

第3章 Spring Boot框架初识

广东财经大学信息学院

罗 东 俊 博士

ZSUJONE@126.COM

(内部资料，请勿外传)



目的和要求

- 了解**Spring Boot**的发展和优点。
- 掌握**Spring Boot**的项目构建。
- 掌握**Spring Boot**的单元测试和热部署。
- 掌握**Spring Boot**项目的打包和部署。



主要内容

- 3.1 Spring Boot概述
- 3.2 Spring Boot入门程序
- 3.3 单元测试和热部署
- 3.4 Spring Boot应用的打包和部署



3.1 Spring Boot概述

- 3.1.1 Spring Boot简介
- 3.1.2 Spring Boot的主要特性
- 3.1.3 Spring Boot的优点



3.1.1 Spring Boot简介

- **Spring Boot**是基于**Spring**框架开发的全新框架，其设计目的是简化新**Spring**应用的初始化搭建和开发过程。
- **Spring Boot**整合了许多框架和第三方库配置，几乎可以达到“开箱即用”。



3.1.2 Spring Boot的主要特性

■ 1.约定优先配置（convention over configuration）

✧ **Spring Boot**实现了与传统**Spring**以及其他常用第三方库的整合连接，提供了默认最优化的整合配置，大多数情况直接使用默认配置即可（基本上不需要额外生成配置代码和**XML**配置文件）。

