# 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。

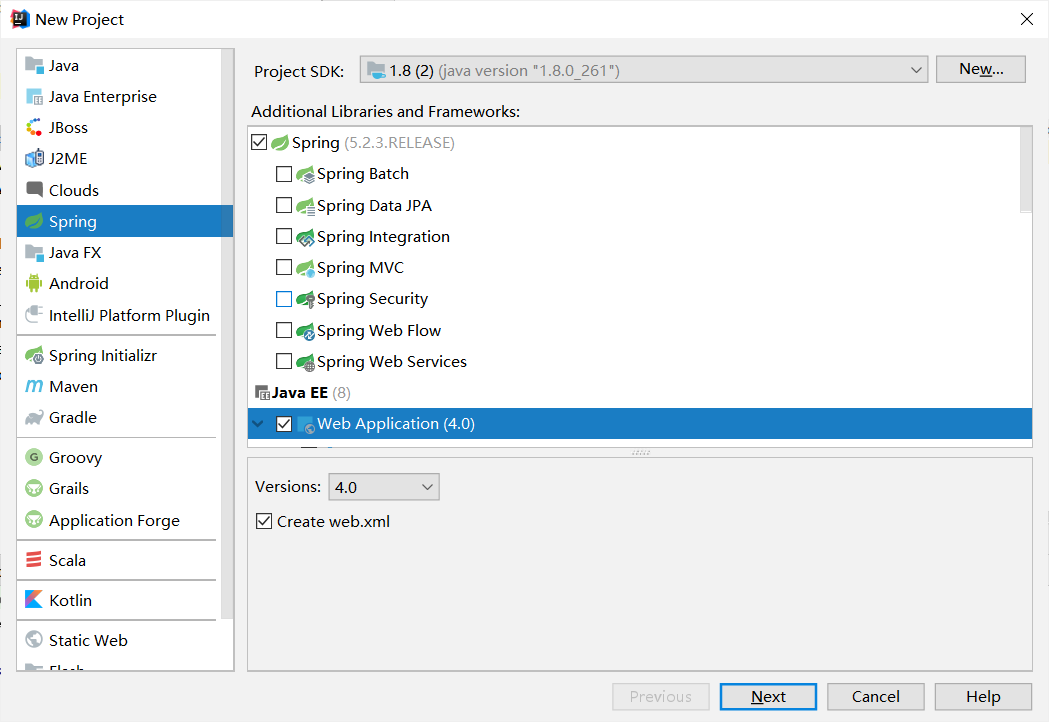
2. 运用依赖注入方式实现反转控制。

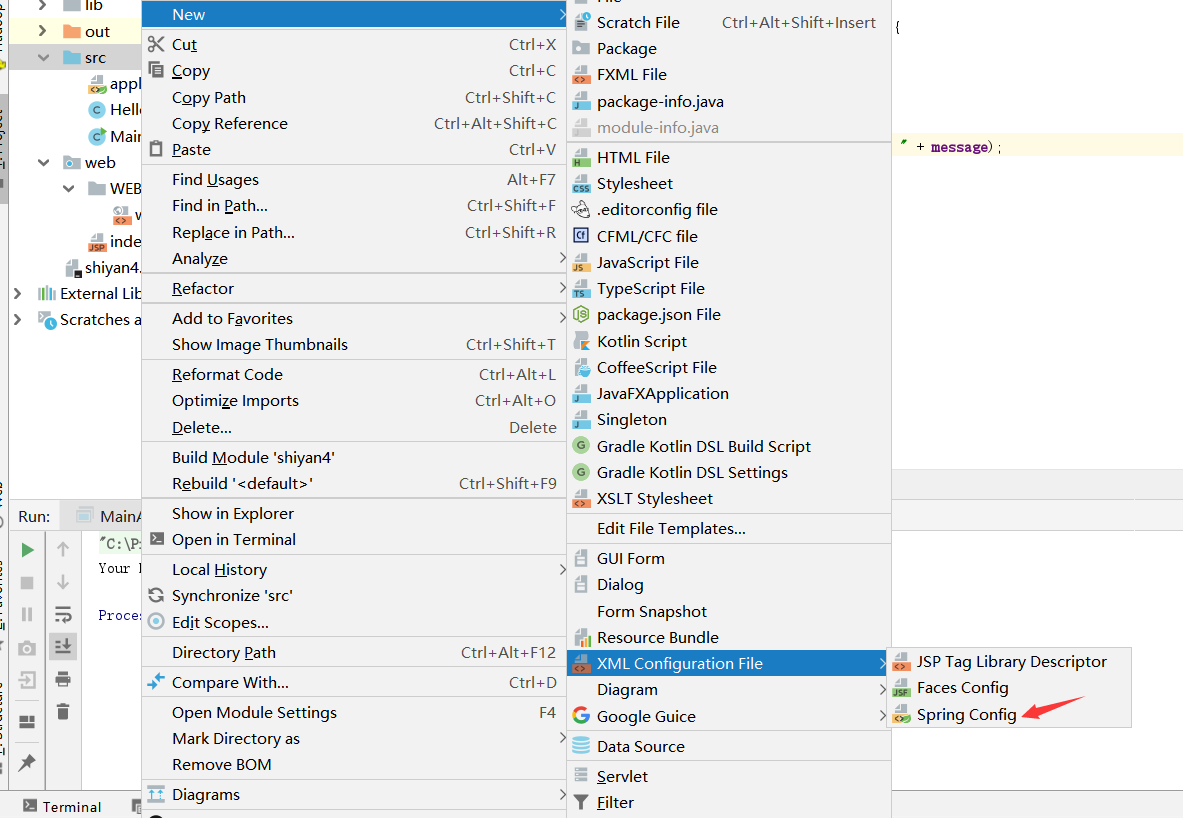
3. 写出实验报告。

四、实验步骤：

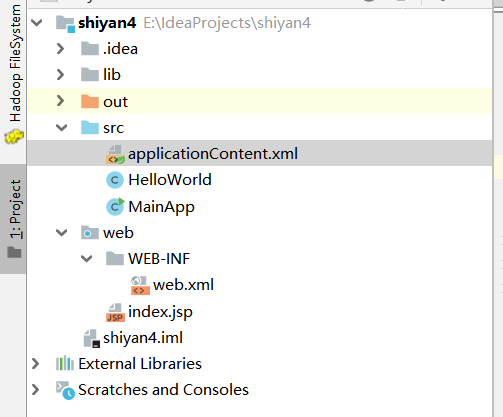
1．进入MyEclipse环境，新建一个Java Project。

我这里用的是IDEA环境，新建Java Enterprise项目





程序目录：



1. 编写程序，并给出关键代码及注释。

applicationContent.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"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"**>  
  
 <**bean id="helloWorld" class="HelloWorld"**>  
 <**property name="message" value="Hello World!"**/>  
 </**bean**>  
</**beans**>

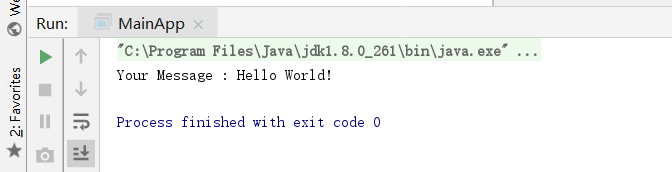
HelloWorld.java

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

MainApp.java

**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();  
 }  
}

1. 程序运行截图。



1. 心得体会：

理解了Spring的IoC容器工作原理：原本需要程序去主动new创建的对象，现在反转过来交给spring的容器去创建。简便了开发流程，提高了效率。