第3章

# Spring Boot 入门

## 学习目的与要求

本章首先介绍什么是 Spring Boot，然后介绍 Spring Boot 应用的开发环境，最后介绍如何快速构建一个 Spring Boot 应用。通过本章的学习，掌握如何构建 Spring Boot 应用的开发环境以及 Spring Boot 应用。

## 本章主要内容

- Spring Boot 概述。
- Spring Boot 应用的开发环境。
- Maven 构建 Spring Boot 应用。
- 快速构建 Spring Boot 应用。

从前两章的学习可知，Spring 框架非常优秀，但问题在于“配置过多”，造成开发效率低、部署流程复杂以及集成难度大等问题。为解决上述问题，Spring Boot 应运而生。作者在编写本书时，Spring Boot 的最新正式版是 2.1.4.RELEASE，Spring Boot 2.2.0M1 里程碑版本已经发布。为了提高程序的稳定性，本书将以 Spring Boot 2.1.4.RELEASE 版本编写示例代码。读者测试本书示例代码时，建议使用 2.1.4.RELEASE 或更高版本。

## 3.1 Spring Boot 概述


视频讲解

### 3.1.1 什么是 Spring Boot

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用

的初始搭建以及开发过程。使用 Spring Boot 框架可以做到专注于 Spring 应用的开发，无须过多关注样板化的配置。

在 Spring Boot 框架中，使用“约定优于配置（Convention Over Configuration，COC）”的理念。针对企业应用开发，提供了符合各种场景的 spring-boot-starter 自动配置依赖模块，这些模块都是基于“开箱即用”的原则，进而使企业应用开发更加快捷和高效。可以说，Spring Boot 是开发者和 Spring 框架的中间层，目的是帮助开发者管理应用的配置，提供应用开发中常见配置的默认处理（即约定优于配置），简化 Spring 应用的开发和运维，降低开发人员对框架的关注度，使开发人员把更多精力放在业务逻辑代码上。通过“约定优于配置”的原则，Spring Boot 致力于在蓬勃发展的快速应用开发领域成为领导者。

## 3.1.2 Spring Boot 的优点

Spring Boot 之所以能够应运而生，是因为它具有以下优点：

（1）使编码变得简单：推荐使用注解。

（2）使配置变得快捷：具有自动配置、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 等 Web 容器。

（4）使监控变得容易：自带项目监控。

## 3.1.3 Spring Boot 的主要特性

**1. 约定优于配置**

Spring Boot 遵循“约定优于配置”的原则，只需很少的配置，大多数情况直接使用默认配置即可。

**2. 独立运行的 Spring 应用**

Spring Boot 可以以 jar 包的形式独立运行。使用 java -jar 命令或者在项目的主程序中执行 main 方法运行 Spring Boot 应用（项目）。

**3. 内嵌 Web 容器**

内嵌 Servlet 容器，Spring Boot 可以选择内嵌 Tomcat、Jetty 等 Web 容器，无须以 war 包形式部署应用。

**4. 提供 starter 简化 Maven 配置**

Spring Boot 提供了一系列的 starter pom 简化 Maven 的依赖加载，基本上可以做到自动化配置，高度封装，开箱即用。

**5. 自动配置 Spring**

Spring Boot 根据项目依赖(在类路径中的 jar 包、类)自动配置 Spring 框架,极大地减少了项目的配置。

**6. 提供准生产的应用监控**

Spring Boot 提供基于 HTTP、SSH、TELNET 对运行的项目进行跟踪监控。

**7. 无代码生成和 XML 配置**

Spring Boot 不是借助于代码生成来实现的,而是通过条件注解来实现的。提倡使用 Java 配置和注解配置相结合的配置方式,方便快捷。

# 3.2 第一个 Spring Boot 应用


视频讲解

## 3.2.1 Maven 简介

Apache Maven 是一个软件项目管理工具。基于项目对象模型(Project Object Model, POM)的理念,通过一段核心描述信息来管理项目构建、报告和文档信息。在 Java 项目中,Maven 主要完成两件工作:①统一开发规范与工具;②统一管理 jar 包。