■ 2. **Java Config**配置方式，实现自动配置

✧ 以**Java**配置类取代传统的**XML**配置文件

✧ 通过依赖启动器（**starters**）简化构建配置，一旦引入某个场景的依赖启动器，相应的自动化配置类和相关的**JAR**包即被生效



3.1.3 Spring Boot的优点

■ 1.使编码变得简单

✧推荐使用注解。

■ 2.使配置变得快捷

✧自动配置、快速构建项目、快速集成第三方技术的能力。

■ 3.使部署变得简便

✧内嵌Tomcat、Jetty、Undertow等Web容器（无需部署WAR文件）

■ 4.使监控变得容易

✧提供基于http、ssh、telnet对运行的项目进行跟踪监控。



3.2 Spring Boot入门程序

■ 3.2.1 环境准备

■ 3.2.2 构建Spring Boot项目



3.2.1 环境准备

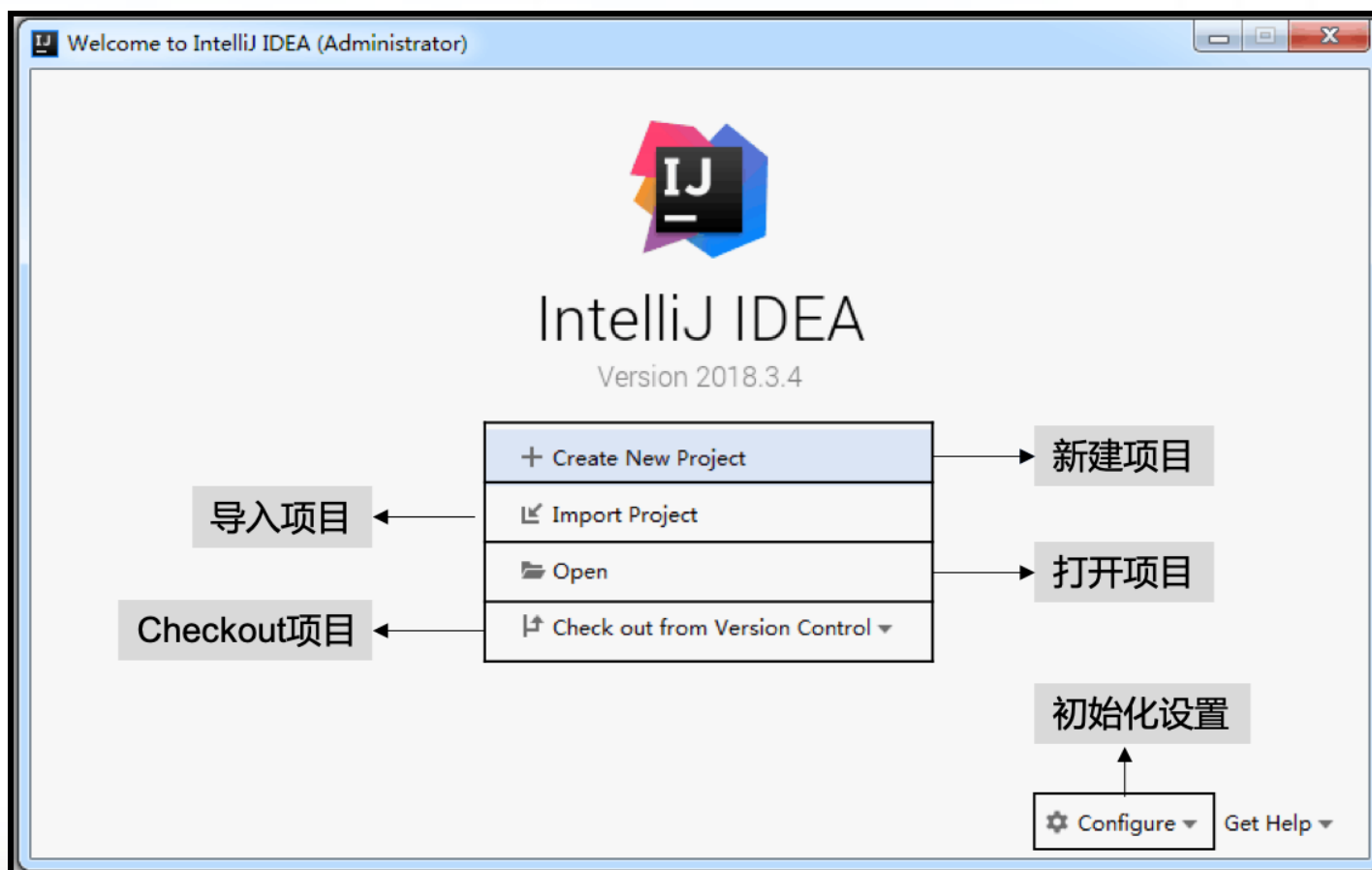
■ 保证安装好的软件如下：

✧ **JDK 1.8.0_201**（及以上版本）

✧ **IntelliJ IDEA Ultimate 2018**旗舰版

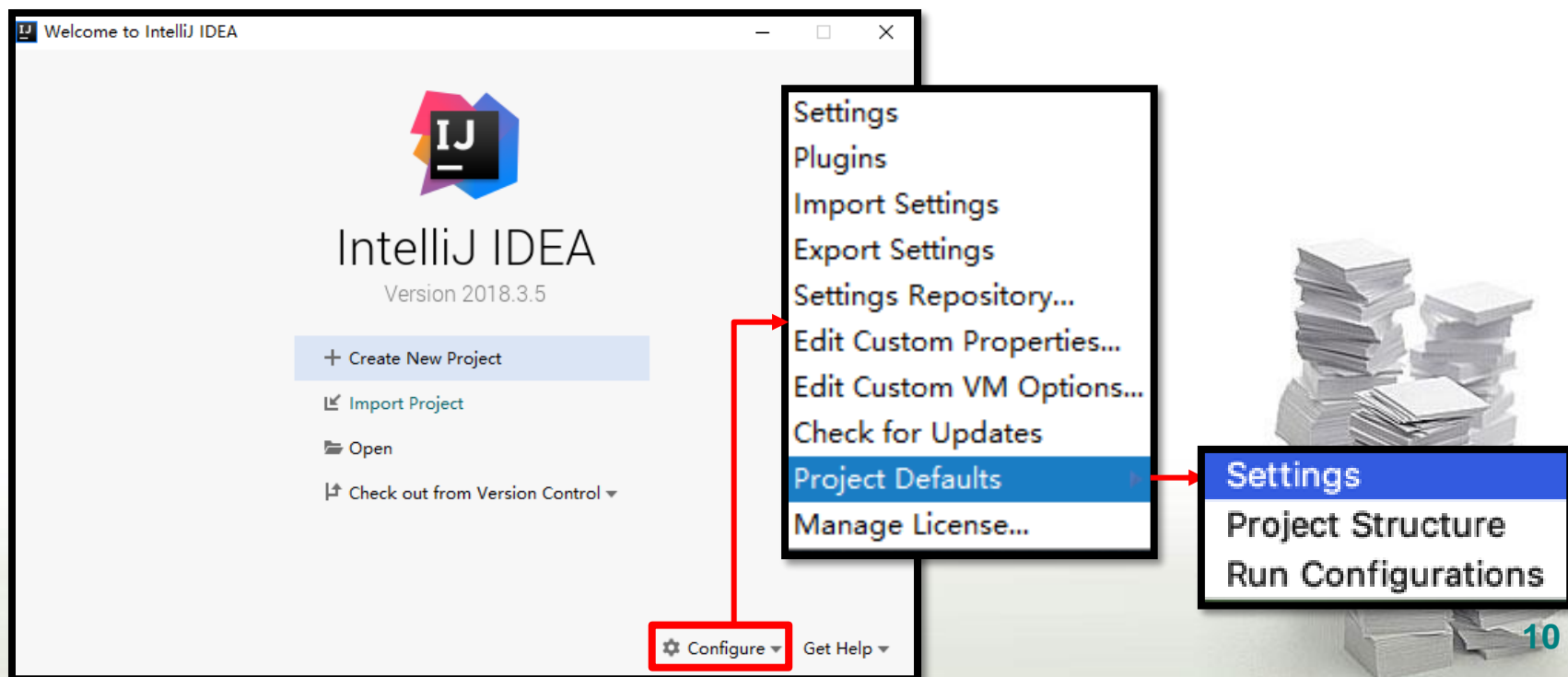


1.认识IDEA欢迎页



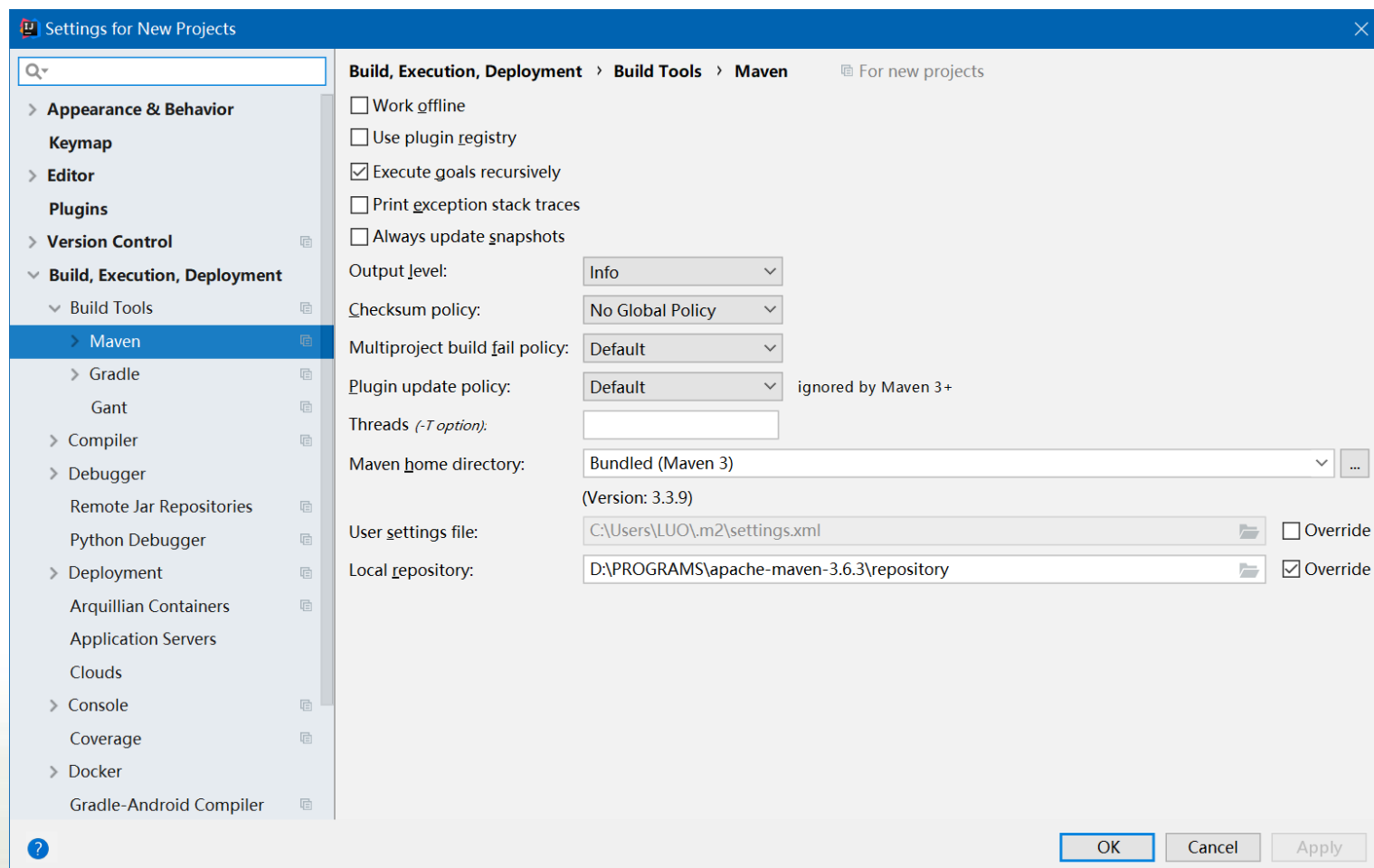
2.初始化Maven设置

- (1) 单击【Configure】→【Project Defaults】→【Settings】进入设置Maven界面



初始化Maven设置

■ (2) 初始化Maven设置



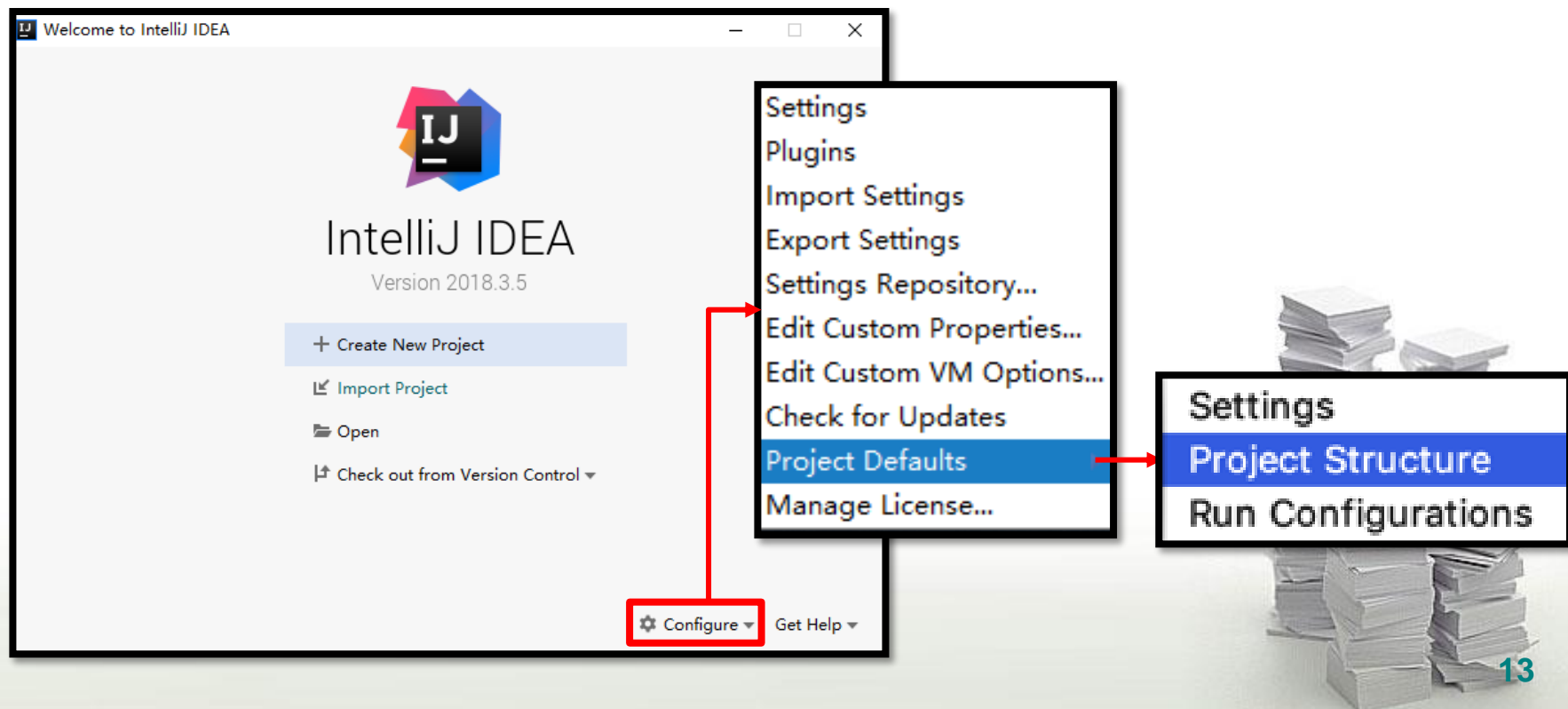
关于Maven

- **Apache Maven**用于统一开发规范与工具和统一管理项目开发所需的jar包，这些包存放在本地仓库中（默认C:\Users\<用户名>\.m2\repository）或先从中央仓库（<http://mvnrepository.com>或映像）自动下载到本地仓库。
- **Maven**基于POM(Project Object Model)理念管理软件项目，Maven项目都有一个pom.xml配置文件来管理项目的依赖和项目的编译等功能。



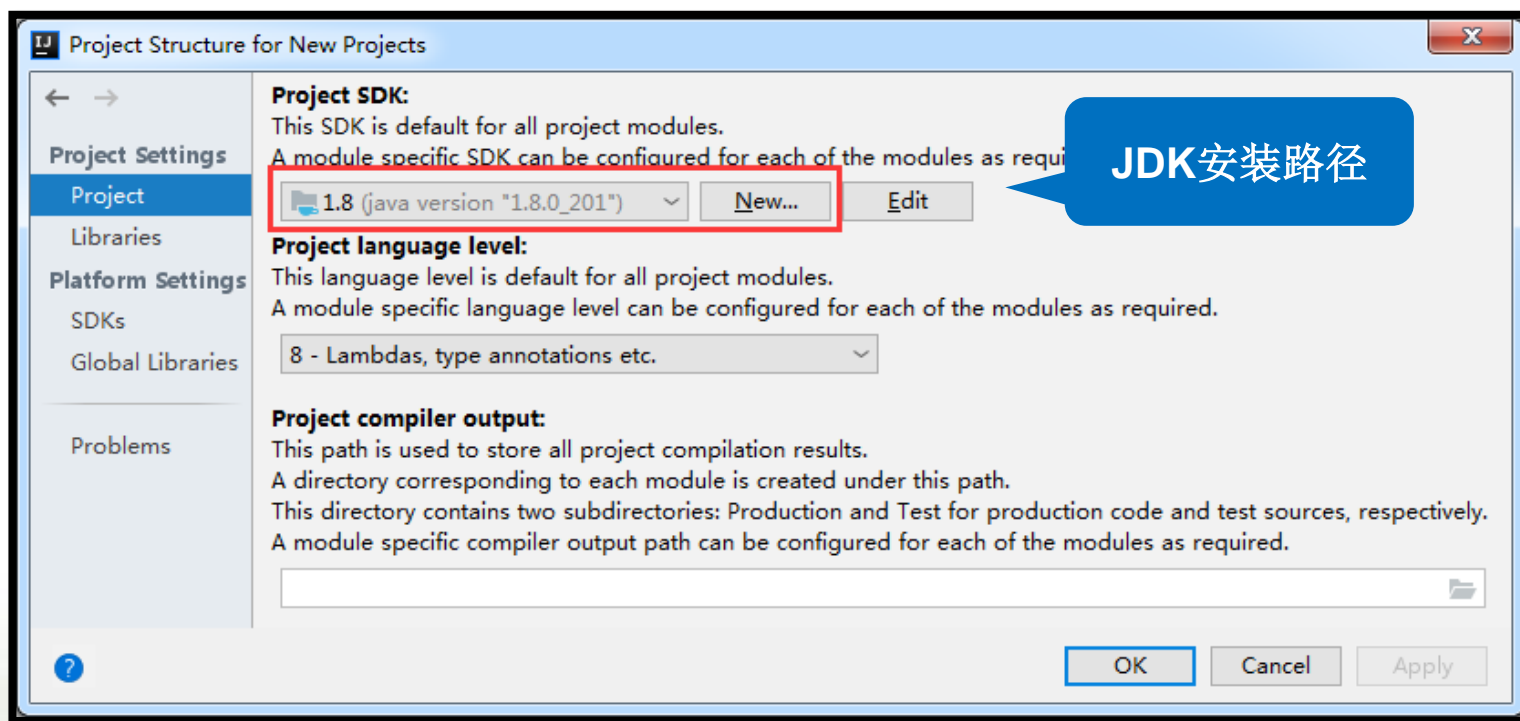
3.初始化JDK设置

- (1) **【Configure】** → **【Project Defaults】** → **【Project Structure】** 进入JDK设置页



