# 实验四 Spring的IoC反转控制

一、实验目的：

理解Spring的IoC容器工作原理，通过该容器管理各组件之间的依赖关系来降低组件之间的耦合度。

二、实验内容：

在MyEclipse环境下创建Java项目实现反转控制，调试运行程序。

三、实验要求：

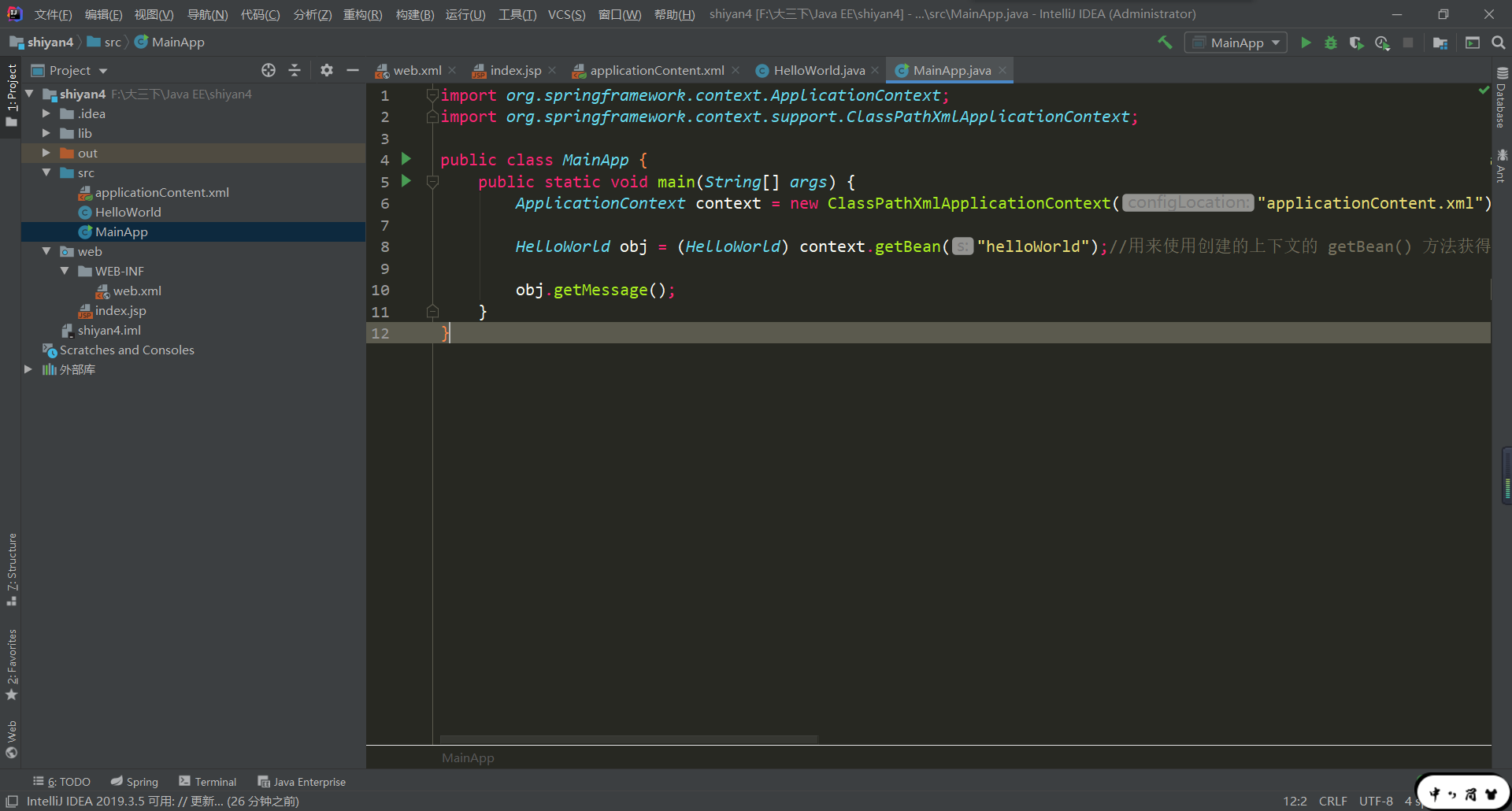
1. 熟悉并掌握创建Java项目的基本步骤。

2. 运用依赖注入方式实现反转控制。

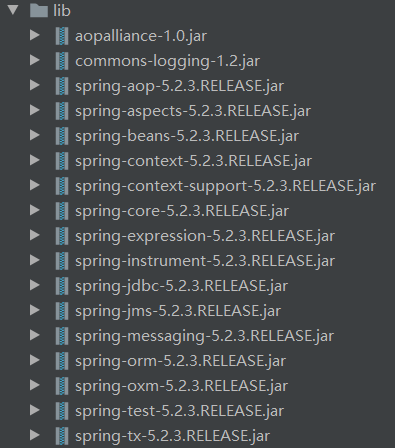
3. 写出实验报告。

四、实验步骤：

1．进入MyEclipse环境，新建一个Java Project。



1. 导入Spring Java包。

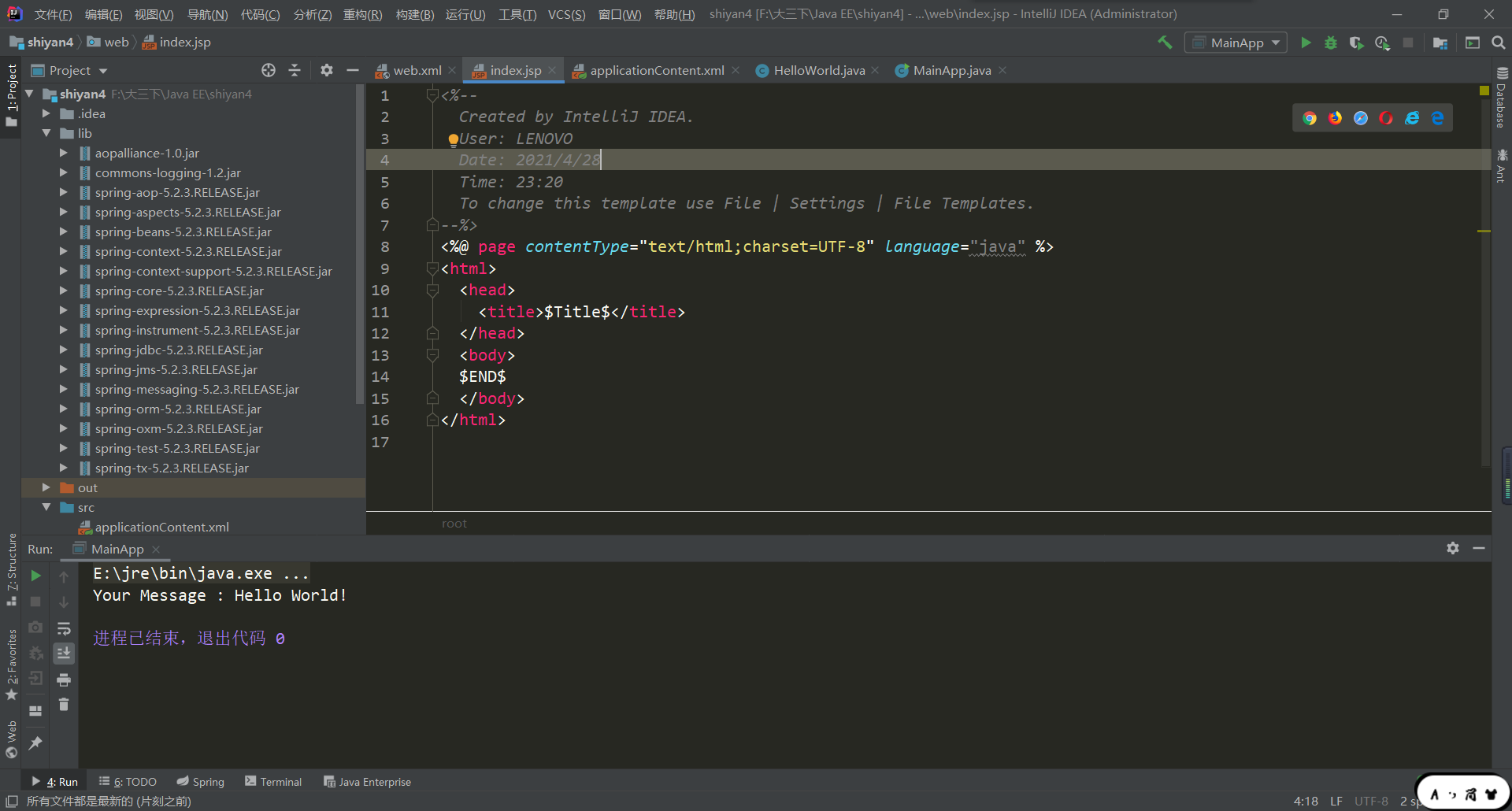


1. 编写程序，并给出关键代码及注释。

import *org.springframework.context.ApplicationContext*;  
import *org.springframework.context.support.ClassPathXmlApplicationContext*;  
  
public class *MainApp* {  
 public static void main(*String*[] *args*) {  