Maven 统一管理项目开发所需要的 jar 包,但这些 jar 包将不再包含在项目内(即不在 lib 目录下),而是存放于仓库中。仓库主要包括以下内容。

**1. 中央仓库**

中央仓库存放开发过程中的所有 jar 包,例如 JUnit,这些 jar 包都可以通过互联网从中央仓库下载,仓库地址:http://mvnrepository.com。

**2. 本地仓库**

本地仓库即本地计算机中的仓库。官方下载 Maven 的本地仓库,配置在"%MAVEN_HOME%\conf\settings.xml"文件中,找到 localRepository 即可;Eclipse 中自带 Maven 的默认本地仓库地址在"{user.home}/.m2/repository/settings.xml"文件中,同样找到 localRepository 即可。

Maven 项目首先会从本地仓库中获取所需要的 jar 包,当无法获取指定 jar 包时,本地仓库会从远程仓库(中央仓库)下载 jar 包,并放入本地仓库以备将来使用。

## 3.2.2 Maven 的 pom.xml

Maven 是基于项目对象模型的理念管理项目的,所以 Maven 的项目都有一个 pom.xml 配置文件来管理项目的依赖以及项目的编译等功能。

在 Maven Web 项目中，重点关注以下元素。

**1. properties 元素**

在< properties ></properties >之间可以定义变量，以便在< dependency ></dependency > 中引用，代码如下：

```
<properties>
    <!-- spring 版本号 -->
    <spring.version>5.1.5.RELEASE</spring.version>
</properties>
<dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>
</dependencies>
```

**2. dependencies 元素**

< dependencies ></dependencies >，此元素包含多个项目依赖需要使用的< dependency > </dependency >元素。

**3. dependency 元素**

< dependency ></dependency >元素内部通过< groupId ></groupId >、< artifactId > </artifactId >、< version ></version > 3 个子元素确定唯一的依赖，也可以称为 3 个坐标。代码如下：

```
<dependency>
    <!-- groupId 组织的唯一标识  -->
    <groupId>org.springframework</groupId>
    <!-- artifactId 项目的唯一标识  -->
    <artifactId>spring-core</artifactId>
    <!-- version 项目的版本号 -->
    <version>${spring.version}</version>
</dependency>
```