初始化JDK设置

- (2) 在界面左侧选择【Project Settings】→【Project】选项



3.2.2 构建Spring Boot项目

- 1. 使用Spring Initializr方式构建
- 2. 使用Maven方式构建



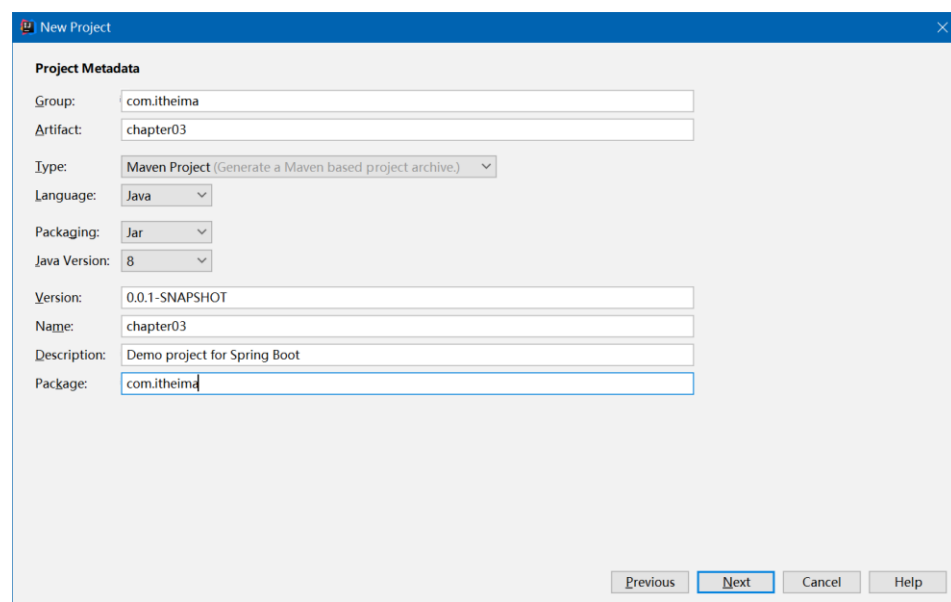
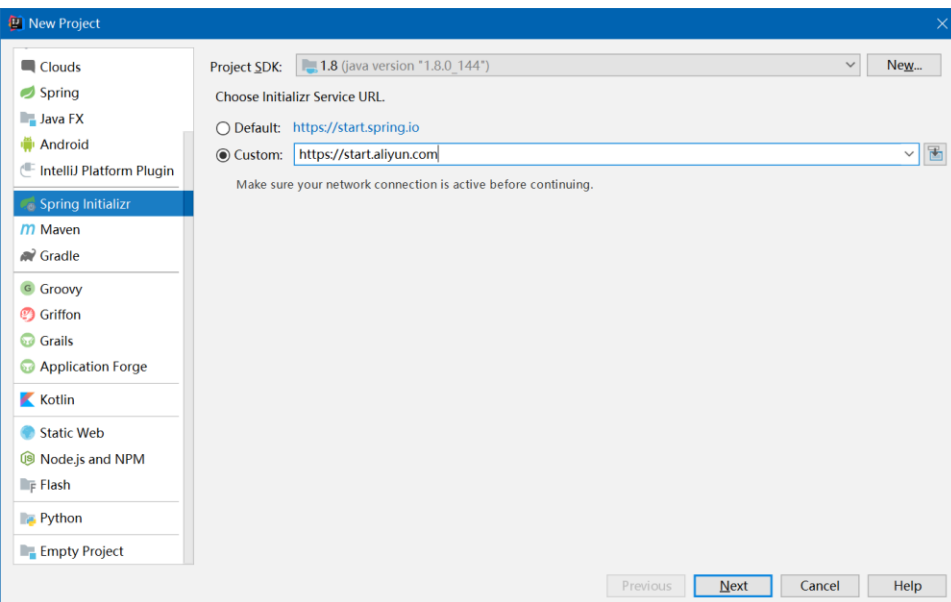
1. 使用Spring Initializr方式构建

■ 搭建步骤:

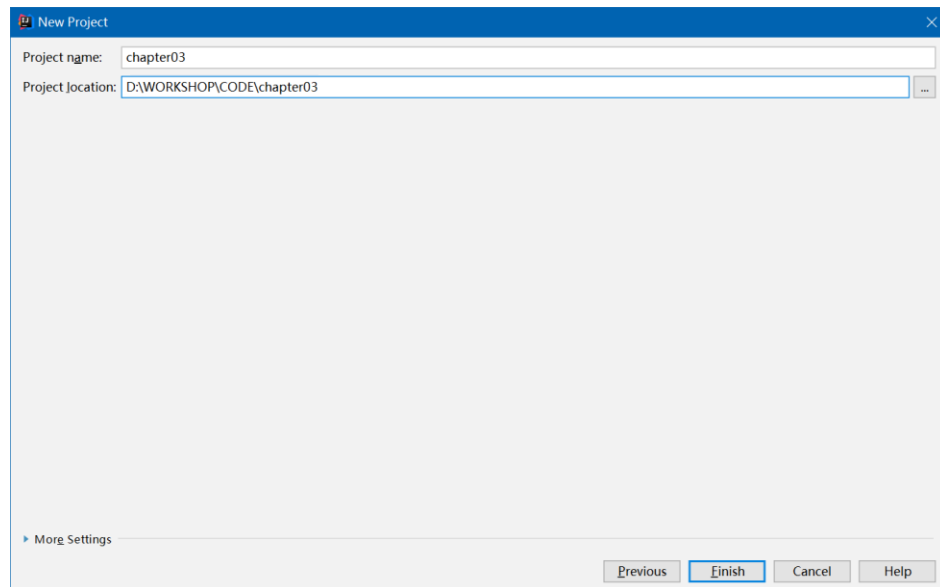
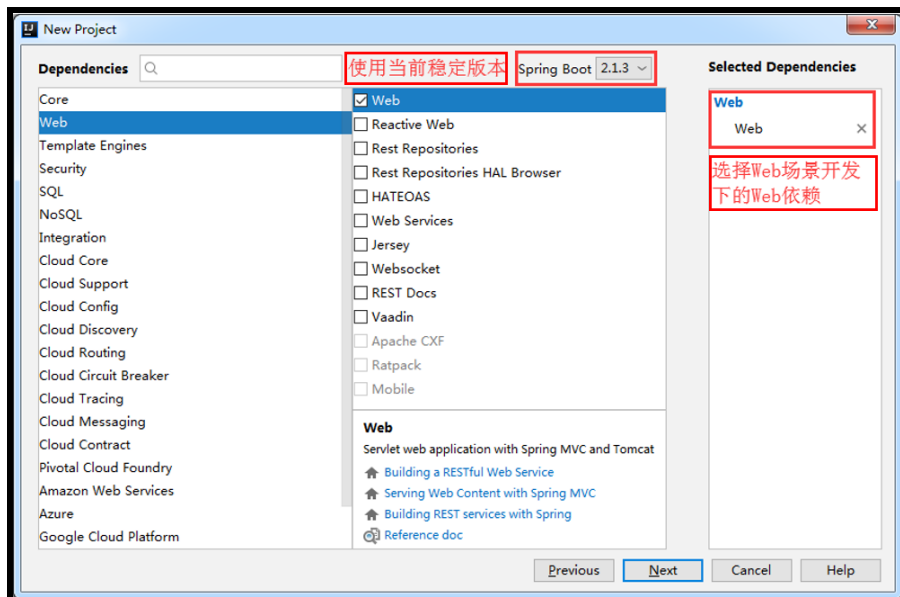
- ✧①创建**Spring Boot**项目
- ✧②创建一个用于**Web**访问的**Controller**
- ✧③运行项目



