浙江工业大学

JavaEE技术实验报告

实验名称：SSH 整合（Spring4+Struts2+Hibernate4）——基于 SSH 的用户注册模块

学 院： 计算机学院

班 级： 软件工程1801

姓 名： 陈南

学 号： 201806062101

组 号： 无

时 间： 2020年11月23日

**一、基础实验——Spring 框架搭建**

（一）实验目的

1、掌握 Spring 环境搭建的基本方法，能在 JAVASE 应用中使用 Spring，并能在

Eclipse 中开发 Spring 应用；

2、初步理解 Spring 的核心机制：控制反转 IoC（Inversion of Control）与依赖注

入 DI（Dependency Injection）；

3、理解Spring配置文件的作用，掌握bean元素及其属性的作用和基本配置方法。

（二）基本知识与原理

1、Spring 为企业应用的开发提供了一个轻量级的解决方案；

2、Spring 框架共包括 7 个模块，每个模块用于提供不同的解决方案：

（1）Spring Core 模块：提供控制反转（IoC）容器，是 Spring 框架的核心机制，

其它特性都基于 IoC 之上；

（2）Spring Context 模块：提供对 Spring 中对象的框架式访问方式；

（3）Spring DAO 模块：提供了集成 JDBC 的封装包；

（4）Spring ORM 模块：提供了集成常用 ORM 框架的封装包；

（5）Spring Web 模块：提供了 Web 开发以及集成 Web 框架的封装包；

（6）Spring AOP 模块：提供了面向切面编程（AOP）的实现；

（7）Spring MVC 模块：提供了一个 MVC 框架；

3、在传统的程序设计过程中，当某个 Java 实例（调用者）需要调用另一个 Java

实例（被调用者）时，通常由调用者来创建被调用者的实例，而在控制反转

模式下，创建被调用者的工作不再由调用者来完成，两者之间的依赖关系由

Spring 管理，使得两者解耦；

4、在 Spring 中创建被调用者的工作由 Spring 容器来完成，然后将被调用者实例

注入调用者，因此也被称为依赖注入；

5、Spring 推荐面向接口编程，这样可以更好地让规范和实现分离，从而提供更

109好的解耦。

（三）实验内容及步骤

1、登 录 http://maven.springframework.org/release/org/springframework/spring/ 站

点，下载 Spring 框架的依赖 JAR 包（如：spring-framework-4.0.0.RELEASE

-dist）；

2、在Eclipse中新建Java工程spring-prj1，并添加common-logging-1.2.jar和Spring

的 4 个基础 JAR 包到工程中，如下图所示；

图 8-1 Spring 的 4 个基础包

3、在 spring-prj1 中新建 cn.edu.zjut.dao 包，并在其中创建 ICustomerDAO 接口定

义数据持久层的操作，以及实现类 CustomerDAO 实现数据持久层的操作，具

体代码如下：

package cn.edu.zjut.dao;

public interface ICustomerDAO {

public void save();

}

package cn.edu.zjut.dao;

public class CustomerDAO implements ICustomerDAO{

publ**ic CustomerDAO(){**

**System.out.println("create CustomerDao.");**

**}**

public void save() {

System.out.println("execute --save()-- method.");

}

}

4、在 spring-prj1 中创建 Spring 配置文件 applicationContext.xml，并在其中配置

CustomerDAO 实例，具体代码如下：

<?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:p="http://www.springframework.org/schema/p"

110xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd">

**<bean id="userDAO" class="cn.edu.zjut.dao.CustomerDAO" />**

</beans>

5、在 spring-prj1 中新建 cn.edu.zjut.app 包，并在其中创建测试类 SpringEnvTest，

调用 CustomerDAO 实例的 save()方法，具体代码如下：

package cn.edu.zjut.app;

import org.springframework.context.ApplicationContext;

import

org.springframework.context.support.ClassPathXmlApplicationContext;

import cn.edu.zjut.dao.ICustomerDAO;

public class SpringEnvTest {

public static void main(String[] args) {

//创建 Spring 容器

**ApplicationContext ctx = new ClassPathXmlApplicationContext(**

**"applicationContext.xml");**

//获取 CustomerDAO 实例

**ICustomerDAO userDao = (ICustomerDAO) ctx.getBean("userDAO");**

userDao.save();

}

}

6、运行测试类 SpringEnvTest，观察控制台的输出，并记录运行结果；

7、在 spring-prj1 中新建 cn.edu.zjut.service 包，并在其中创建 IUserService 接口

定义注册逻辑，以及实现类 UserService 实现注册逻辑，具体代码如下：

package cn.edu.zjut.service;

public interface IUserService {

public void register();