## 3.2.3 在 Eclipse 中创建 Maven Web 项目

本节在基于 Eclipse 平台的 Java Web 应用开发环境(见 1.2 节)的基础上，创建 Maven Web 项目。

**1. 新建 Maven Web 项目**

在 Eclipse 中新建 Maven Web 项目，具体实现步骤如下：

(1) 选择菜单 File|New|Maven Project，打开如图 3.1 所示的 Select project name and

location 对话框。

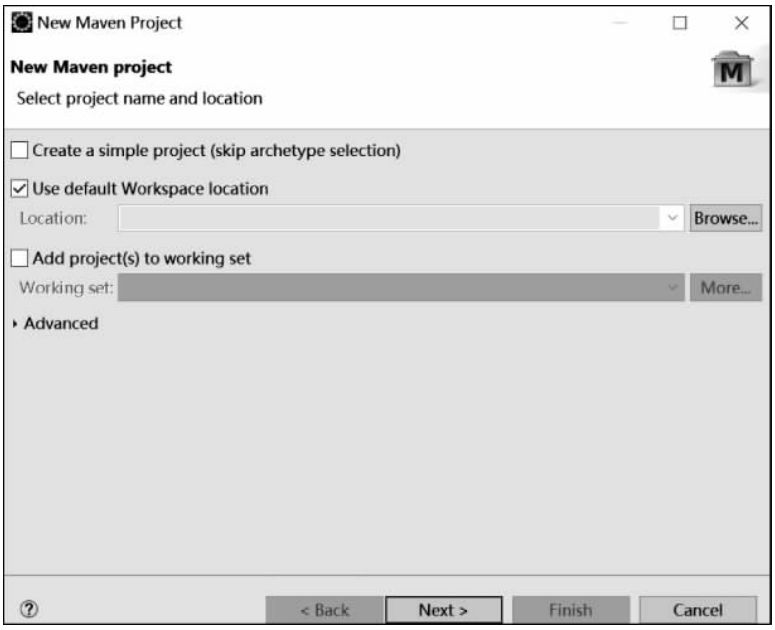



图 3.1 Select project name and location

(2) 单击图 3.1 中的 Next 按钮,打开如图 3.2 所示的 Select an Archetype 对话框,在该对话框中,选择 Archetype 为 webapp。

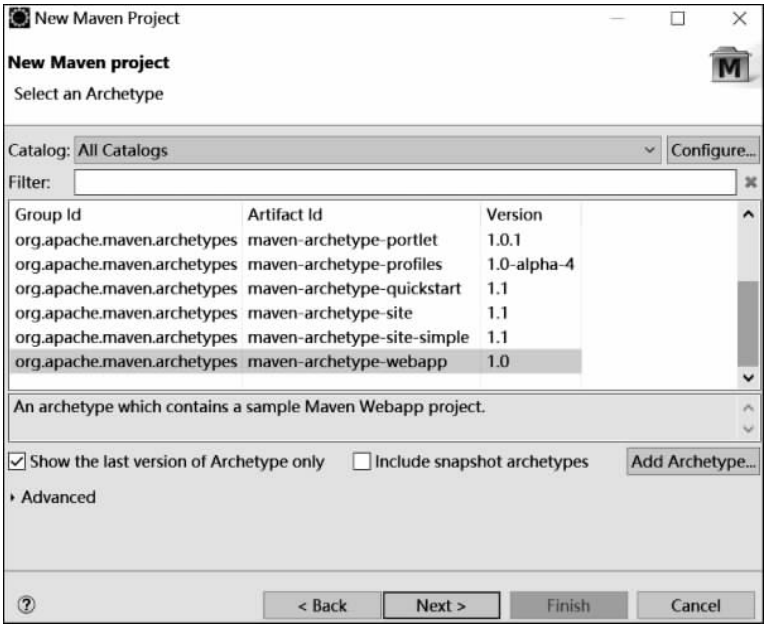



图 3.2 Select an Archetype

(3) 单击图 3.2 中的 Next 按钮,打开如图 3.3 所示的 Specify Archetype parameters 对话框,在该对话框中,输入一些必要信息,单击 Finish 按钮。