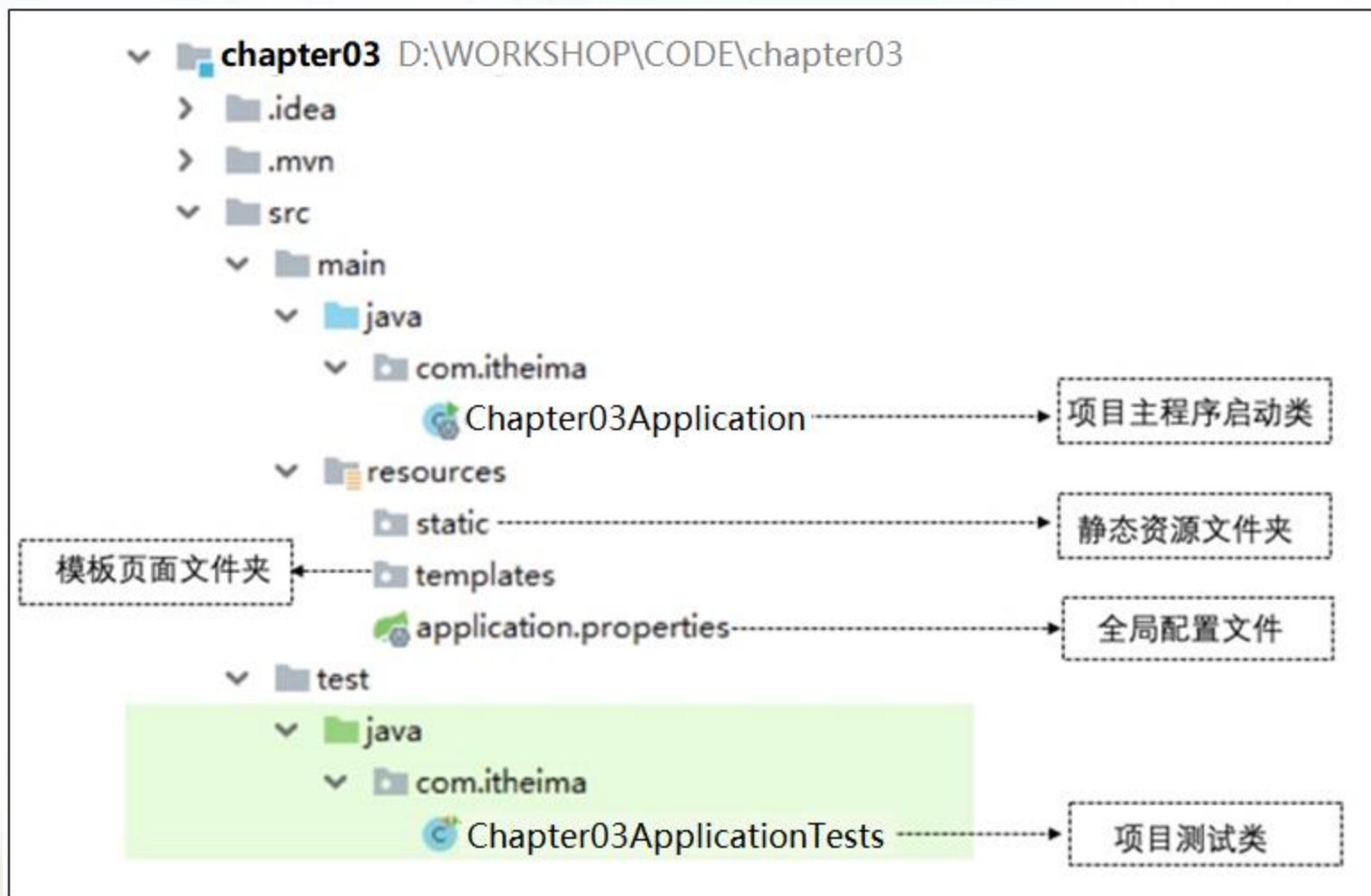
①创建Spring Boot项目



创建Spring Boot项目



创建好的项目结构



②创建Controller

```
package com.itheima.controller;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
```

该注解为组合注解，等同于Spring中
@Controller+ @ResponseBody注解

```
public class HelloController {
```

```
@GetMapping("/hello")
```

```
    public String hello(){  
        return "hello Spring Boot";  
    }  
}
```

等同于Spring框架中
@RequestMapping(RequestMethod.GET)注解

HelloController: web请求处理控制类;

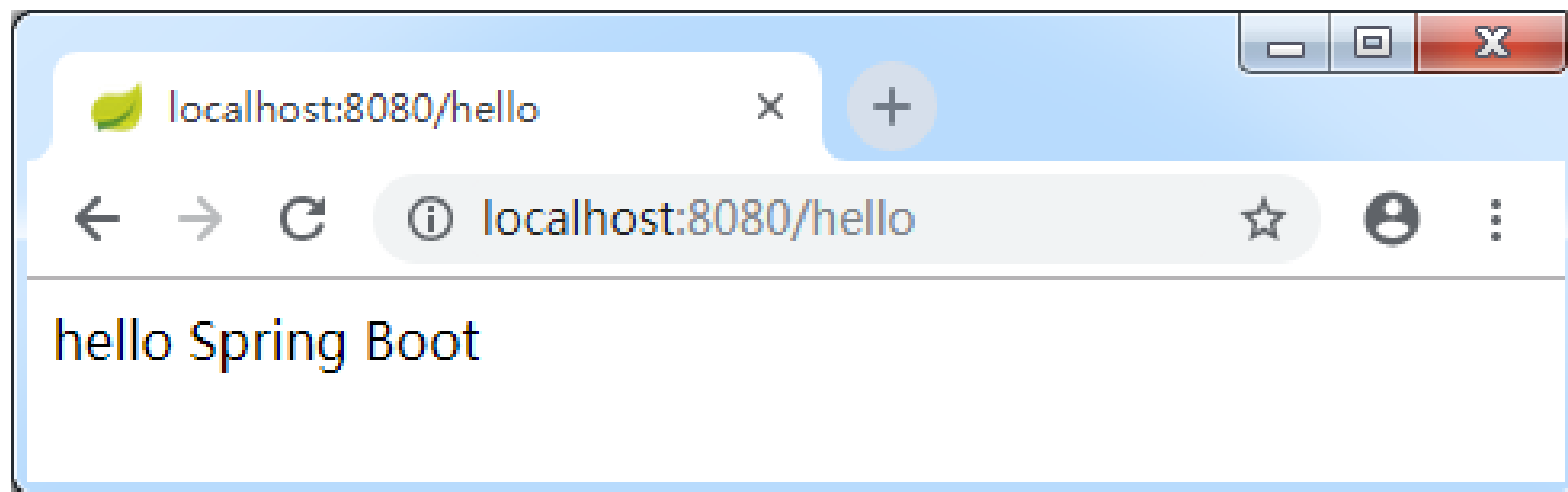
@RestController: 将当前类作为控制层组件添加到Spring容器（ApplicationContext）中，同时当前类的处理方法会返回JSON字符串;

@GetMapping: 设置方法的web访问路径并限定其访问方式是Get。



③运行项目

- 启动项目，在浏览器上访问
`http://localhost:8080/hello`



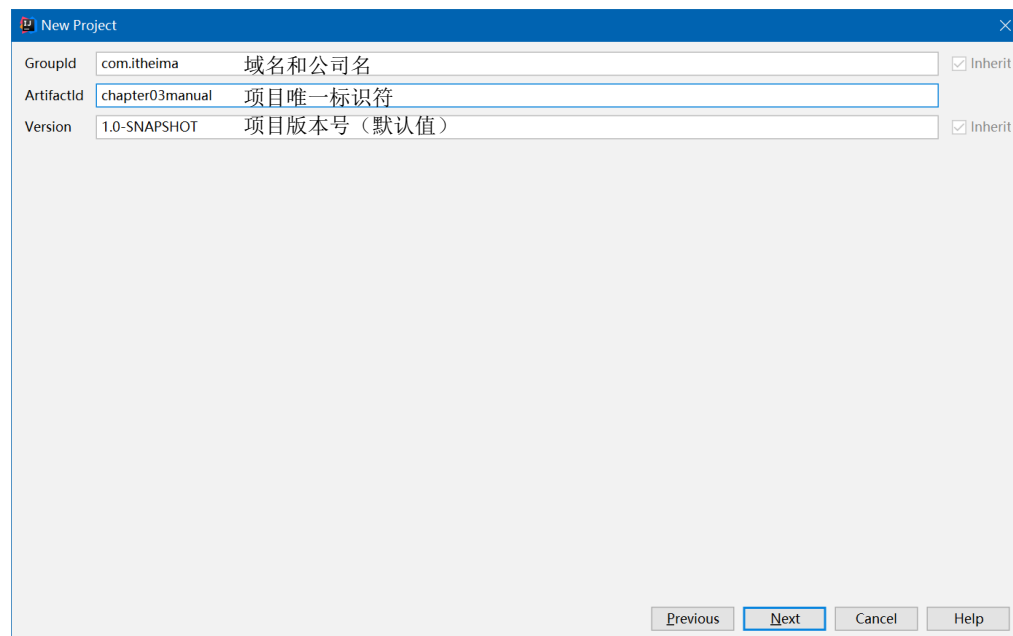
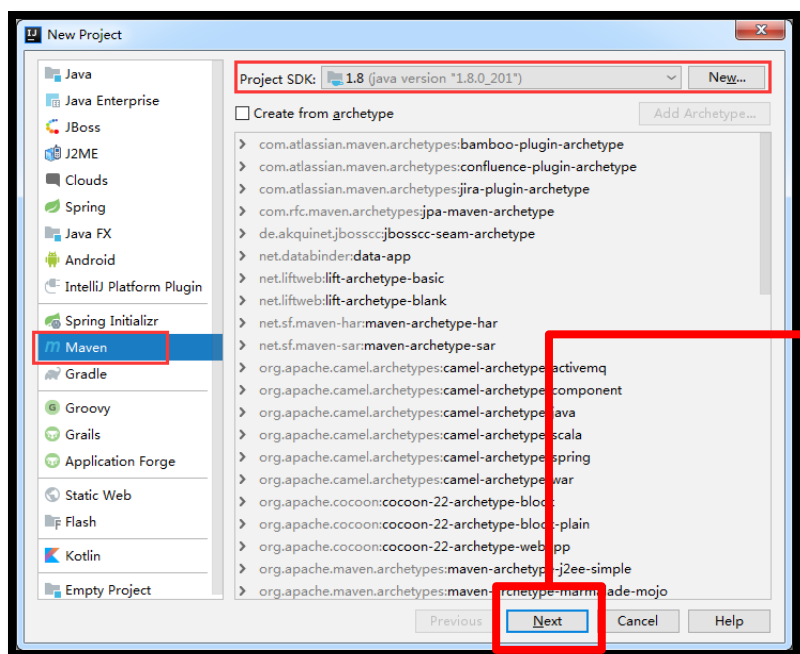
2. 使用Maven方式构建

■ 搭建步骤:

- ✧ ① 创建Maven项目
- ✧ ② 在pom.xml中添加Spring Boot相关依赖
- ✧ ③ 编写项目启动类
- ✧ ④ 创建一个用于Web访问的Controller
- ✧ ⑤ 运行项目



①创建Maven项目



创建Maven项目

New Project

GroupId	com.itheima	域名和公司名	<input checked="" type="checkbox"/> Inherit
ArtifactId	chapter03manual	项目唯一标识符	
Version	1.0-SNAPSHOT	项目版本号（默认值）	<input checked="" type="checkbox"/> Inherit

