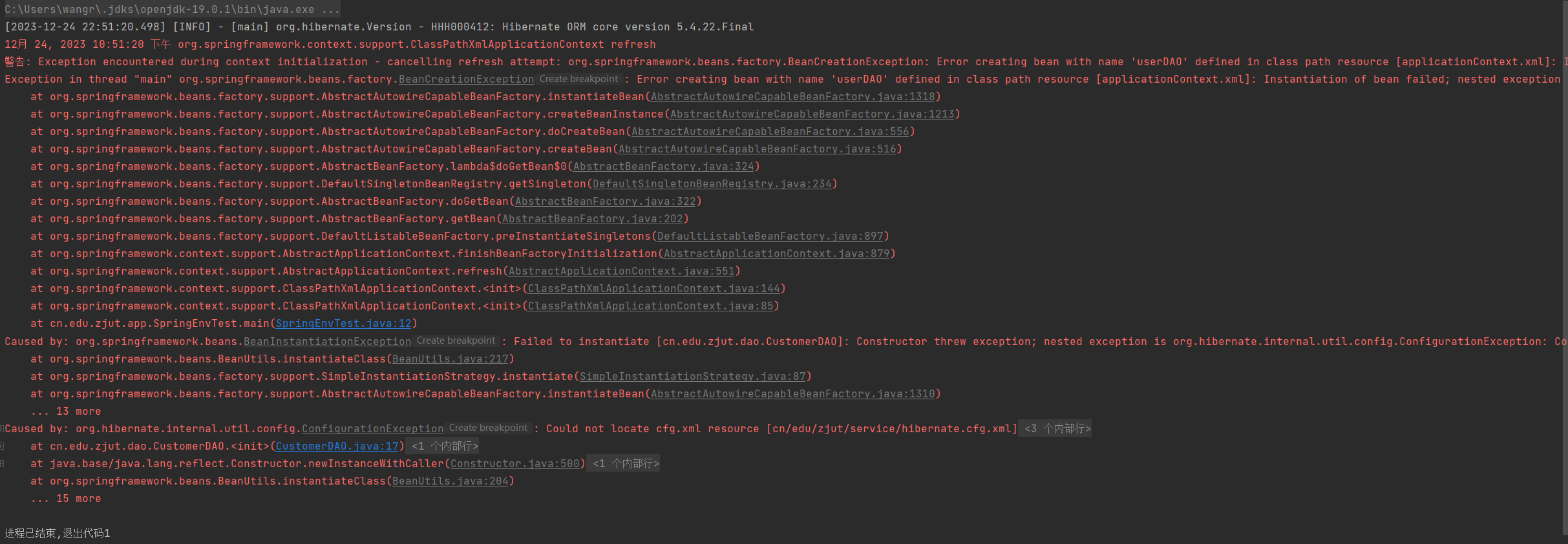


图1.5程序运行时出现的报错信息

分析：发现是缺少hibernate.cfg.xml配置文件

解决办法：增加hibernate.cfg.xml配置文件，如下图

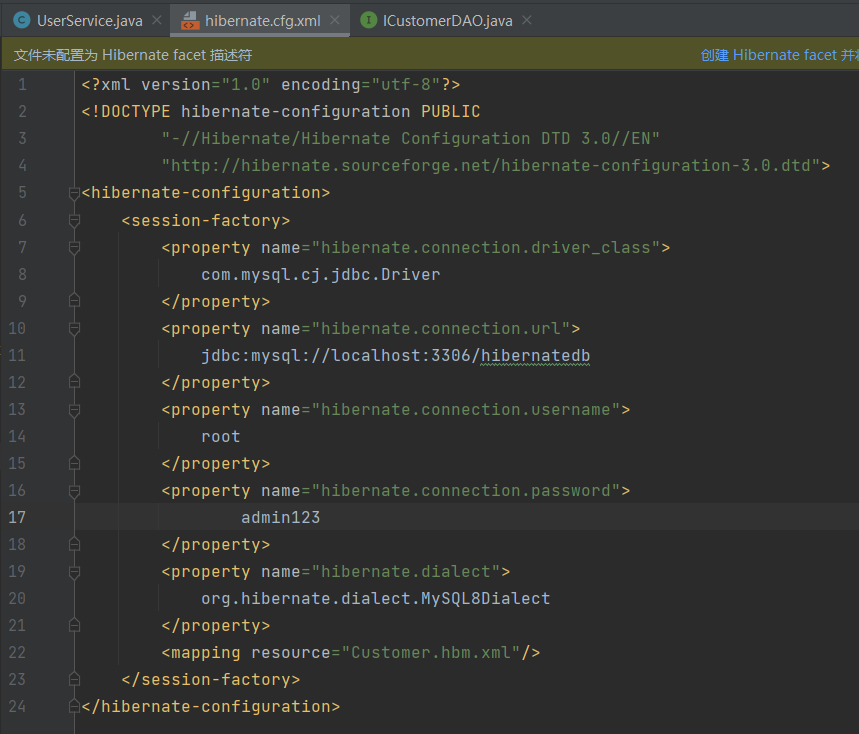


图1.6 hibernate.cfg.xml配置文件

## 5.实验收获及总结

这个实验帮助我更好地理解了Spring和Hibernate的整合，以及如何在Spring应用中配置数据源、SessionFactory和DAO层，以实现数据库操作。此外，我还加深了对Spring IoC容器和属性注入的理解。这些知识和技能对于开发现代化的Java应用程序非常重要。

# 二、提高实验——Spring、Struts 与 Hibernate 的整合

## 1. 运行结果截图



图2.1 register.jsp注册页面



图2.2 regSuccess.jsp注册成功页面



图2.3 regFail.jsp注册失败页面

## 2. 总结 Spring 整合 Struts2 框架的关键步骤

①创建持久化类和Hibernate映射文件：在项目中创建持久化类Customer.java和对应的Hibernate映射文件Customer.hbm.xml以定义数据表的结构。

②创建数据访问层：在项目中创建数据访问层接口ICustomerDAO.java和实现类CustomerDAO.java来进行数据库操作。

③创建业务逻辑层：创建业务逻辑层UserService.java来处理业务逻辑，并提取接口IUserService.java以便后续的依赖注入。

④创建Struts2 Action类：在项目中创建Struts2的Action类UserAction.java，用于接收用户请求并调用业务逻辑层的方法进行处理。