}

package cn.edu.zjut.service;

import cn.edu.zjut.dao.ICustomerDAO;

public class UserService implements IUserService {

**private ICustomerDAO customerDAO = null;**

public UserService(){

System.out.println("create UserService.");

}

111**public void setCustomerDAO(ICustomerDAO customerDAO) {**

**System.out.println("--setCustomerDAO--");**

**this.customerDAO = customerDAO;**

**}**

public void register() {

System.out.println("execute --register()-- method.");

**customerDAO.save();**

}

}

8、修改 Spring 配置文件 applicationContext.xml，在其中增加 UserService 实例的

配置，具体代码如下：

<?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:p="http://www.springframework.org/schema/p"

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

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd">

<bean id="userDAO" class="cn.edu.zjut.dao.CustomerDAO" />

**<bean id="userService" class="cn.edu.zjut.service.UserService">**

**<property name="customerDAO" ref="userDAO" />**

**</bean>**

</beans>

9、修改测试类 SpringEnvTest，调用 UserService 实例的 register()方法，具体代码

如下：

package cn.edu.zjut.app;

import org.springframework.context.ApplicationContext;

import

org.springframework.context.support.ClassPathXmlApplicationContext;

import cn.edu.zjut.dao.ICustomerDAO;

public class SpringEnvTest {

public static void main(String[] args) {

//创建 Spring 容器

**ApplicationContext ctx = new ClassPathXmlApplicationContext(**

**"applicationContext.xml");**

//获取 UserService 实例

**IUserService userService =**

**(IUserService) ctx.getBean("userService");**

userService.register();

}

112}

10、运行测试类 SpringEnvTest，观察控制台的输出，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验过程，观察运行后的控制台输出，查找相关资料，总结 Spring 容

器管理 Bean 组件的过程（如：何时加载、何时调用 Bean 实例中的方法等）

以及进行依赖注入的过程，并记录下来；

（3）根据实验步骤 4、8，查找相关资料，总结 Spring 配置文件中 bean 元素及

其属性、子元素的作用，并记录下来；

（4）根据实验步骤 5、9，查找相关资料，总结控制反转模式下两个 Java 实例的

依赖关系与传统的程序设计过程体现出的依赖关系由什么区别，控制反转

的优点是什么，并记录下来；

（5）碰到的问题及解决方案或思考；

（6）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**二、提高实验——Spring 与 Hibernate 的整合**

（一）实验目的

1、进一步熟悉 Spring 基础环境搭建的方法，以及在 Eclipse 中开发 Spring 应用

的主要步骤；

2、掌握 Spring 框架与 Hibernate 框架整合的基本步骤，理解 Spring 容器对

DataSource 实例和 SessionFactory 实例的管理；

3、进一步理解 Spring 中控制反转 IoC 的核心机制；

4、进一步熟悉 Spring 配置文件，掌握 Spring 配置文件中对 DataSource 和

SessionFactory 的配置方法和属性注入的方式。

（二）基本知识与原理

1、Spring 框架与 Hibernate 框架整合，将生成 DataSource 对象和 SessionFactory

对象的过程交给 Spring 容器实现，而不在代码中实现，即由 IoC 容器控制对

113114

象的生成和属性的注入；

2、使用 IoC 装配对象（如配置 SessionFactory 对象），就必须在 Spring 配置文

件（默认为 applicationContext.xml）中进行配置；

3、Spring 框架中的 IoC 容器管理的对象都被称为 bean，bean 都需要再配置文件

的<beans>元素下使用<bean>元素配置；

4、Spring IoC 容器的代表者是 API 中的 BeanFactory 接口，IoC 容器装配成功的

对象，都将通过 BeanFactory 获得，进而在应用中使用。

（三）实验内容及步骤

1、在 MySQL 中创建一个名称为 hibernatedb 的数据库，并在该数据库中创建一

个名称为 customer 的数据表，表结构如表 1-1 所示：

2、在 Java 工程 spring-prj1 中添加 MySQL 驱动程序库文件和 Hibernate 核心包，

并添加 Spring 框架中与数据库操作相关的三个 JAR 包：spring-jdbc-4.0.0.

RELEASE.jar、spring-orm-4.0.0.RELEASE.jar、spring-tx-4.0.0.RELEASE.jar；

3、在 spring-prj1 中新建 cn.edu.zjut.po 包，并在其中创建持久化类 Customer.java

以及 Hibernate 映射文件 Customer.hbm.xml，具体代码可参照“实验五

Hibernate 基础应用”中基础实验里的内容；

4、在 cn.edu.zjut.dao 包中创建数据库操作基础类 BaseHibernateDAO.java，具体

代码如下：package cn.edu.zjut.dao;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

public class BaseHibernateDAO{

**private SessionFactory sessionFactory;**

public Session getSession(){

return sessionFactory.openSession();