Previous **Next** Cancel Help

New Project

Project name: chapter03manual

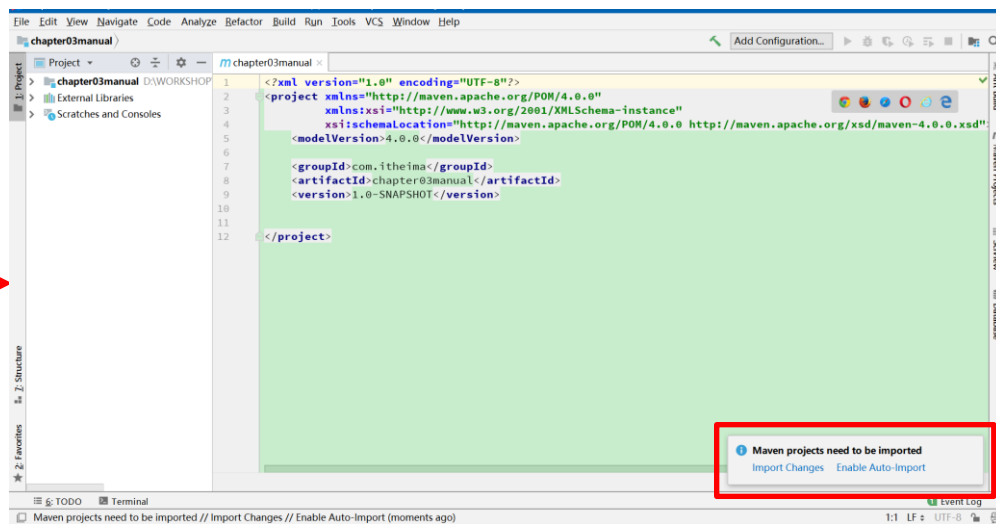
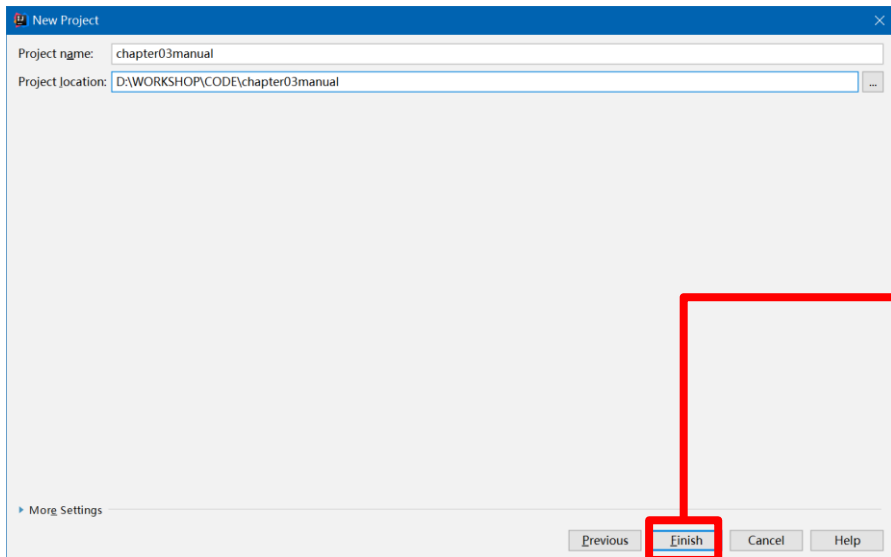
Project location: D:\WORKSHOP\CODE\chapter03manual

More Settings

Previous **Finish** Cancel Help



创建Maven项目



②添加Spring Boot相关依赖

<!-- 引入Spring Boot依赖的父包，为项目提供统一的子依赖版本管理-->

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.4.4</version>

</parent>

父依赖启动器

<dependencies>

<!-- 引入Web场景依赖启动器，所需依赖文件（.pom文件，如spring-webmvc、spring-web、spring-boot-starter-tomcat等）和相应JAR包便会自动获取和下载-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

Web依赖启动器

③编写项目启动类

```
package com.itheima;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Chapter03manualApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(Chapter03manualApplication.class, args);  
    }  
}
```

标记该类为项目启动类

SpringApplication.run()方法启动项目启动类

④创建Controller

```
package com.itheima.controller;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
```

该注解为组合注解，等同于Spring中
@Controller+ @ResponseBody注解

```
public class HelloController {
```

```
    @GetMapping("/hello")
```

```
    public String hello(){
```

```
        return "hello Spring Boot";
```

```
    }
```

```
}
```

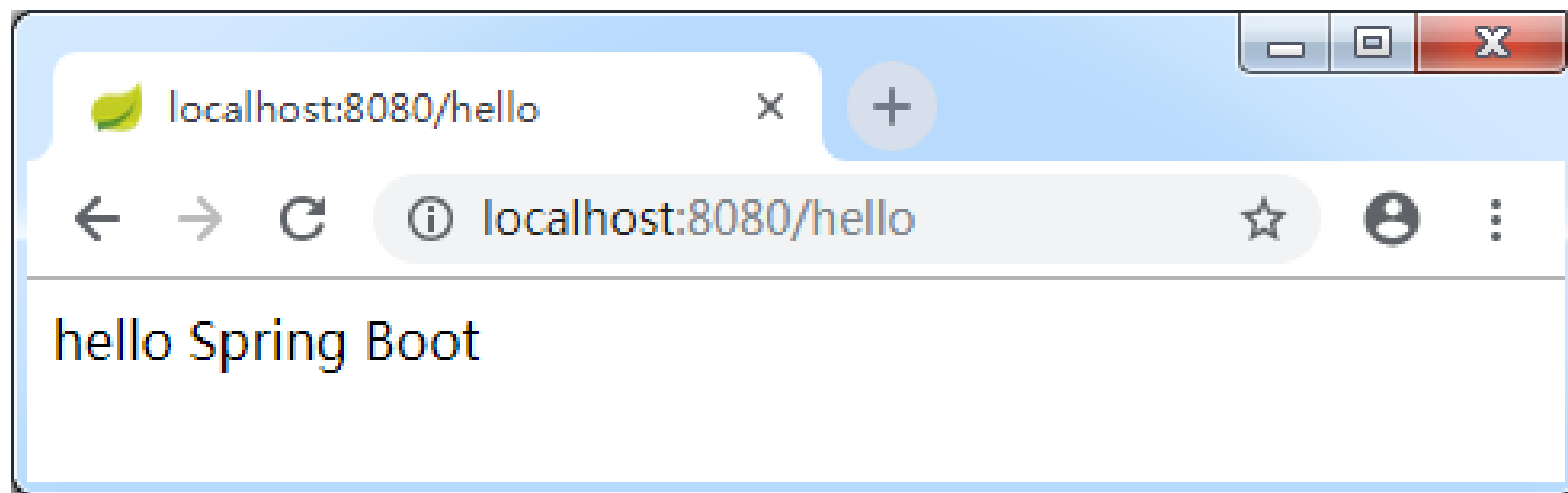
等同于Spring框架中

@RequestMapping(RequestMethod.GET)注解



⑤运行项目

- 启动项目，在浏览器上访问
`http://localhost:8080/hello`



3.3 单元测试和热部署

■ 3.3.1 单元测试

■ 3.3.2 热部署



3.3.1 单元测试

■ 搭建步骤:

- ✧①在pom.xml中添加spring-boot-starter-test测试依赖启动器
- ✧②编写项目测试类
- ✧③编写单元测试方法
- ✧④运行结果



①添加测试依赖启动器

```
<!-- 引入测试依赖启动器 -->
```

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-test</artifactId>
```

```
    <scope>test</scope>
```

```
</dependency>
```

Spring Initializr方式构建的项目会自动引入该测试依赖启动器。



②编写项目测试类

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
@SpringBootTest
public class chapter03ApplicationTests {
    @Test
    public void contextLoads() {
    }
}
```

标记该类为项目测试类

@SpringBootTest：标记项目测试类，并加载项目的Spring容器ApplicationContext
Spring Initializr方式构建的项目会自动创建该测试类。



③编写单元测试方法

@Autowired

自动注入HelloController实例对象

```
private HelloController helloController;
```

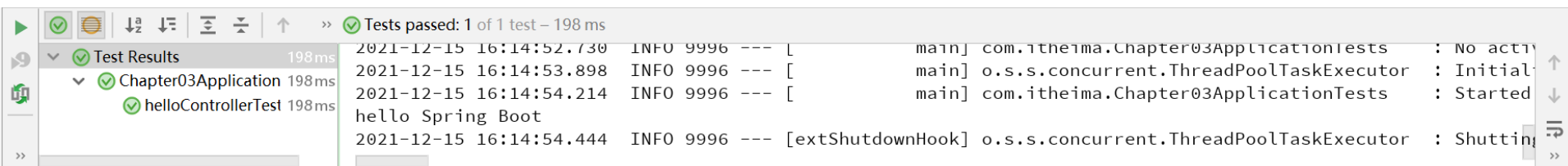
@Test

```
public void helloControllerTest() {  
    String hello = helloController.hello();  
    System.out.println(hello);  
}
```

@Autowired: 将Spring容器创建的一个Bean（HelloController实例对象）自动装配(自动注入)到另一个Bean（项目测试类实例）的Property中

④运行结果

■ 执行测试方法**helloControllerTest()**，控制台输出如图。



```
Tests passed: 1 of 1 test - 198 ms
Test Results 198ms
  Chapter03Application 198ms
    helloControllerTest 198ms
2021-12-15 16:14:52.730 INFO 9996 --- [main] com.itheima.Chapter03ApplicationTests : No active profiles found
2021-12-15 16:14:53.898 INFO 9996 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
2021-12-15 16:14:54.214 INFO 9996 --- [main] com.itheima.Chapter03ApplicationTests : Started ApplicationTests in 1.544 seconds
hello Spring Boot
2021-12-15 16:14:54.444 INFO 9996 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'main'
```



3.3.2 热部署

■ 搭建步骤:

- ✧ ①在pom.xml中添加spring-boot-devtools热部署依赖
- ✧ ②IDEA中热部署设置
- ✧ ③热部署测试



①添加热部署依赖

```
<!-- 引入热部署依赖 -->
```

```
<dependency>
```

```
  <groupId>org.springframework.boot</groupId>
```

