**实验八 Spring项目搭建**

学生姓名： 张小锋 学 号： 8002119102 专业班级： 计软192班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期： 2021.12.1 实验成绩：

1. 实验项目名称

Spring项目搭建

1. 实验目的

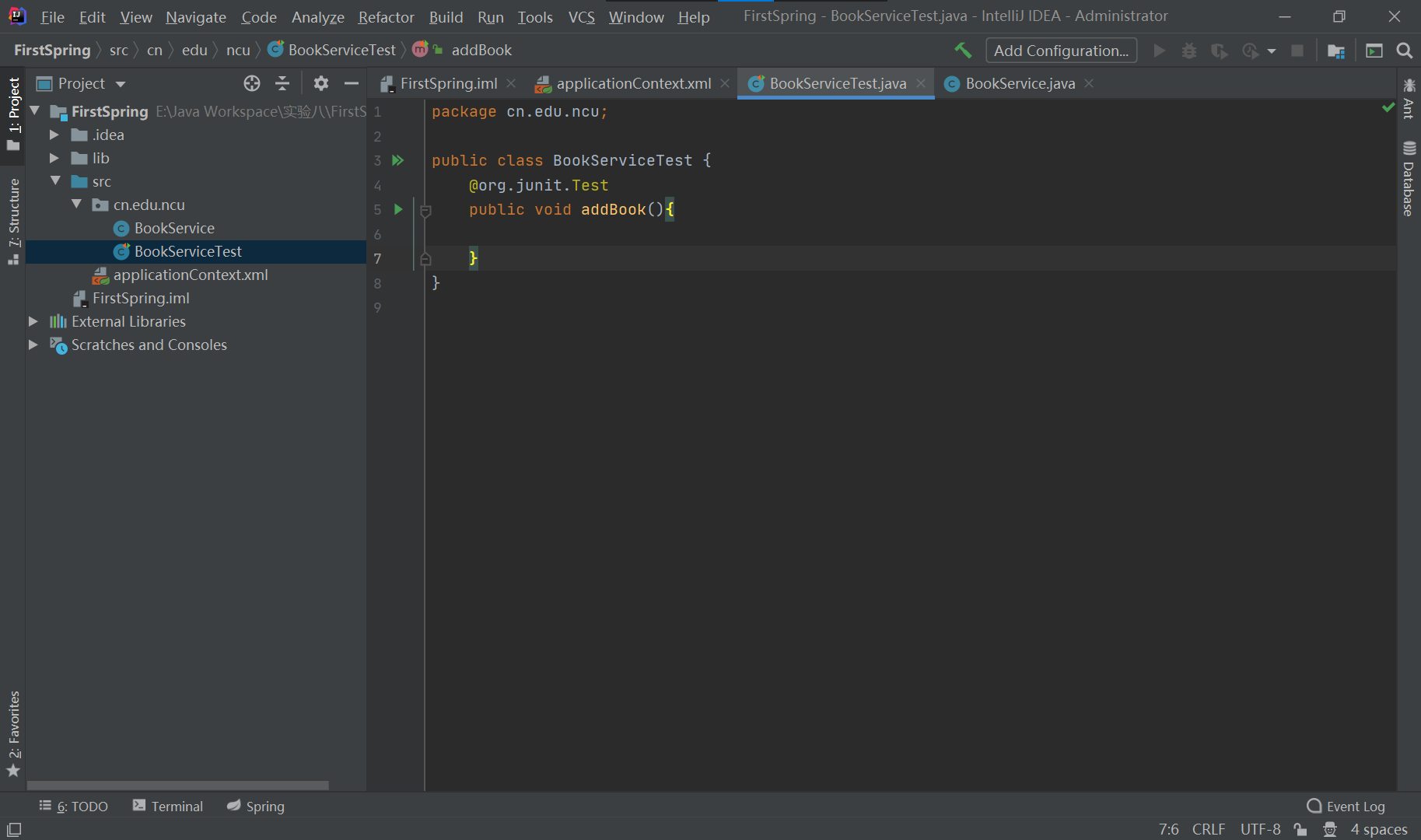
使用Spring搭建项目框架，并熟悉SpringBean的属性。

1. 实验基本原理
2. Spring框架的安装
3. 使用Spring框架构建项目
4. 主要仪器设备及耗材