}

**public void setSessionFactory(SessionFactory sessionFactory) {**

**this.sessionFactory = sessionFactory;**

**}**

}

5、修改 cn.edu.zjut.dao 包中的 ICustomerDAO 接口和 CustomerDAO 实现类，为

save()方法增加一个 Customer 类型的输入参数，并使 CustomerDAO 继承数据

库操作基础类 BaseHibernateDAO，具体代码如下：

package cn.edu.zjut.dao;

import cn.edu.zjut.po.Customer;

public interface ICustomerDAO {

void save(Customer transientInstance);

}

package cn.edu.zjut.dao;

import org.hibernate.Session;

import org.hibernate.Transaction;

import cn.edu.zjut.po.Customer;

public class CustomerDAO extends BaseHibernateDAO

implements ICustomerDAO{

public void save(Customer transientInstance) {

Transaction tran = null;

Session session = null;

try {

session = getSession();

tran = session.beginTransaction();

session.save(transientInstance);

tran.commit();

} catch (RuntimeException re) {

if(tran != null) tran.rollback();

115throw re;

} finally {

session.close();

}

}

}

6、相应地修改 cn.edu.zjut.service 包中的 IUserService 接口和 UserService 实现类，

为 register()方法增加一个 Customer 类型的输入参数，代码略；

7、修改 Spring 配置文件 applicationContext.xml，修改 beans 元素的属性，并在其

中增加数据源的配置，代码片段如下：

<?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"**

**xmlns:aop="http://www.springframework.org/schema/aop"**

**xmlns:tx="http://www.springframework.org/schema/tx"**

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

**http://www.springframework.org/schema/beans/spring-beans-4.0.xsd**

**http://www.springframework.org/schema/aop**

**http://www.springframework.org/schema/aop/spring-aop-4.0.xsd**

**http://www.springframework.org/schema/tx**

**http://www.springframework.org/schema/tx/spring-tx-4.0.xsd**

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

**http://www.springframework.org/schema/context/spring-context-4.0.xs**

**d">**

**<bean id="dataSource" class="org.springframework.jdbc.**

**datasource.DriverManagerDataSource">**

<property name="driverClassName"

value="com.mysql.jdbc.Driver"/>

<property name="url"

value="jdbc:mysql://localhost:3306/hibernatedb"/>

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

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

</bean>

……

</beans>

8、修改 Spring 配置文件 applicationContext.xml，在其中增加 SessionFactory 实例

的配置，代码片段如下：

<bean **id="sessionFactory"**

class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="hibernateProperties">

116<props>

<prop key="hibernate.dialect">

org.hibernate.dialect.MySQLDialect

</prop>

</props>

</property>

<property name="mappingResources">

<list>

<value>cn/edu/zjut/po/Customer.hbm.xml</value>

</list>

</property>

</bean>

9、修改 Spring 配置文件 applicationContext.xml，在其中增加 BaseHibernateDAO

实例的配置并在其中注入 sessionFactory，同时修改 CustomerDAO 实例的配

置属性，代码片段如下：

<bean id="baseDAO" class="cn.edu.zjut.dao.BaseHibernateDAO">

**<property name="sessionFactory" ref="sessionFactory" />**

</bean>

<bean id="userDAO" class="cn.edu.zjut.dao.CustomerDAO"

**parent="baseDAO"**/>

10、修改测试类 SpringEnvTest，注册一个新用户，具体代码如下：

package cn.edu.zjut.app;

import org.springframework.context.ApplicationContext;

import

org.springframework.context.support.ClassPathXmlApplicationContext;

import cn.edu.zjut.po.Customer;

import cn.edu.zjut.service.IUserService;

public class SpringEnvTest {

public static void main(String[] args) {

ApplicationContext ctx = new ClassPathXmlApplicationContext(

"applicationContext.xml");

IUserService userService =

(IUserService) ctx.getBean("userService");

**Customer cust = new Customer();**

**cust.setAccount("SPRING");**

**cust.setPassword("SPRING");**

**userService.register(cust);**

}

}

11、运行测试类 SpringEnvTest，记录运行结果。

117（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验过程，总结 DataSource、SessionFactory、CustomerDAO、UserService

对象之间的依赖关系，并记录下来；

（3）根据实验步骤 7-9，查找相关资料，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；

（4）碰到的问题及解决方案或思考；

（5）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**三、扩展实验——Spring、Struts 与 Hibernate 的整合**