⑤创建Spring配置文件：创建Spring配置文件applicationContext.xml来配置数据源、SessionFactory、以及各个类的bean。

⑥配置Struts2：创建Struts2配置文件struts.xml来配置Action的映射和页面导航。

⑦配置web.xml：在web.xml中配置Struts2核心Filter和Spring监听器

## 3. 总结本实验中的 UserAction 与以往实验中的写法关键区别

依赖注入方式：本实验中的UserAction类通过setter方法注入了一个名为userService的接口实例（IUserService）。这表示UserAction类依赖于IUserService接口，通过接口实现的方式来调用用户注册逻辑。而以前的写法中，UserAction类直接创建了UserService类的实例，然后调用其方法。

数据传递方式：在本实验中，UserAction类使用一个名为loginUser的Customer对象来接收用户输入的数据，然后将其传递给userService的register方法进行处理。而以前的写法中，UserAction类使用一个名为loginUser的UserBean对象来传递数据。

返回结果方式：在本实验中，execute方法返回一个字符串（"success"），该字符串用于Struts2框架进行页面导航。而以前的写法中，execute方法返回一个字符串（"success"或"fail"），然后在UserAction类内部进行页面跳转。

## 4. 总结配置文件 applicationContext.xml 中 bean 元素的 prototype 属性及其取值的含义

scope 属性的取值为 prototype，表示这个bean的作用域是原型，具体含义如下：

**Singleton（单例）：** 当**scope** 属性的取值为 "singleton" 时，Spring容器会在启动时创建该bean的一个单一实例，然后在整个应用程序的生命周期内共享这个实例。这意味着无论何时请求该bean，都会得到同一个实例。