计算机、Idea、Tomcat9.0、JDK1.8

五、实验步骤（完整内容见光盘）

1.Spring安装和项目创建，创建完成后项目结构如图所示：

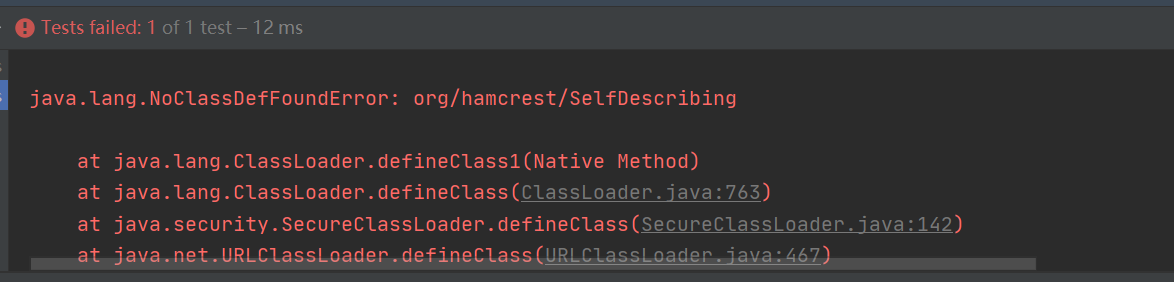


2.Spring Bean管理

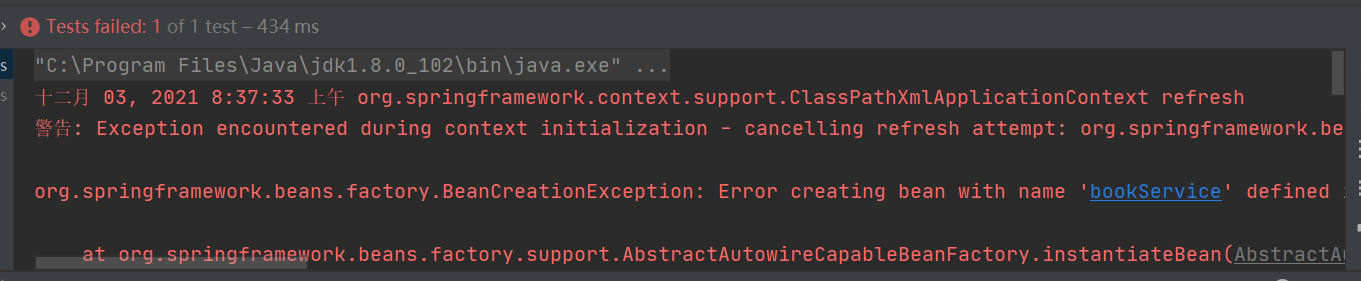
SpringBean三种实例化方式

（1）构造函数方法

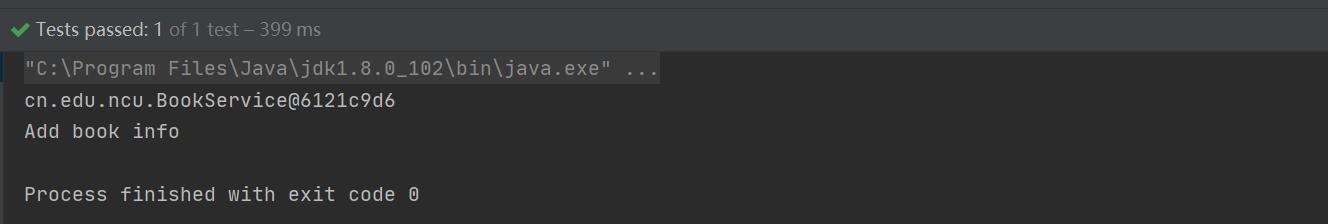
i.使用无参数构造方法在BookService类中添加一个带参数的构造方法，执行单元测试