（一）实验目的

1、掌握 Spring 框架与 Hibernate 框架、Struts 框架整合的基本步骤，理解 Spring

容器对 Bean 实例的管理；

2、理解 Spring 容器对 Struts2 核心控制器 Action 的管理，并理解 Struts 配置文件

和 web.xml 中相应产生的变化；

3、进一步理解 Spring 中控制反转 IoC 的核心机制。

（二）基本知识与原理

1、使用 Spring 整合 Struts2 框架，其核心思想是将 Struts2 的 Action 实例交给

Spring 框架的 IoC 容器装配管理，因此在 Action 类中应提供必要的 setters 方

法以注入所需的属性，同时 struts.xml 文件中的<action>元素的 class 属性将不

再是该 Action 对应的实际类型，而是与 applicationContext.xml 中 Action 的

bean 的 id 对应；

2、使用 Spring 整合 Struts2 框架，还需要在 web.xml 文件中配置一个 listener 来

完成加载 Spring 配置文件的功能。

（三）实验内容及步骤

1181、在 Eclipse 中新建 Web 工程 spring-prj1，并将 common-logging-1.2.jar、MySQL

驱动程序库文、Hibernate 核心包和 Spring 的 7 个 JAR 包（其中包括 Spring

的 4 个基础 JAR 包以及与数据库操作相关的三个 JAR 包）添加到工程中；

2、将 Struts2 中的 8 个核心包（参考实验二 Struts2 基础应用中的基础实验步骤），

以及 Struts2 对 Spring 进行支持的 JAR 包 struts2-spring-plugin-2.3.15.1.jar 添加

到工程中；

3、将 Spring 支持 web 开发的 JAR 包 spring-web-4.0.0.RELEASE.jar 添加到工程

中；

4、在 spring-prj1 中新建 cn.edu.zjut.po 包，并在其中创建持久化类 Customer.java

以及 Hibernate 映射文件 Customer.hbm.xml，代码略；

5、在 spring-prj1 中新建 cn.edu.zjut.dao 包，并在其中创建数据库操作基础类

BaseHibernateDAO.java、ICustomerDAO 接口和 CustomerDAO 实现类，代码

略；

6、在 spring-prj1 中新建 cn.edu.zjut.service 包，并在其中创建 IUserService 接口

和 UserService 实现类，代码略；

7、在 spring-prj1 中新建 cn.edu.zjut.action 包，并在其中创建 UserAction.java，用

于调用用户注册逻辑，代码片段如下：

package cn.edu.zjut.action;

import cn.edu.zjut.po.Customer;

import cn.edu.zjut.service.IUserService;

public class UserAction {

private Customer loginUser;

private IUserService userService = null;

//省略 loginUser 的 getters/setters 方法

**public void setUserService(IUserService userService) {**

**this.userService = userService;**

**}**

public String execute() {

userService.register(loginUser);

return "success";

}

}

8、在项目的 WebRoot/WEB-INF/路径下创建 Spring 配置文件 applicationContext.

xml，并参考提高实验中的内容进行配置；

9、修改 Spring 配置文件 applicationContext.xml，增加对 UserAction 实例的配置，

119代码片段如下：

<bean id="userAction" class="cn.edu.zjut.action.UserAction"

**scope="prototype"**>

**<property name="userService" ref="userService" />**

</bean>

10、在项目的 src/路径下创建 Struts2 配置文件 struts.xml，用于配置 Action 并设

置页面导航，具体代码如下：

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

<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts

Configuration 2.1//EN"

"http://struts.apache.org/dtds/struts-2.1.dtd">

<struts>

<package name="strutsBean" extends="struts-default"

namespace="/">

<action name="register" **class="userAction"**>

<result name="success">/regSuccess.jsp</result>

<result name="fail">/regFail.jsp</result>

</action>

</package>

</struts>

11、编辑 Web 应用的 web.xml 文件，增加 Struts2 核心 Filter 的配置，并添加对

Spring 监听器的配置，代码片段如下：

<web-app>

**<listener>**

**<listener-class>**

**org.springframework.web.context.ContextLoaderListener**

**</listener-class>**

**</listener>**

……

<web-app>

12、在 spring-prj1 中 新 建 register.jsp 页 面 ， 作 为 用 户 注 册 视 图 ， 新 建

regiSuccess.jsp 和 regFail.jsp 页面，分别作为注册成功和失败的视图（代码略）；

13、将 spring-prj1 部署在 Tomcat 服务器上；

14、通过浏览器访问 register.jsp 页面，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）结合实验过程，总结 Spring 整合 Struts2 框架的关键步骤，并记录下来；

120（3）根据实验步骤 7，总结本实验中的 UserAction 与以往实验中的写法关键区

别，并记录下来；

（4）根据实验步骤 9，查找相关资料，总结配置文件 applicationContext.xml 中

bean 元素的 prototype 属性及其取值的含义，并记录下来；

（5）根据实验步骤 11，查找相关资料，总结 web.xml 文件中添加监听器的目的，

并记录下来；

（6）碰到的问题及解决方案或思考；

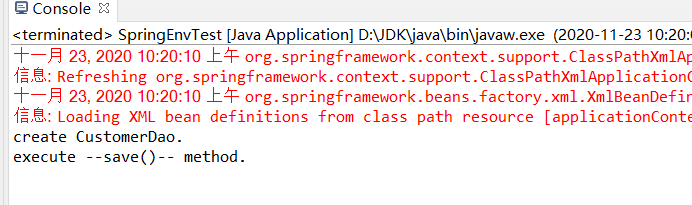
（7）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