 *ApplicationContext* context = new ClassPathXmlApplicationContext("applicationContent.xml");//在这里我们使用框架API ClassPathXmlApplicationContext() 应用程序上下文。这个API加载 applicationContext 的配置文件，并最终基于所提供的API，它需要创建并初始化所有的对象。在配置文件中提到的beans 类。  
  
 *HelloWorld* obj = (*HelloWorld*) context.getBean("helloWorld");//用来使用创建的上下文的 getBean() 方法获得所需的bean。此方法使用 bean 的 id 返回，最终可以创建实际对象的通用对象。一旦有了对象，就可以使用这个对象调用任何类方法。  
  
 obj.getMessage();  
 }  
}

public class *HelloWorld* {  
 private *String* message;  
  
 public void setMessage(*String message*){  
 this.message = *message*;  
 }  
  
 public void getMessage(){  
 *System*.***out***.println("Your Message : " + message);  
 }  
}

1. 程序运行截图。



五、心得体会：

Ioc比以前更好使用，实现了资源的统一管理，实现资源的可配置和以管理，降低了使用资源双方的依赖程度，也就是耦合度，便于项目的扩展，易于维护，在ioc+接口的情况下，删除任何实现类都不会导致程序编译出错，虽然运行到特定的代码会报错，但是其他代码在使用时不会有问题。