提示错误，经过上网查询得知新版本4.12版本的junit还需导入hamcrest-core-1.3.jar，导入成功后再测试：



提示错误，因为没有对应的无参构造函数，添加无参构造方法后再执行单元测试方法。



可以看到，添加无参构造函数后能成功运行。

使用这种方法:

要实例化的类中如果有构造器的话，一定要有一个无参的构造器。

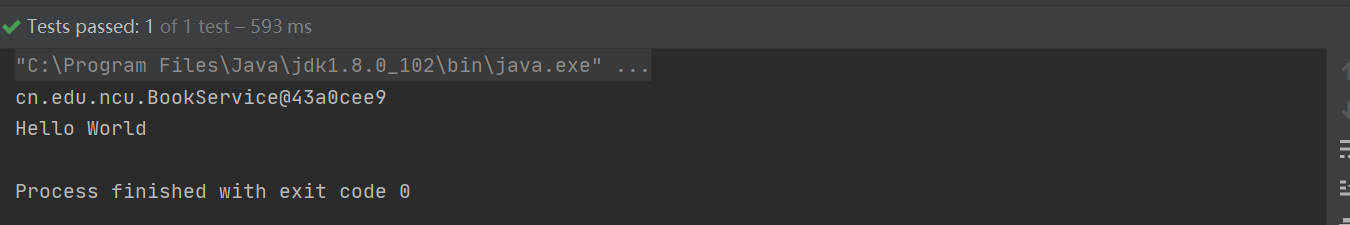
ii.有参构造

修改xml内容，指定构造函数参数

xml内容如下所示：

1. <bean id="userService" class="cn.edu.ncu.BeanFactory" factory-method="getUserService"></bean>

调用测试方法得到如下结果：



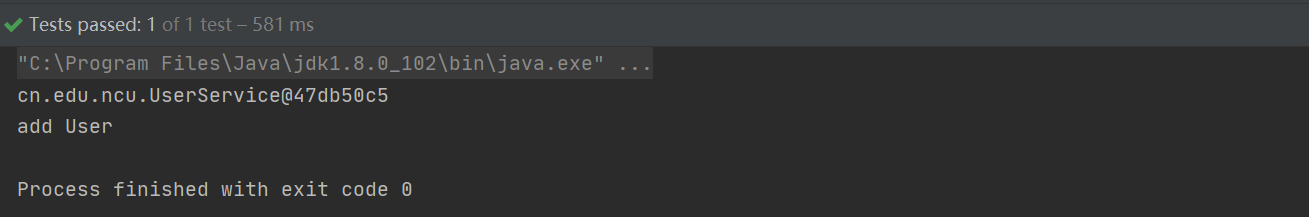
（2）使用静态工厂创建

新建UserService类和BeanFactory类，在BeanFactory中创建静态方法getUserService返回UserService对象，修改xml文件如下所示：

修改测试方法

1. public void addBook(){
2. ApplicationContext context =  new ClassPathXmlApplicationContext("applicationContext.xml");
3. UserService service = (UserService) context.getBean("userService");
4. System.out.println(service);
5. service.addUser();
6. }

运行测试方法得到如下结果：



（3）使用实例工厂创建

新建CategoryService类，在BeanFactory中创建方法get CategoryService返回CategoryService对象

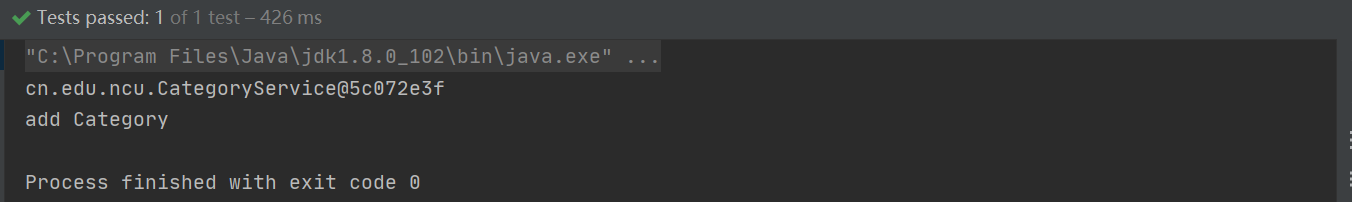
在xml文件中添加如下配置：

1. <bean id="beanFactory" class="cn.edu.ncu.BeanFactory"></bean>
2. <bean id="categoryService" factory-bean="beanFactory" factory-method="getCategoryService"></bean>