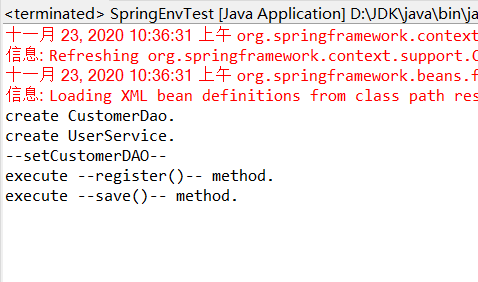
**一、基础实验——Spring 框架搭建**

**（1）运行结果截图；**

1.第6小题



1. 第10小题



**（2）根据实验过程，观察运行后的控制台输出，查找相关资料，总结 Spring 容**

**器管理 Bean 组件的过程（如：何时加载、何时调用 Bean 实例中的方法等）**

**以及进行依赖注入的过程，并记录下来；**

根据控制台的输出可知，它先调用了CustomerDAO的构造方法，再调用了UserService中的构造方法，再调用了setCustomerDAO的方法，后先调用UserService这种的register方法，最后调用CustomerDAO的save方法。

Spring bean的生命周期：

1. 实例化

AbstractAutowireCapableBeanFactory.doCreateBean中会调用createBeanInstance()方法,该阶段主要是从beanDefinitionMap循环读取bean,获取它的属性，然后利用反射(core包下有ReflectionUtil会先强行将构造方法setAccessible(true))读取对象的构造方法(spring会自动判断是否是有参数还是无参数，以及构造方法中的参数是否可用),然后再去创建实例（newInstance）

1. 初始化

初始化主要包括两个步骤,一个是属性填充，另一个就是具体的初始化过程

2.2.1：属性赋值

PopulateBean()会对bean的依赖属性进行填充，@AutoWired注解注入的属性就发生这个阶段，假如我们的bean有很多依赖的对象，那么spring会依次调用这些依赖的对象进行实例化，注意这里可能会有循环依赖的问题。后面我们会讲到spring是如何解决循环依赖的问题

2.2.2：初始化

初始化的过程包括将初始化好的bean放入到spring的缓存中、填充我们预设的属性进一步做后置处理等

1. 使用和销毁

在Spring将所有的bean都初始化好之后，我们的业务系统就可以调用了。而销毁主要的操作是销毁bean，主要是伴随着spring容器的关闭，此时会将spring的bean移除容器之中。此后spring的生命周期到这一步彻底结束，不再接受spring的管理和约束。

实例化和初始化的区别：实例化是在jvm的堆中创建了这个对象实例，此时它只是一个空的对象，所有的属性为null。而初始化的过程就是讲对象依赖的一些属性进行赋值之后，调用某些方法来开启一些默认加载。比如spring中配置的数据库属性Bean，在初始化的时候就会将这些属性填充，比如driver、jdbcurl等,然后初始化连接

3.1:实例化阶段

该阶段会调用对象的空构造方法进行对象的实例化，在进行实例化之后，会调用InstantiationAwareBeanPostProcessor的postProcessBeforeInstantiation方法

BeanPostProcessor(具体实现是InstantiationAwareBeanPostProcessor). postProcessBeforeInstantiation();

这个阶段允许在Bena进行实例化之前，允许开发者自定义逻辑，如返回一个代理对象。不过需要注意的是假如在这个阶段返回了一个不为null的实例，spring就会中断后续的过程。

BeanPostProcessor.postProcessAfterInstantiation()

这个阶段是Bean实例化完毕后执行的后处理操作，所有在初始化逻辑、装配逻辑之前执行

3.2:初始化阶段

3.2.1:BeanPostProcessor.postProcessBeforeInitialization

该方法在bean初始化方法前被调用，Spring AOP的底层处理也是通过实现BeanPostProcessor来执行代理逻辑的

3.2.2:InitializingBean.afterPropertiesSet

自定义属性值 该方法允许我们进行对对象中的属性进行设置，假如在某些业务中，一个对象的某些属性为null,但是不能显示为null，比如显示0或者其他的固定 数值，我们就可以在这个方法实现中将null值转换为特定的值