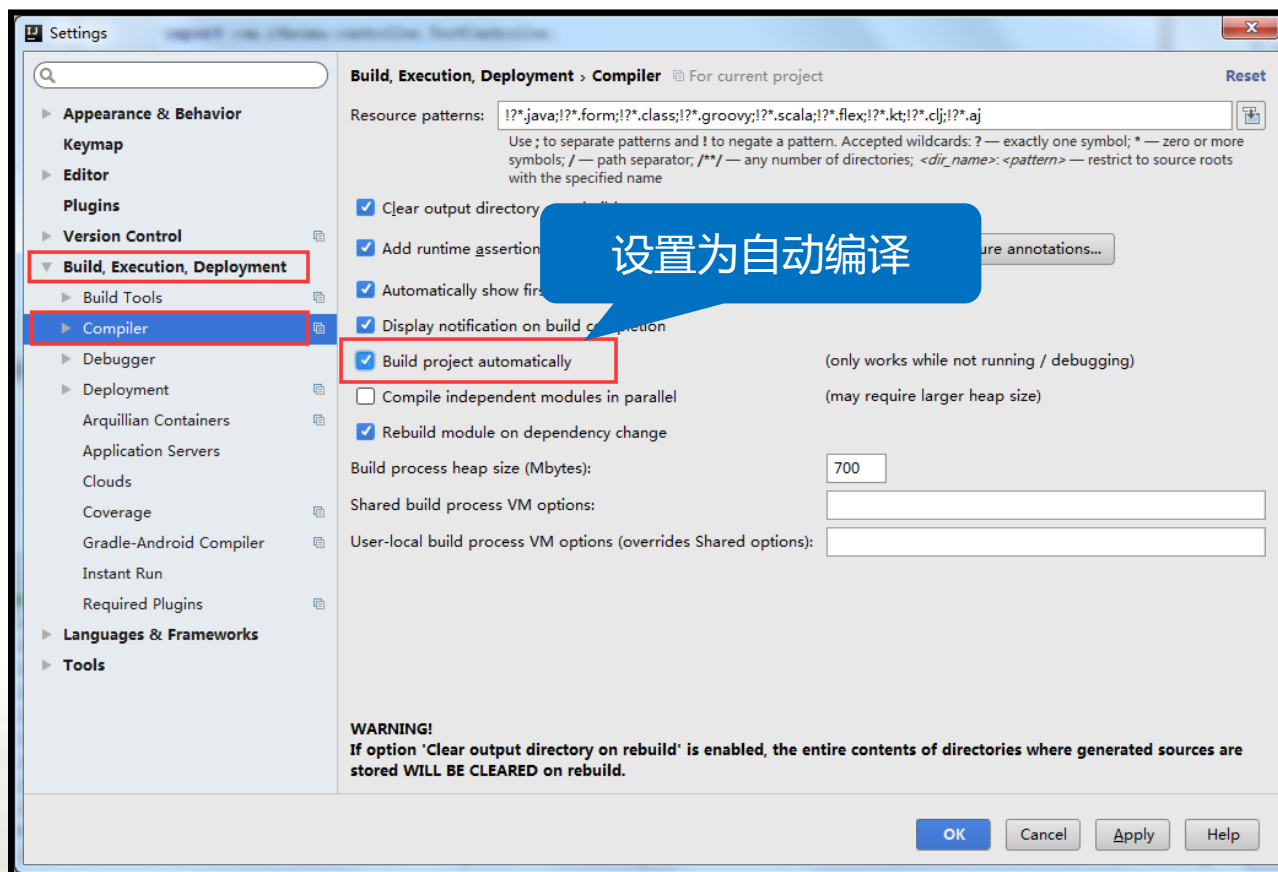
```
  <artifactId>spring-boot-devtools</artifactId>
```

```
</dependency>
```



②IDEA中热部署设置

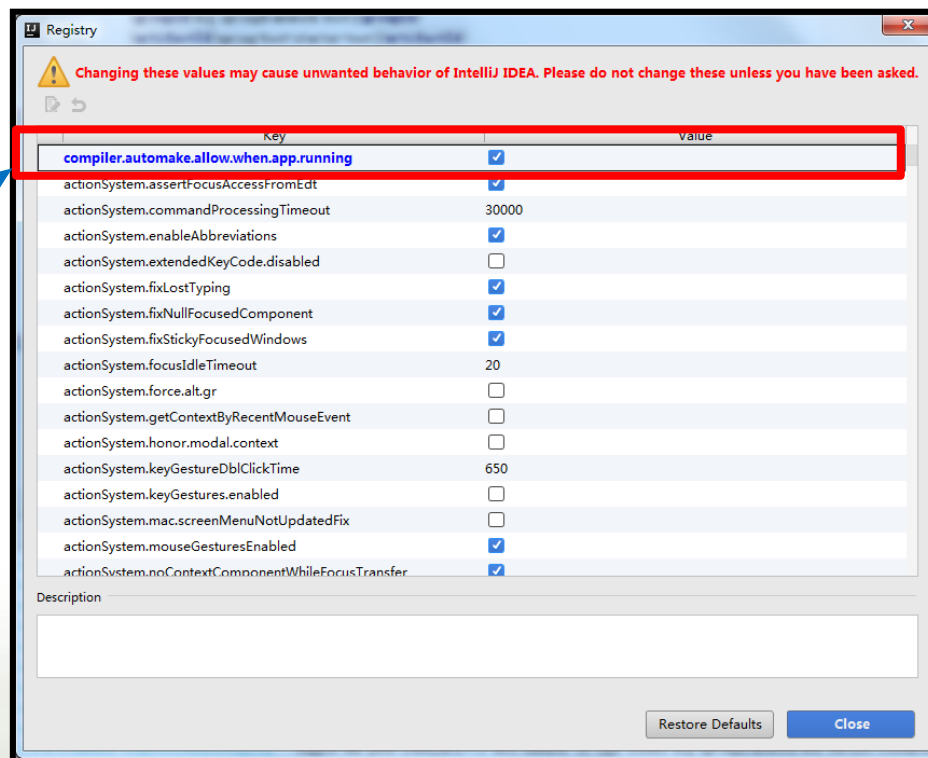
■ 选择【File】→【Settings】选项，打开Compiler面板设置页。



IDEA中热部署设置

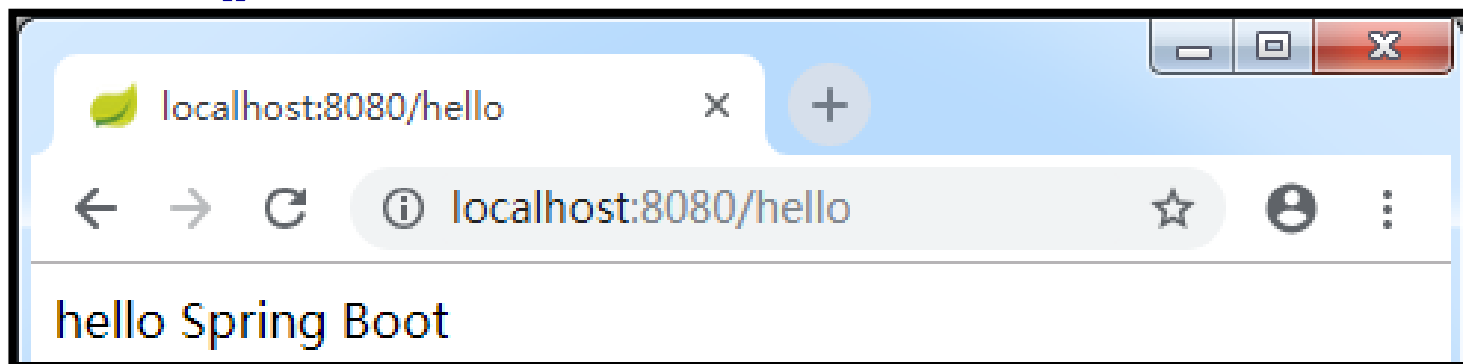
- 使用快捷键“**Ctrl+Shift+Alt+/,**”打开**Maintenance**选项框，选中并打开**Registry**页面。

指定IDEA工具在程序运行过程中自动编译



③热部署测试

- 启动chapter03项目，通过浏览器访问 <http://localhost:8080/hello>
- 修改类HelloController中的请求处理方法hello()的返回值，刷新浏览器。



3.4 Spring Boot应用的打包和部署

■ 3.4.1 Jar包方式打包部署

■ 3.4.2 War包方式打包部署



3.4.1 Jar包方式打包部署

- 1.Jar包方式打包
- 2.Jar包方式部署



1.Jar包方式打包

■ Jar包方式打包步骤:

- ✧①添加**Maven**打包插件
- ✧②使用**IDEA**进行打包
- ✧③观察**Jar**包目录结构



①添加Maven打包插件

■在pom.xml文件中添加Maven打包插件

```
<build>
```

```
  <plugins>
```

```
    <plugin>
```

```
      <groupId>org.springframework.boot</groupId>
```

```
      <artifactId>spring-boot-maven-plugin</artifactId>
```

```
    </plugin>
```