修改测试类如下所示：

1. public void addBook(){
2. ApplicationContext context =  new ClassPathXmlApplicationContext("applicationContext.xml");
3. CategoryService service = (CategoryService) context.getBean("categoryService");
4. System.out.println(service);
5. service.addCategory();
6. }

运行测试方法得到如下运行结果：



3. Bean的作用范围

（1）Singleton(单例)

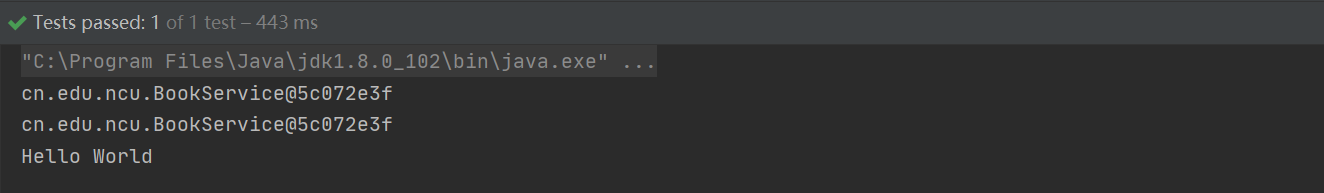
修改BookServiceTest测试方法中的addBook方法，增加一个service的实例；

打印了两个类的hashcode，可以看到是一致的，表明这两个实例实际是一个。

修改后的测试方法如下所示：

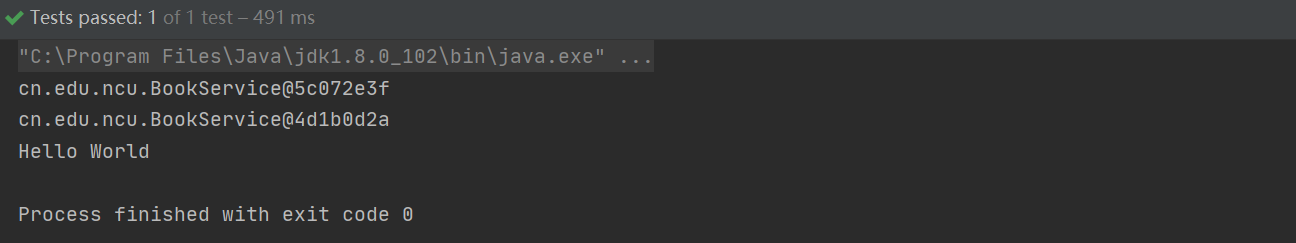
1. public void addBook(){
2. ApplicationContext context =  new ClassPathXmlApplicationContext("applicationContext.xml");
3. BookService service = (BookService) context.getBean("bookService");
4. BookService service2 = (BookService) context.getBean("bookService");
5. System.out.println(service);
6. System.out.println(service2);
7. service.addBook();
8. }