3.2.3:BeanPostProcessor.postProcessAfterInitialization(Object bean, String beanName)。可以在这个方法中进行bean的实例化之后的处理，比如我们的自定义注解，对依赖对象的版本控制自动路由切换。比如有一个服务依赖了两种版本的实现，我们如何实现自动切换呢？这时候可以自定义一个路由注解,假如叫@RouteAnnotaion，然后实现BeanPostProcessor接口，在其中通过反射拿到自定义的注解@RouteAnnotaion再进行路由规则的设定。

**（3）根据实验步骤 4、8，查找相关资料，总结 Spring 配置文件中 bean 元素及其属性、子元素的作用，并记录下来；**

bean标签的属性

1）scope：用来配置spring bean的作用域

2）singleton：表示bean为单例的

3）abstract：设置为true，将该bean仅仅作为模板使用，应用程序上下文不会试图预先初始化它

4）lazy-init：设为true，延迟加载，该bean不会在ApplicationContext启动时提前被实例化，而是第一次向容器通过getBean索取bean时实例化

注：只对singleton的bean起作用

5）autowire：自动装配

6）dependency-check：依赖检查

7）depends-on：表示一个bean的实例化依靠另一个bean先实例化

8）autowire-candidate：设为false，容器在查找自动装配对象时，将不考虑该bean，即它不会被考虑作为其他bean自动装配的候选者，但是该bean本身可以使用自动装配来注入其他bean

9）primary：该bean优先被注入

10）init-method：初始化bean时调用的方法

11）destory-method：容器销毁之前所调用的方法

12）factory-method：当调用factory-method所指向的方法时，才开始实例化bean

13）factory-bean：调用静态工厂方法的方式创建bean

bean的子元素

1）meta：元数据，当需要使用里面的信息时可以通过key获取

2）lookup-method：获取器注入，是把一个方法声明为返回某种类型的bean但实际要返回的bean是在配置文件里面配置的

3）replaced-method：可以在运行时调用新的方法替换现有的方法，还能动态的更新原有方法的逻辑

4）constructor-arg：对bean自动寻找对应的构造函数，并在初始化的时候将设置的参数传入进去

5）property：基本数据类型赋值

6）qualifier：通过Qualifier指定注入bean的名称

**（4）根据实验步骤 5、9，查找相关资料，总结控制反转模式下两个 Java 实例的依赖关系与传统的程序设计过程体现出的依赖关系由什么区别，控制反转**

**的优点是什么，并记录下来；**

它把传统上由程序代码直接操控的对象的调用权交给外部容器，通过容器来实现对象组件的装配和管理。所谓的“控制反转”概念就是组件对象的控制权转移了，从程序代码本身转移到了外部容器

IoC它将实现组件间关系从程序内部提到外部容器来管理。也就是说由容器在运行期将组件间的某种依赖关系动态的注入到组件中。控制程序间关系的实现交给了外部的容器来完成

依赖注入的基本原则是：应用组件不应该负责查找资源或者其他依赖的协作对象。配置对象的工作应该由IoC容器负责，“查找资源”的逻辑应该从应用组件的代码中抽取出来，交给IoC容器负责。

IoC最大的好处

因为把对象生成放在了XML里定义，所以当我们需要换一个实现子类将会变成很简单(一般这样的对象都是实现于某种接口的)，只要修改XML就可以了，这样我们甚至可以实现对象的热插拔(有点像USB接口和SCSI硬盘了)。