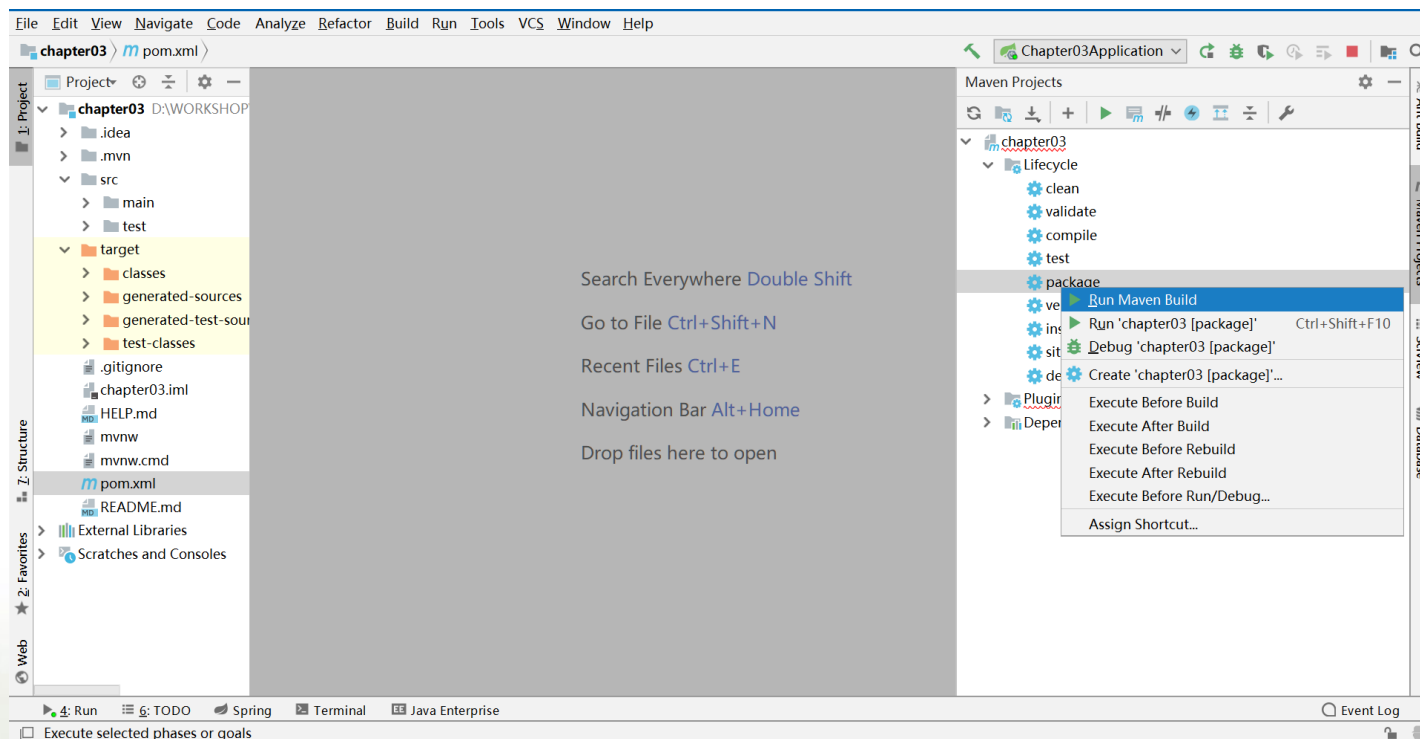
```
  </plugins>
```

```
</build>
```

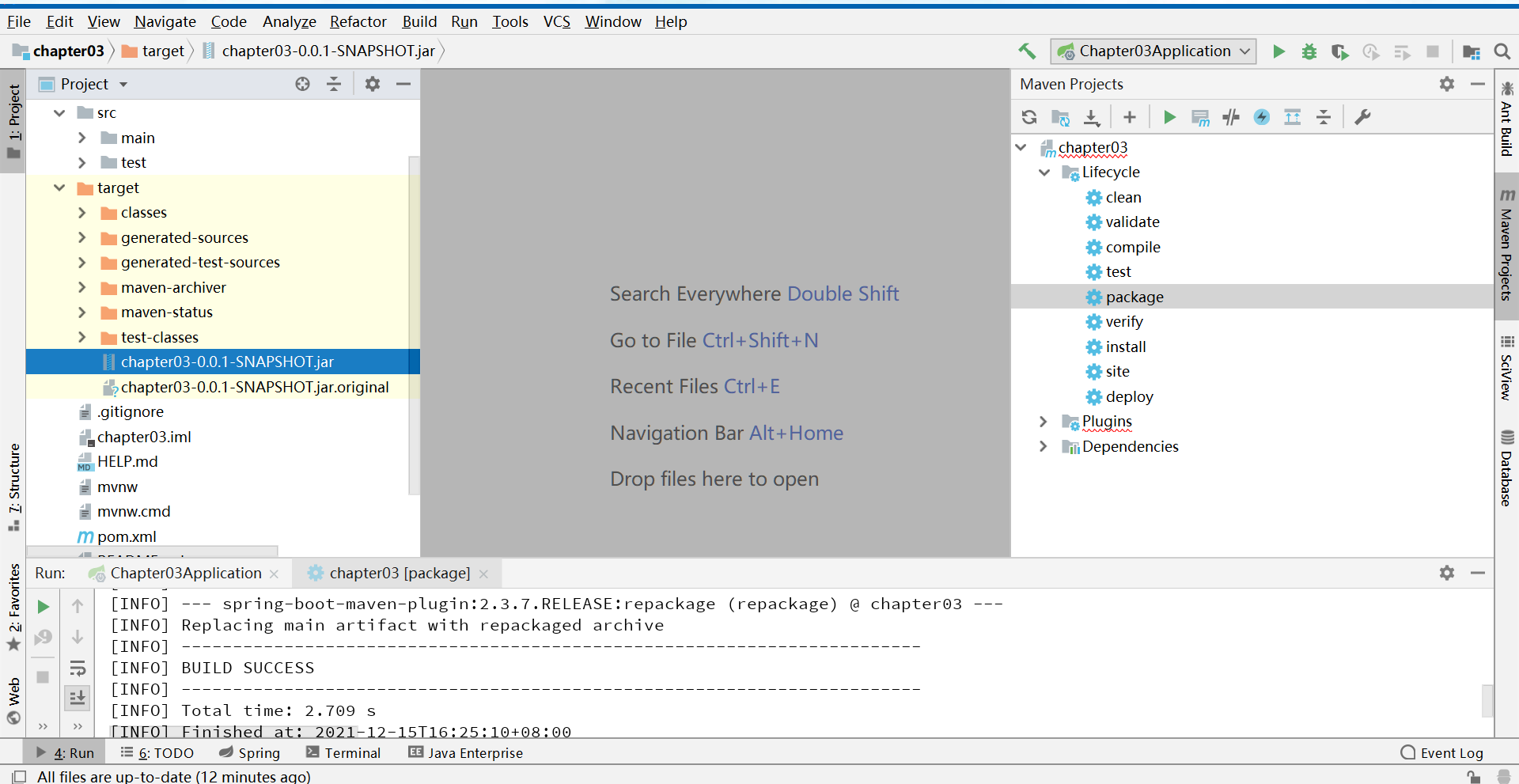


②使用IDEA进行打包

■按如图操作，**jar**包会生成在项目的**target**目录下。



生成的Jar包



其中，*.jar.original是不包含项目依赖的原始Jar包。



③观察Jar包目录结构

■ 右键Jar包名称使用压缩软件打开并进入到BOOT-INF目录。



2.Jar包方式部署

- 按如图在**IDEA**控制台上切换到**Terminal**界面，输入部署命令执行，成功后即可访问项目。



The screenshot shows the IntelliJ IDEA interface with the Terminal window active. The terminal displays the following text:

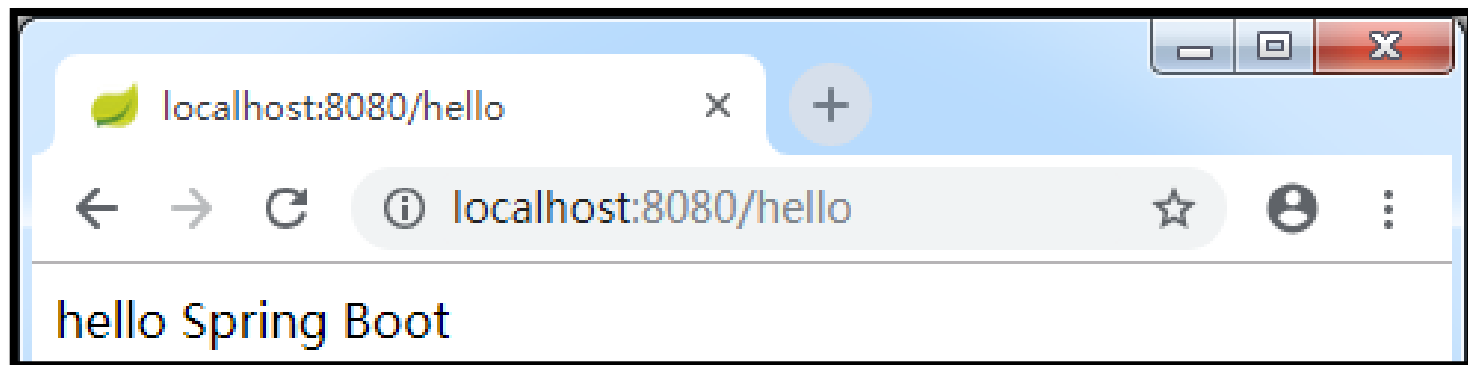
```
Terminal
+ Microsoft Windows [版本 10.0.19042.1348]
x (c) Microsoft Corporation. 保留所有权利。

D:\WORKSHOP\CODE\chapter03>java -jar target\chapter03-0.0.1-SNAPSHOT.jar
```

The bottom status bar of the IDE shows tabs for '4: Run', '6: TODO', 'Spring', 'Terminal', and 'Java Enterprise'. The 'Terminal' tab is currently selected.



访问效果



3.4.2 War包方式打包部署

- 1.War包方式打包
- 2.War包方式部署



1.War包方式打包

■ War包方式打包步骤:

- ✧①添加**Maven**打包插件
- ✧②声明打包方式为**War**包
- ✧③声明使用外部**Web**服务器
- ✧④提供**Servlet**初始化器
- ✧⑤使用**IDEA**进行打包



①添加Maven打包插件

■在pom.xml文件中添加Maven打包插件

```
<build>  
  <plugins>  
    <plugin>  
      <groupId>org.springframework.boot</groupId>  
      <artifactId>spring-boot-maven-plugin</artifactId>  
    </plugin>  
  </plugins>  
</build>
```



②声明打包方式为War包

■ 在pom.xml文件中声明打包方式为War包。

```
<groupId>com.itheima</groupId>  
<artifactId>chapter03</artifactId>  
<version>0.0.1-SNAPSHOT</version>  
<name>chapter03</name>  
<description>Demo project for Spring Boot</description>  
<packaging>war</packaging>  
<properties>  
    <java.version>1.8</java.version>  
</properties>
```



③声明使用外部Web服务器

■在pom.xml文件中声明使用外部Tomcat服务器

```
<!-- 声明使用外部提供的Tomcat -->
```

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-tomcat</artifactId>
```

```
    <scope>provided</scope>
```

```
</dependency>
```



pom.xml完整内容

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.4.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.itheima</groupId>
    <artifactId>chapter03</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>chapter03</name>
    <description>Demo project for Spring Boot</description>
```


pom.xml完整内容

```
<packaging>war</packaging>
<properties>
  <java.version>1.8</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
```



pom.xml完整内容

```
<!-- 声明使用外部提供的Tomcat -->
```

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-tomcat</artifactId>
```

```
    <scope>provided</scope>
```

```
</dependency>
```

```
<!-- 引入热部署依赖 -->
```

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-devtools</artifactId>
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-test</artifactId>
```

```
    <scope>test</scope>
```

```
</dependency>
```

```
</dependencies>
```

pom.xml完整内容

```
<build>  
  <plugins>  
    <plugin>  
      <groupId>org.springframework.boot</groupId>  
      <artifactId>spring-boot-maven-plugin</artifactId>  
    </plugin>  
  </plugins>  
</build>  
</project>
```



④提供Servlet初始化器

- 为Spring Boot提供启动的Servlet初始化器
SpringBootServletInitializer的典型做法：
 - ✧ 让项目启动类继承**SpringBootServletInitializer**类并实现**configure()**方法；
- 除此之外，还可以在项目中单独提供一个继承**SpringBootServletInitializer**的子类，并实现**configure()**方法。