运行该测试方法的结果如下所示：



（2）prototype（多例）

修改xml中BookService的scope为prototype，然后同样执行上述测试方法，得到如下结果：



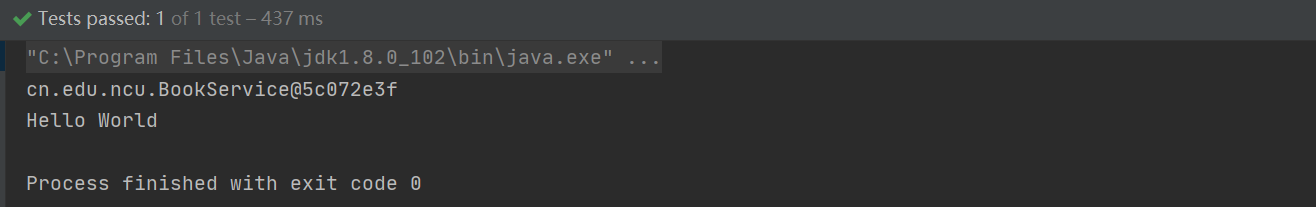
可以看到不是同一个对象，说明多例模式下每次获取到的对象不相同。

4．Spring的属性注入

（1）有参构造

通过xml指定构造函数参数

执行结果如下所示：



（2）Setter方法

新建一个book的实体bean

该类代码如下所示：

1. package cn.edu.ncu;
2. public class Book {
3. private String bookID;
4. private String bookName;
5. public String getBookID() {
6. return bookID;
7. }
8. public void setBookID(String bookID) {
9. this.bookID = bookID;
10. }
11. public String getBookName() {
12. return bookName;
13. }
14. public void setBookName(String bookName) {
15. this.bookName = bookName;
16. }
17. }

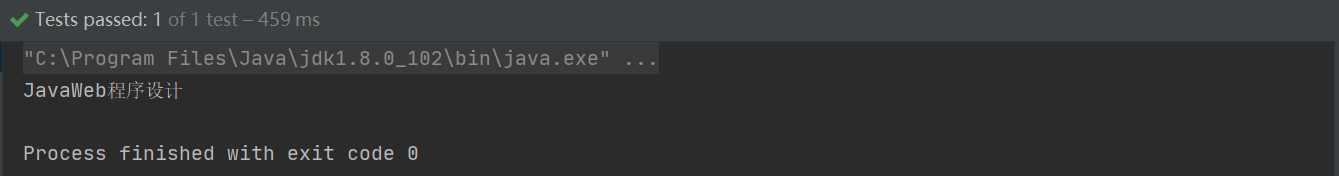
修改xml文件，将该类在Spring容器中注册，xml中增加如下配置：

1. <bean id="book" class="cn.edu.ncu.Book">
2. <property name="bookName" value="JavaWeb程序设计"></property>
3. </bean>

增加测试类如下所示：

1. @Test
2. public void testBookVo(){
3. ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
4. Book book = (Book)context.getBean("book");
5. System.out.println(book.getBookName());
6. }

运行测试类得到如下结果：



5.Spring Bean注入

实际项目中Controller调用Service，Service调用Dao等等，在代码中需要注入相应的bean，实现类之间的解耦。

（1）setter方法注入

先创建一个bookDao ，为了方便只是做了一个控制台输出。

增加BookDao类代码如下：

1. public class BookDao {
2. public void deleteBookByID(String bookID){
3. System.out.println("delete id='"+bookID+"'book info");
4. }
5. }

修改BookService类代码如下：

1. public class BookService {
2. private String bookID;
3. private BookDao bookDao;
4. public void setBookDao(BookDao bookDao) {
5. this.bookDao = bookDao;
6. }
7. public BookService() {
8. }
9. public BookService(String bookID) {
10. this.bookID = bookID;
11. }
12. public void addBook(){
13. System.out.println(bookID);
14. }
15. public void deleteBookByID(String ID){
16. bookDao.deleteBookByID(ID);
17. }
18. }

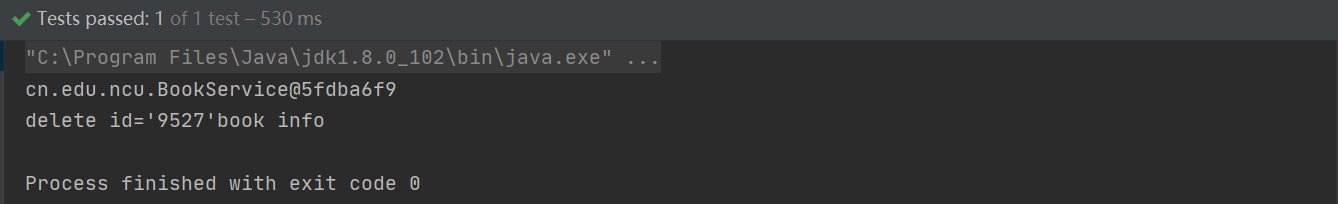
修改bean的配置文件如下：

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
5. <bean id="bookService" class="cn.edu.ncu.BookService" scope="prototype">
6. <constructor-arg index="0" value="Hello World"/>
7. <property name="bookDao" ref="bookDao"></property>
8. </bean>
9. <bean id="bookDao" class="cn.edu.ncu.BookDao"></bean>
10. <bean id="beanFactory" class="cn.edu.ncu.BeanFactory"></bean>
11. <bean id="categoryService" factory-bean="beanFactory" factory-method="getCategoryService"></bean>
12. <bean id="book" class="cn.edu.ncu.Book">
13. <property name="bookName" value="JavaWeb程序设计"></property>
14. </bean>
15. </beans>