IoC最大的缺点

(1)生成一个对象的步骤变复杂了(事实上操作上还是挺简单的)，对于不习惯这种方式的人，会觉得有些别扭和不直观。

(2)对象生成因为是使用反射编程，在效率上有些损耗。但相对于IoC提高的维护性和灵活性来说，这点损耗是微不足道的，除非某对象的生成对效率要求特别高。

(3)缺少IDE重构操作的支持，如果在Eclipse要对类改名，那么你还需要去XML文件里手工去改了，这似乎是所有XML方式的缺陷所在。

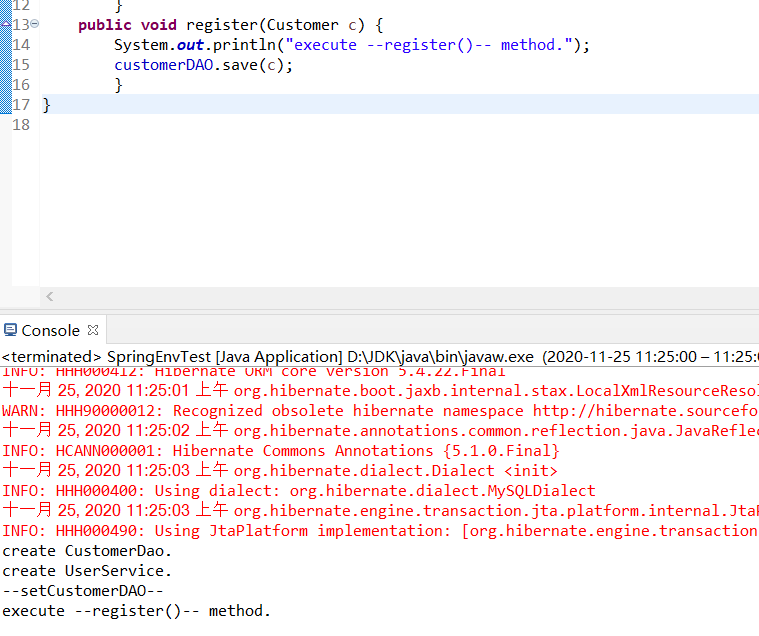
**（5）碰到的问题及解决方案或思考；**

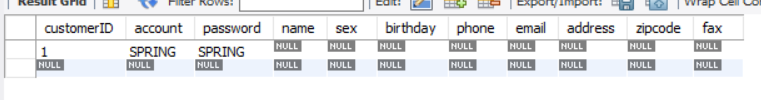
1.思考：spring与传统java的思想有了本质上的区别，就是spring的控制翻转，ioc的思想让人能够设计出松耦合的程序。

1. **实验收获及总结。**
2. 了解spring中bean的生命周期。
3. 了解了对bean标签的属性及其书写。

**二、提高实验——Spring 与 Hibernate 的整合**

1. **运行结果截图；**





**（2）根据实验过程，总结 DataSource、SessionFactory、CustomerDAO、UserService对象之间的依赖关系，并记录下来；**

DataSource：

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

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

<property name="url" value="jdbc:mysql://localhost:3306/hibernatedb?useSSL=true&amp;serverTimezone=UTC"/>

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

<property name="password" value="cjk33014423"/>

</bean>

SessionFactory

<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect"> org.hibernate.dialect.MySQLDialect </prop>

</props>

</property>

<property name="mappingResources">

<list>

<value>cn/edu/zjut/po/Customer.hbm.xml</value>

</list>

</property>

</bean>

CustomerDAO

<bean id="userDAO" class="cn.edu.zjut.dao.CustomerDAO" parent="baseDAO"/>

UserService

<bean id="userService" class="cn.edu.zjut.service.UserService">

<property name="customerDAO" ref="userDAO" />

</bean>

先创建DataSource，在创建SqlSessionFactory在创建事务，最后包装到SqlSessionTemplate中，其中需要制定分库的mapper文件地址，以及分库到层代码，之后扫描dao层，并且给dao层注入指定的SqlSessionTemplate。所有@Bean都需要按照命名指定正确

**（3）根据实验步骤 7-9，查找相关资料，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；**

在Spring的applicationContext.xml中配置bean

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

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

<property name="url" value="jdbc:mysql://localhost:3306/hibernatedb?useSSL=true&amp;serverTimezone=UTC"/>

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

<property name="password" value="cjk33014423"/>

</bean>

<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect"> org.hibernate.dialect.MySQLDialect </prop>

</props>

</property>

<property name="mappingResources">

<list>

<value>cn/edu/zjut/po/Customer.hbm.xml</value>

</list>

</property>

</bean>

<bean id="userDAO" class="cn.edu.zjut.dao.CustomerDAO" parent="baseDAO"/>

<bean id="userService" class="cn.edu.zjut.service.UserService">

<property name="customerDAO" ref="userDAO" />

</bean>

**（4）碰到的问题及解决方案或思考；**

1.思考：对bean中的配置有了更深入的了解与思考，明白了与数据库连接所需要的bean的配置，思考了更换数据库时所需的代码修改。

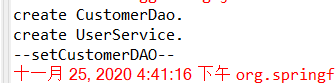
**（5）实验收获及总结。**

1.了解并实践了spring与hibernate的结合，测试了数据库的连接。

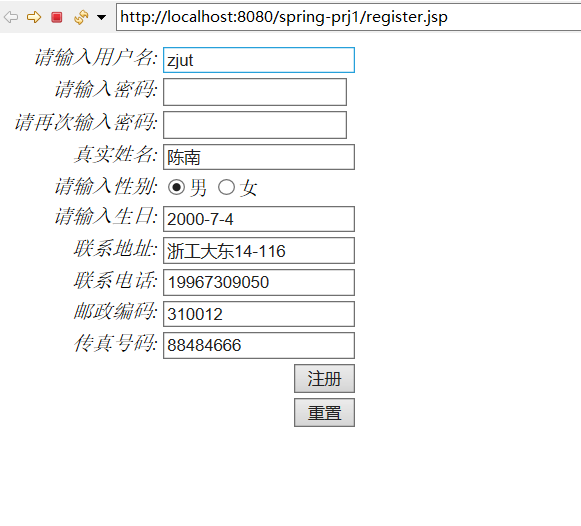
**三、扩展实验——Spring、Struts 与 Hibernate 的整合**