**Prototype（原型）：** 当**scope** 属性的取值为 "prototype" 时，Spring容器会在每次请求该bean时创建一个新的实例，而不是共享一个单一实例。每次请求该bean时，都会返回一个独立的新实例。

## 11. 总结 web.xml 文件中添加监听器的目的

**①加载Spring配置文件：** 当Web应用程序启动时，该监听器会加载指定的Spring配置文件，例如**applicationContext.xml**。这个配置文件包含了Spring应用程序的各种bean定义和配置信息。

**②初始化Spring容器：** 监听器会创建并初始化Spring容器，根据配置文件中的定义，将各个bean实例化并设置其属性。

**③管理Spring Bean：** 一旦Spring容器被初始化，它将负责管理和维护所有在配置文件中定义的Spring bean。这包括创建bean实例、注入依赖、维护作用域等。

**④整合Spring和Web应用程序：** 通过配置Spring监听器，Web应用程序可以让Spring容器管理应用程序中的组件，例如在Struts2框架中，可以通过Spring来管理Action类和其他组件。

## 6. 碰到的问题及解决方案或思考

问题：在UserService.java中，出现报错类 "UserService" 必须声明为抽象，或为实现 "Service" 中的抽象方法 "handleRequest(HttpServletRequest, HttpServletResponse)"

解决方法：报错出现的原因是和jakarta不匹配，将servlet-api.jar包导入后使用javax，问题成功解决。

## 7.实验收获及总结

通过本次实验，我深入了解了如何整合Spring、Struts2和Hibernate框架，以构建一个完整的Web应用程序。实验中涉及到了配置Spring容器、将Struts2的Action交由Spring容器管理、配置Struts2的核心过滤器、以及配置Hibernate映射等关键步骤。这对我的Web开发技能提供了宝贵的经验和知识。

# 三、扩展实验——Spring AOP 实现事务管理

## 1.运行结果截图

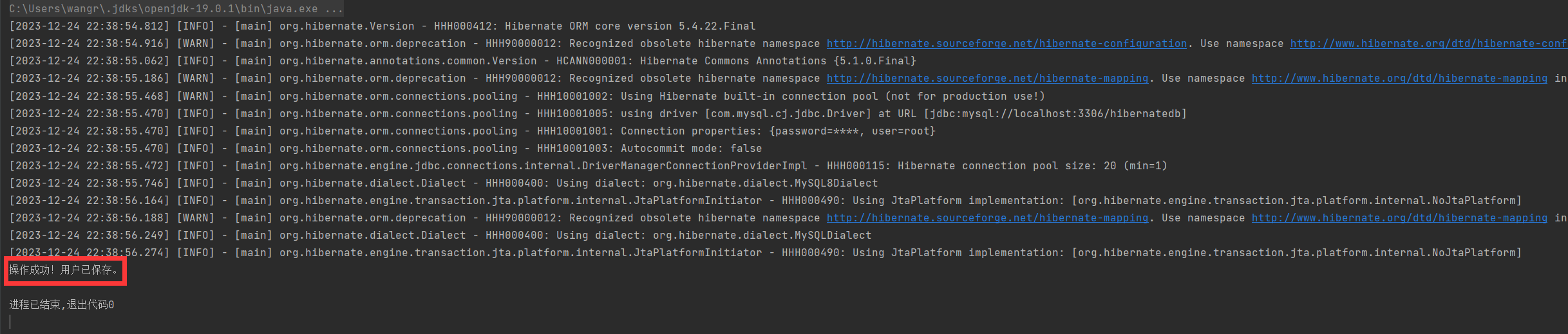
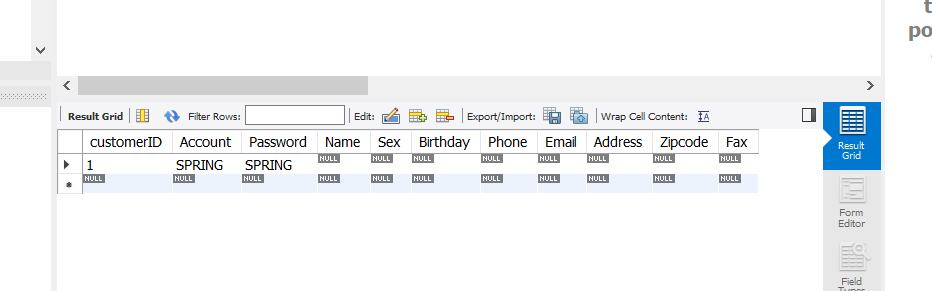