增加测试方法如下所示：

1. @Test
2. public void testBookVo(){
3. ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
4. Book book = (Book)context.getBean("book");
5. System.out.println(book.getBookName());
6. }

执行该测试方法，得到如下运行结果：



由此可知此种方式注入成功了！

（2）注解方法注入

新建一个spring项目

在xml文件中打开自动注解扫描，修改xml文件如下所示：

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
6. <context:component-scan base-package="cn"/>
7. </beans>

新建BookDao和BookService，代码如下所示

**BookDao**

1. package cn.edu.ncu;
2. import org.springframework.stereotype.Repository;
3. @Repository
4. public class BookDao {
5. public void addBook(){
6. System.out.println("add book info");
7. }
8. }

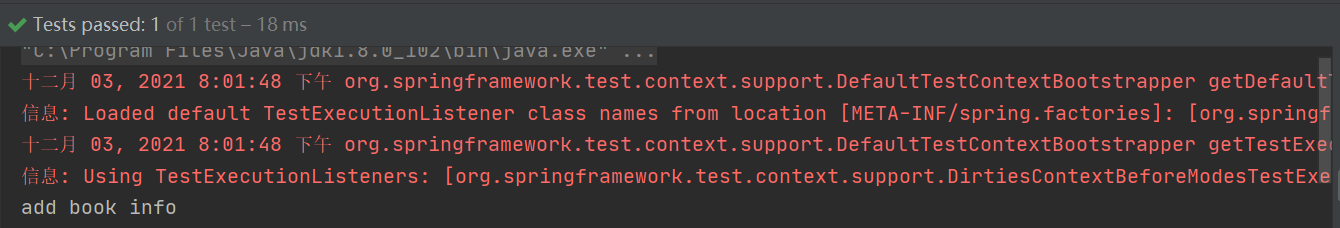
**BookService**

1. package cn.edu.ncu;
2. import org.springframework.beans.factory.annotation.Autowired;
3. import org.springframework.stereotype.Service;
4. @Service
5. public class BookService {
6. @Autowired
7. private  BookDao bookDao;
8. public void addBook(){
9. bookDao.addBook();
10. }
11. }

**CommonTest**

1. package cn.edu.ncu;
2. import org.junit.Test;
3. import org.junit.runner.RunWith;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import org.springframework.test.context.ContextConfiguration;
6. import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
7. @RunWith(SpringJUnit4ClassRunner.class)
8. @ContextConfiguration("classpath:applicationContext.xml")
9. public class CommonTest {
10. @Autowired
11. private BookService service;
12. @Test
13. public void addBook(){
14. service.addBook();
15. }
16. }