**（1）运行结果截图；**

启动tomcat，打开register.jsp之前



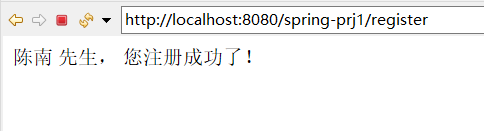
注册页面：



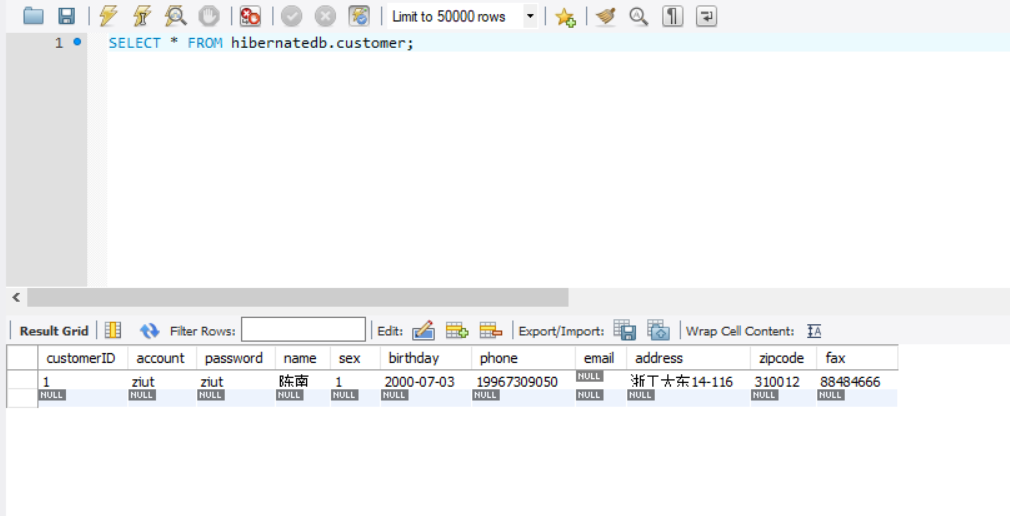
注册后工作台信息：



注册后跳转页面：



数据库页面：



**（2）结合实验过程，总结 Spring 整合 Struts2 框架的关键步骤，并记录下来；**

Sturt2框架的配置，Action，struts.xml，web.xml的配置还需与之前的实验一样，其中UserService的配置与之前有了相应的变动。

与Spring结合后的相关配置：

<bean id="userDAO" class="cn.edu.zjut.dao.CustomerDAO" parent="baseDAO"/>

<bean id="userService" class="cn.edu.zjut.service.UserService">

<property name="customerDAO" ref="userDAO" />

</bean>

<bean id="userAction" class="cn.edu.zjut.action.UserAction" scope="prototype">

<property name="userService" ref="userService" />

</bean>

UserService中的代码不许过大的改动，完成所需要的业务逻辑即可。

**（3）根据实验步骤 7，总结本实验中的 UserAction 与以往实验中的写法关键区别，并记录下来；**

UserAction的配置中多了一个函数：

public void setUserService(IUserService userService) {

this.userService = userService; }

**（4）根据实验步骤 9，查找相关资料，总结配置文件 applicationContext.xml 中**

**bean 元素的 prototype 属性及其取值的含义，并记录下来；**

1、如果是基本数据类型，可以通过setter方法为对象中的属性设置初始值，应用：可以把以前写dbc的东西写进去

2、如果属性的类型不是基本类型或String ，可以使用引用的方式为对象赋值（bean中property中的ref)

扩展-以此方式可以把数据库的连接值给实现类赋值

3、集合属性的赋值，注意要集合要初始化。基本数据类型不用初始化的原因就是它默认初始化（不常用）

4、通过构造方法为属性赋值（不常用，一般都是使用setter方法赋值）

5、属性自动绑定（不常用）

**（5）根据实验步骤 11，查找相关资料，总结 web.xml 文件中添加监听器的目的，并记录下来；**

在启动Web容器时，自动装配Spring applicationContext.xml 的配置信息。

**（6）碰到的问题及解决方案或思考；**

1.主要问题：tomcat启动不起来，与添加的包不对有关系。

解决方案:根据报错信息寻找所未添加的包，添加到lib下就可。（所需要的包还是挺多的，ppt里所要的包也不能落下）

1. **实验收获及总结。**

1.实践了spring、hibernate与strut2框架的整合。

2.实践之后更加清楚的认识到了bean中参数所对应的关系，struts.xml中的class不能为带包的类，而是spring框架中bean的名字。