图3.1程序运行成功

图3.2数据被成功加入数据库

## 2. 总结 Spring 配置文件中与实现声明式事务 管理相关的主要元素及其属性的作用

**Ⅰ.<tx:advice>**元素：用于配置事务增强处理，指定了事务的属性和传播行为。

**①id**属性：指定增强处理的唯一标识符。

**②transaction-manager**属性：指定事务管理器的引用，用于管理事务。

**③<tx:attributes>**子元素：定义了一组事务属性配置。

**<tx:method>**子元素：用于定义方法级别的事务配置。

**name**属性：指定方法名的模式，可以使用通配符，如**add\***表示以"add"开头的所有方法。

**propagation**属性：指定事务传播行为，如**REQUIRED**表示如果没有事务，就创建一个新的事务；**REQUIRES\_NEW**表示每次都创建一个新的事务。

**read-only**属性：指定方法是否为只读事务，如果为**true**，则表示只读事务，不会修改数据。

**Ⅱ.<bean>**元素：用于配置事务管理器。

**①id**属性：指定事务管理器的唯一标识符。

**②class**属性：指定事务管理器的实现类，通常为**org.springframework.orm.hibernate5.HibernateTransactionManager**。

**③<property>**子元素：用于设置事务管理器的属性，其中**sessionFactory**属性用于指定关联的**SessionFactory**。

**Ⅲ.<aop:config>**元素：用于配置AOP切面。

**①<aop:pointcut>**子元素：定义一个切入点，指定需要增强的方法。

**id**属性：指定切入点的唯一标识符。

**expression**属性：指定方法的表达式，如**execution(\* cn.edu.zjut.service.\*.\*(..))**表示匹配**cn.edu.zjut.service**包中所有类的所有方法。

**②<aop:advisor>**子元素：定义通知方法和切入点的关联。

**pointcut-ref**属性：指定关联的切入点。

**advice-ref**属性：指定关联的通知方法，即**txAdvice**

## 3. 总结 Spring 事务管理中的七种事务传播行为（propagation） 所起到的不同作用

**REQUIRED**：如果当前存在事务，则加入该事务；如果当前没有事务，则创建一个新的事务。这是默认的传播行为。

**SUPPORTS**：如果当前存在事务，则加入该事务；如果当前没有事务，则以非事务方式执行。

**MANDATORY**：如果当前存在事务，则加入该事务；如果当前没有事务，则抛出异常。

**REQUIRES\_NEW**：无论当前是否存在事务，都创建一个新的事务。如果当前存在事务，会将当前事务挂起。

**NOT\_SUPPORTED**：以非事务方式执行操作，如果当前存在事务，会将当前事务挂起。

**NEVER**：以非事务方式执行操作，如果当前存在事务，则抛出异常。

**NESTED**：如果当前存在事务，则创建一个嵌套事务，嵌套事务可以独立提交或回滚，而不会影响外部事务。如果外部事务回滚，嵌套事务也会回滚，但嵌套事务回滚不会影响外部事务。

## 4. 碰到的问题及解决方案或思考

这个实验过程很顺利，几乎没有遇到问题。

## 5. 实验收获及总结

通过本次实验，我深入了解了Spring框架中的声明式事务管理和AOP的基本概念和原理。通过配置Spring的事务管理器、事务增强处理以及AOP切面，我成功实现了在业务方法中的事务控制，使得数据库操作具有了事务性。

同时，我了解了Spring中的七种事务传播行为，如REQUIRED、SUPPORTS、MANDATORY等，以及它们在不同场景下的应用。这使我能够更灵活地选择适当的事务传播行为来管理事务。