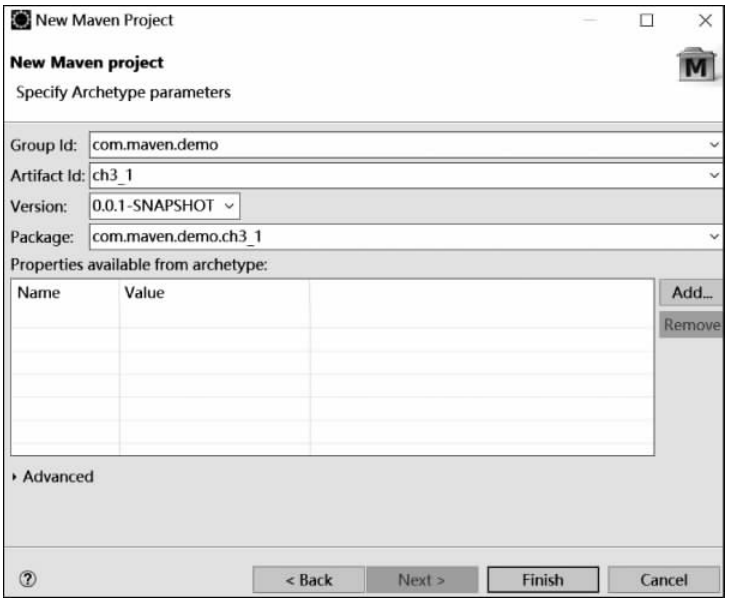



图 3.3 Enter a group id for the artifact

(4) 创建的 Maven Web 项目 ch3_1 的目录结构如图 3.4 所示。

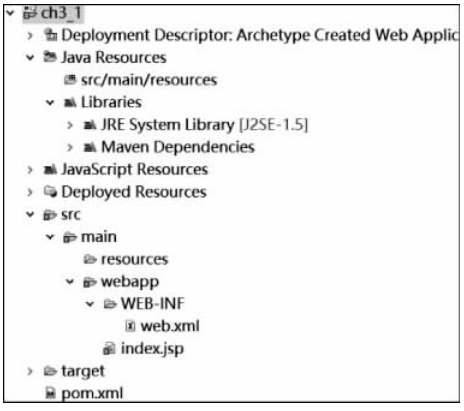



图 3.4 项目 ch3_1 的目录结构

### 2. 配置 Maven Web 项目

新建的 Maven Web 项目,需要修改一些配置,具体步骤如下:

(1) 右击项目 ch3_1,选择 Build Path | Configure Build Path,打开如图 3.5 所示的 Properties for ch3_1 对话框。

(2) 在图 3.5 中,选择 Libraries 标签,选中 JRE System Library,单击 Edit 按钮,打开如图 3.6 所示的 Select JRE for the project build path 对话框。

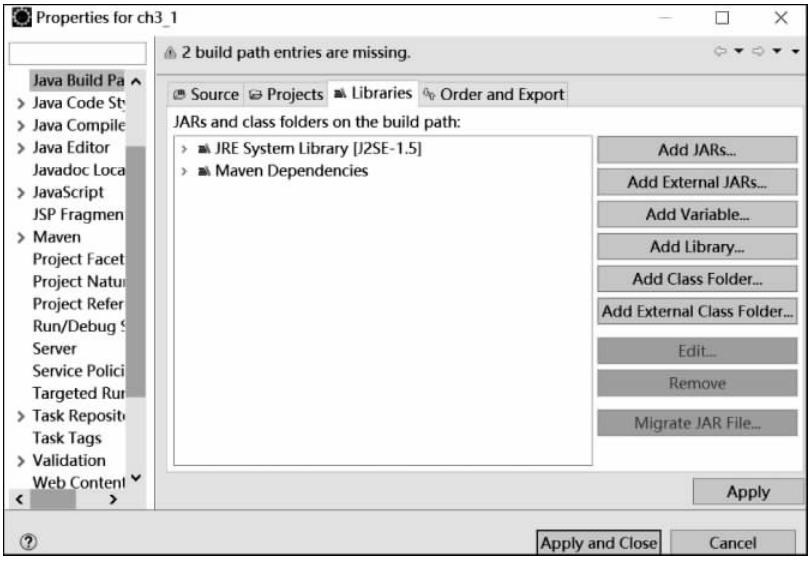



图 3.5 Properties for ch3_1

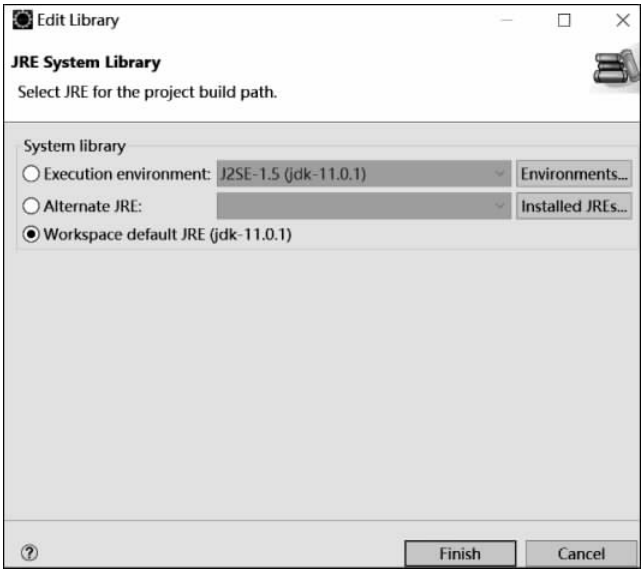



图 3.6 Select JRE for the project build path

(3) 在图 3.6 中,选择 Workspace 默认的 JRE,单击 Finish 按钮,最后,单击 Apply and Close 即可。完成后,项目结构如图 3.7 所示。