示例代码

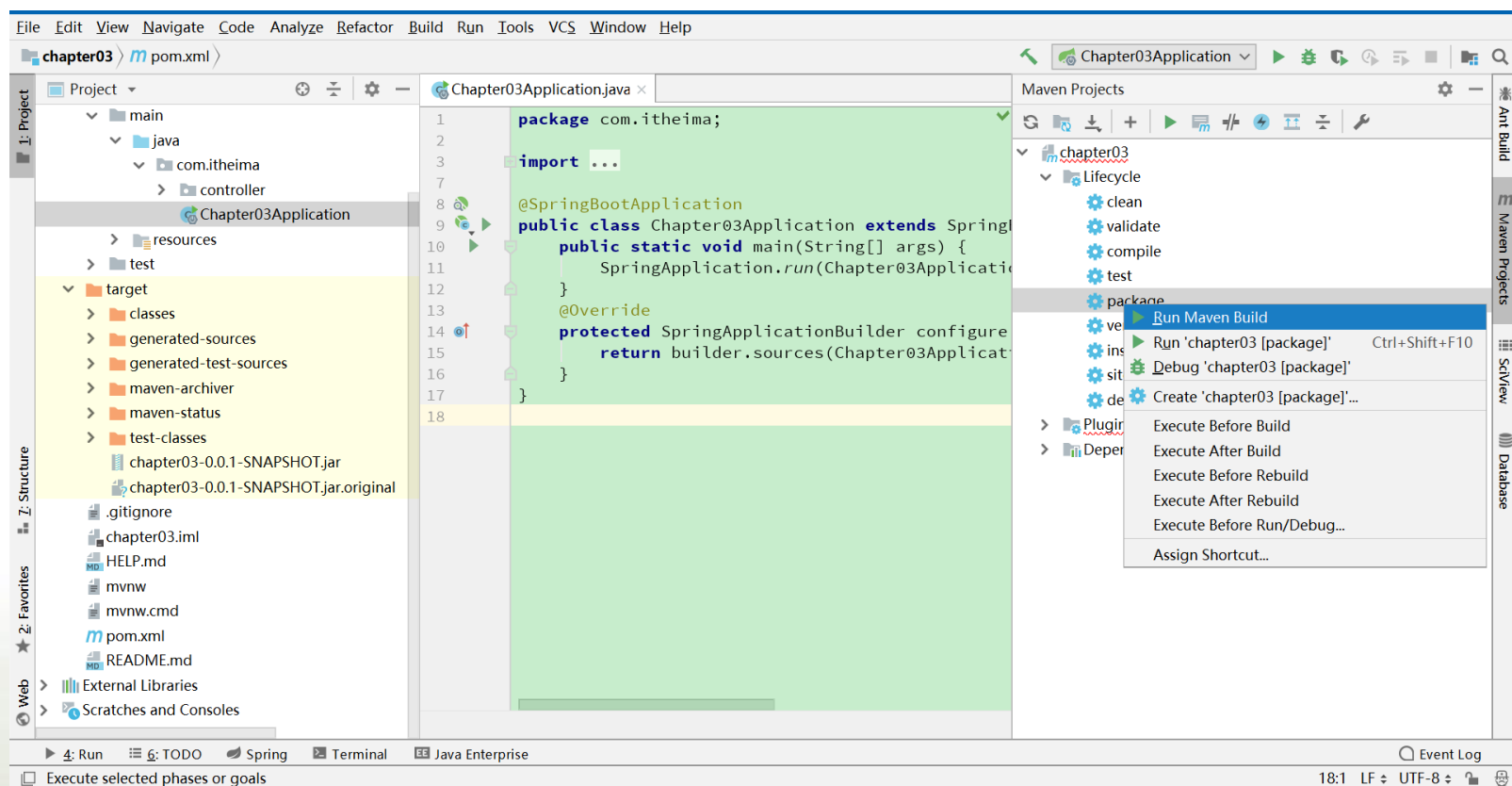
```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.builder.SpringApplicationBuilder;  
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
```

```
@SpringBootApplication
```

```
public class Chapter03Application extends SpringBootServletInitializer {  
    public static void main(String[] args) {  
        SpringApplication.run(Chapter03Application.class, args);  
    }  
    @Override  
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {  
        return builder.sources(Chapter03Application.class);  
    }  
}
```

⑤使用IDEA进行打包

■按如图操作，war包会生成在项目的target目录下。

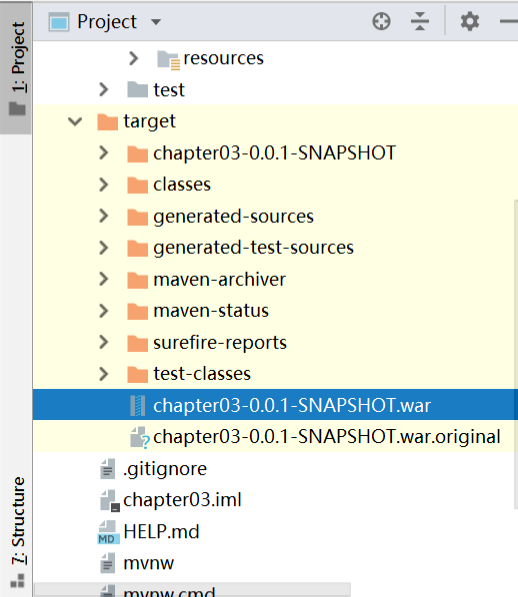


生成的war包

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

chapter03 > target > chapter03-0.0.1-SNAPSHOT.war

Chapter03Application



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>2.4.4</version>
9         <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>com.itheima</groupId>
12    <artifactId>chapter03</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <name>chapter03</name>
15    <description>Demo project for Spring Boot</description>
16    <packaging>war</packaging>
17    <properties>
18        <java.version>1.8</java.version>
19    </properties>
</project>
```

Run: chapter03 [package] x

```
[INFO]
[INFO] --- spring-boot-maven-plugin:2.4.4:repackage (repackage) @ chapter03 ---
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 13.797 s
[INFO] Finished at: 2021-12-16T07:31:04+08:00
```

4: Run 6: TODO Spring Terminal Java Enterprise

Event Log

IDE and Plugin Updates: IntelliJ IDEA is ready to update. (yesterday 18:09)

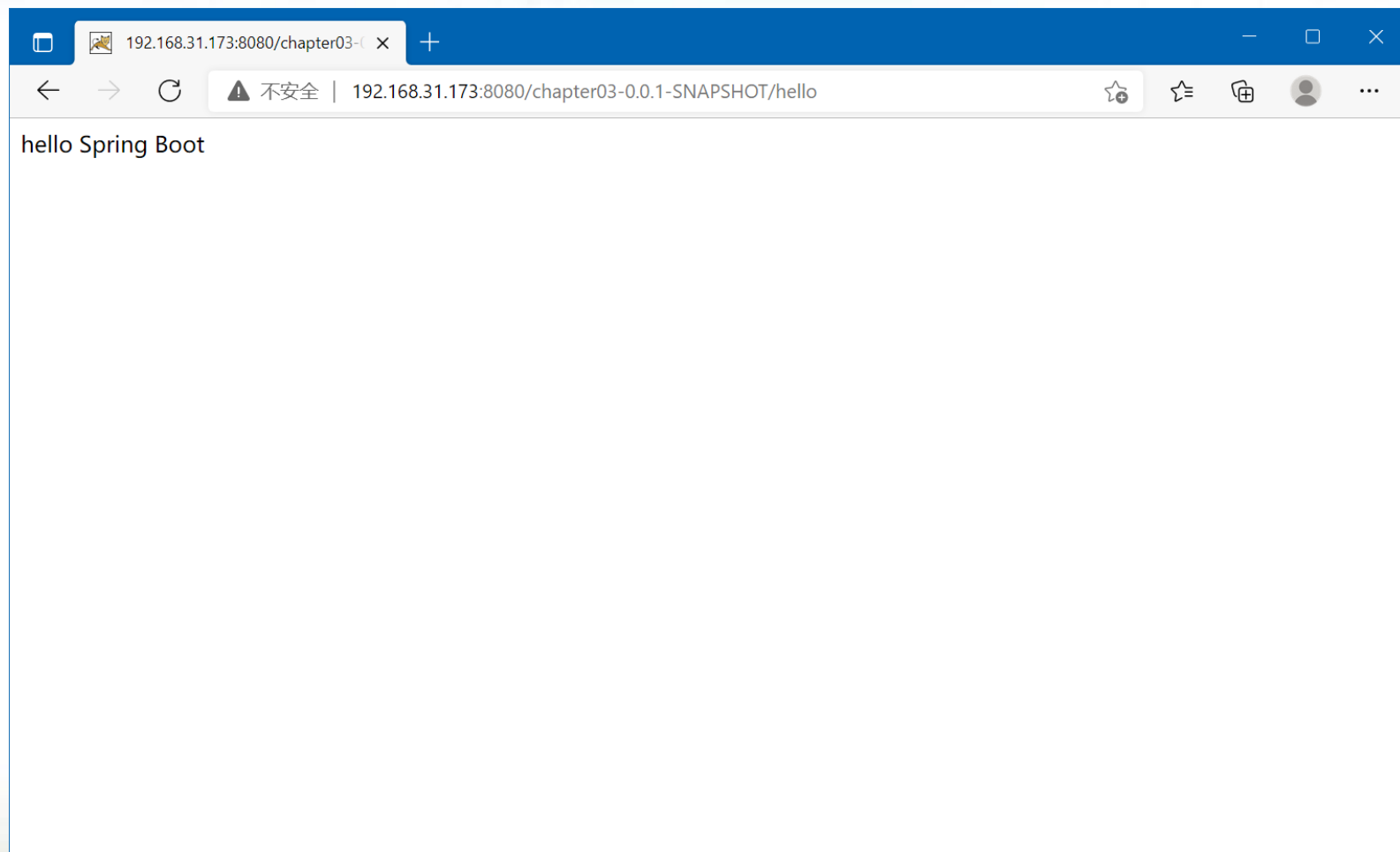
15:7 LF UTF-8

2.War包方式部署

- 将打包好的War包拷贝到Tomcat安装目录下的webapps目录中，执行Tomcat安装目录下bin目录中的startup.bat命令启动War包项目。
- 使用外部Tomcat部署的项目进行访问时，必须加上项目名称（打成war包后的项目全名chapter03-0.0.1-SNAPSHOT）。



访问效果



本章小结

■本章具体讲解了：

- ✧3.1 Spring Boot概述
- ✧3.2 Spring Boot入门程序
- ✧3.3 单元测试和热部署
- ✧3.4 Spring Boot应用的打包和部署