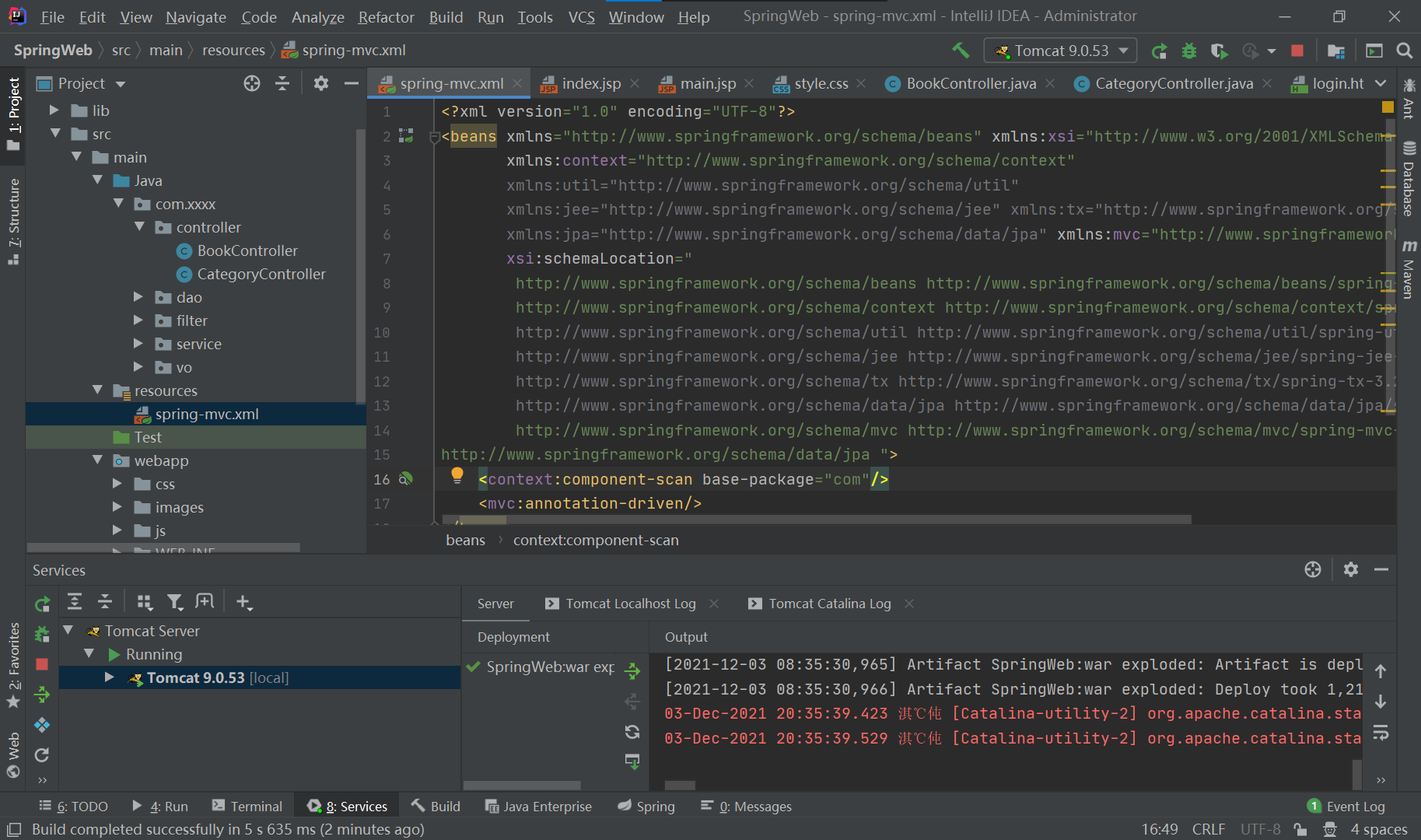
运行测试方法，结果如下：



六、思考讨论题或体会或对改进实验的建议

1．修改实验7中的项目使用spring框架，实现多个模块之间的解耦。

可以使用maven来搭建Spring的Web项目，使用注解方式进行开发，搭建截图如下所示



将对象都交由Spring容器托管，例如

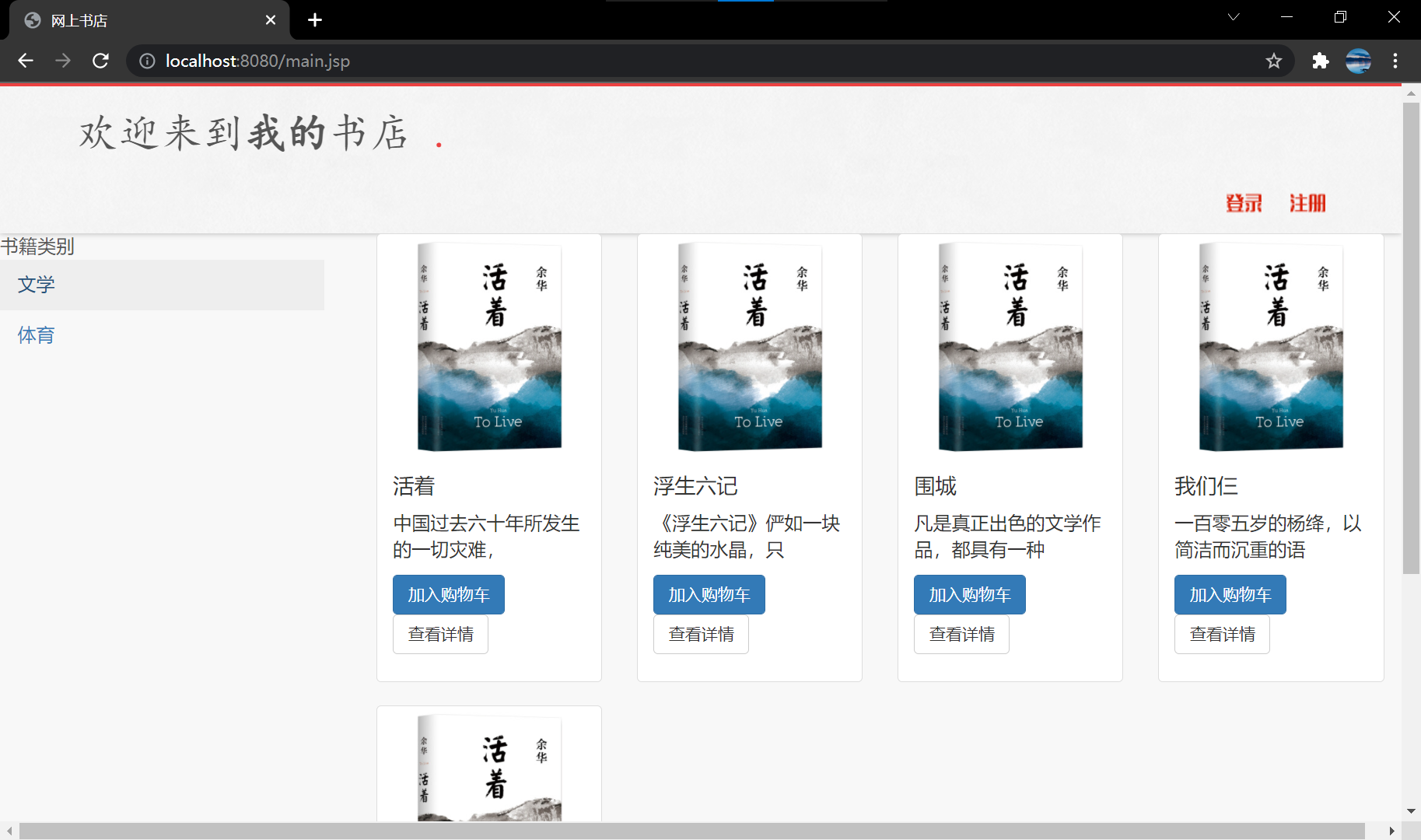
**CategoryService**

1. package com.xxxx.service;
2. import com.xxxx.dao.CategoryDao;
3. import com.xxxx.dao.implPackage.\*;
4. import com.xxxx.vo.Category;
5. import org.springframework.beans.factory.annotation.Autowired;
6. import org.springframework.stereotype.Service;
7. import javax.xml.ws.ServiceMode;
8. import java.util.List;
9. @Service
10. public class CategoryService {
11. @Autowired
12. private CategoryDao category;
13. public List<Category> findAll() {
14. return category.findAll();
15. }
16. }

**Book Service**

1. package com.xxxx.service;
2. import com.xxxx.dao.BookDao;
3. import com.xxxx.dao.implPackage.BookDaoImpl;
4. import com.xxxx.vo.Book;
5. import org.springframework.beans.factory.annotation.Autowired;
6. import org.springframework.stereotype.Service;
7. import java.util.List;
8. @Service
9. public class BookService {
10. @Autowired
11. private BookDao book;
12. public List<Book> getBookByCategoryID(int category\_id) {
13. return book.getBookByCategoryID(category\_id);
14. }
15. }

最终运行结果如下：



七、参考资料

1.JavaWeb程序设计