图 3.7 中 src/main/java 目录包含项目的 Java 源代码; src/main/resources 目录包含项目所需的资源(如配置文件); src/test/java 目录包含用于测试的 Java 代码; src/main/webapp 目录包含 Java Web 应用程序; 目录由 Maven 创建。target 包含所有编译的类、JAR 文件等。当执行 mvn clean 命令时,Maven 将清除此目录。

(4) 右击项目名 ch3_1,选择 Run As|Run on Server,运行 ch3_1 项目,运行结果如图 3.8 所示。

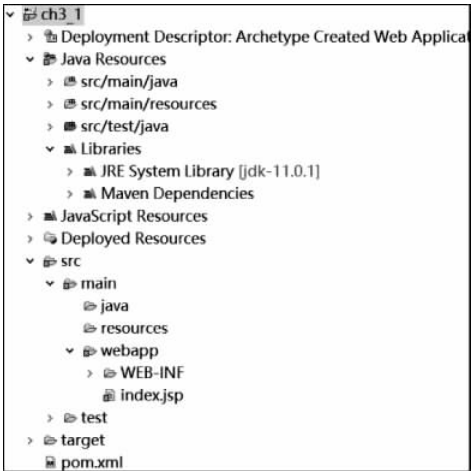



图 3.7　项目结构图


图 3.8　运行 ch3_1 项目

## 3.2.4　Maven 手工构建第一个 Spring Boot 应用

本节在 3.2.3 节 Maven Web 项目 ch3_1 的基础上，手工构建一个 Spring Boot 应用，具体实现步骤如下。

**1. 配置 Spring Boot 的核心启动器**

构建基于 Spring Boot 的应用。首先，在 pom.xml 文件的< url…/>元素之后添加< parent…/>元素配置 Spring Boot 的核心启动器 spring-boot-starter-parent，代码如下：

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.4.RELEASE</version>
</parent>
```

**2. 添加 starter 模块**

然后，在 pom.xml 文件的< dependencies…/>元素中增加一个< dependency…/>元素添加需要的 starter 模块，此处只添加了 spring-boot-starter-web 模块。Spring Boot 包含了很多 starter 模块(见 4.1.6 节)，每一个 starter 模块就是一系列的依赖组合。如 spring-boot-starter-web 模块包含 Spring Boot 预定义的 Web 开发的常用依赖包(包括 Tomcat 和 spring-webmvc)。由于指定了 spring-boot-starter-parent 的版本，所以此处的 spring-boot-starter-web 模块不需要指定版本，Spring Boot 将自动选择匹配的版本进行加载。代码如下：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Web 项目 ch3_1 的 pom.xml 文件修改后的内容如下：

```
<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 http://maven.apache.org/maven -
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.maven.demo</groupId>
  <artifactId>ch3_1</artifactId>
  <packaging>war</packaging>
  <version>0.0.1 - SNAPSHOT</version>
  <name>ch3_1 Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring - boot - starter - parent</artifactId>
        <version>2.1.4.RELEASE</version>
  </parent>
  <dependencies>
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring - boot - starter - web</artifactId>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>ch3_1</finalName>
  </build>
</project>
```

pom.xml 文件修改保存后，Maven 将自动在互联网环境下，下载所需的所有 jar 文件。

### 3. 编写测试代码

在 src/main/java 目录下，创建 com.test 包，并在该包中创建 TestController 类，具体代码如下：

```
package com.test;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class TestController {
    @RequestMapping("/hello")
    public String hello() {
        return "您好,Spring Boot!";
    }
}
```

上述代码中使用的@RestController 注解是一个组合注解，相当于 Spring MVC 中的@Controller 和@ResponseBody 注解的组合，具体应用如下：

（1）如果只是使用@RestController 注解 Controller，则 Controller 中的方法无法返回 JSP 或者 html 页面，返回的内容就是 return 的内容。

（2）如果需要返回指定页面，则需要用@Controller 注解。如果需要返回 JSON，XML 或自定义 mediaType 内容到页面，则需要在对应的方法上加上@ResponseBody 注解。

**4. 创建应用程序的 App 类（启动类）**

在 com.test 包中创建 Ch3_1Application 类，具体代码如下：

```
package com.test;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class Ch3_1Application {
    public static void main(String[] args) {
        SpringApplication.run(Ch3_1Application.class, args);
    }
}
```

上述代码中使用@SpringBootApplication 注解指定该程序是一个 Spring Boot 应用，该注解也是一个组合注解，相当于@Configuration、@EnableAutoConfiguration 和@ComponentScan 注解的组合，具体细节在第 4 章讲解。SpringApplication 类调用 run 方法启动 Spring Boot 应用。

**5. 运行 main 方法启动 Spring Boot 应用**

运行 Ch3_1Application 类的 main 方法后，控制台信息如图 3.9 所示。

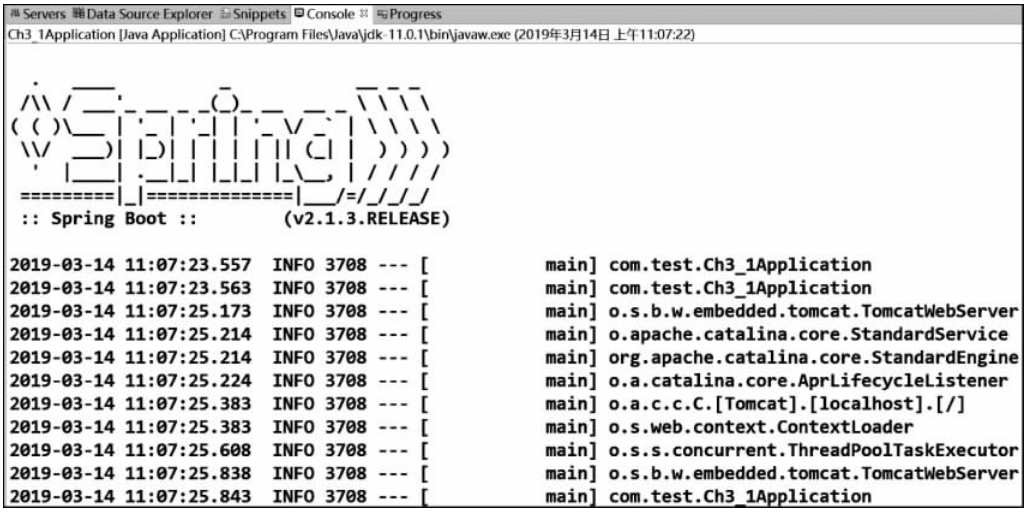



图 3.9　启动 Spring Boot 应用后的控制台信息

从上面的控制台信息可以看到 Tomcat 的启动过程、Spring MVC 的加载过程。注意 Spring Boot 内嵌 Tomcat 容器，因此 Spring Boot 应用不需要开发者配置与启动 Tomcat。

**6. 测试 Spring Boot 应用**

启动 Spring Boot 应用后，默认访问地址为 http://localhost:8080/，将项目路径直接设为根路径，这是 Spring Boot 的默认设置。因此，可以通过 http://localhost:8080/hello 测试应用（hello 与测试类 TestController 中的@RequestMapping("/hello")对应），测试效果如图 3.10 所示。


图 3.10 访问 Spring Boot 应用

视频讲解

# 3.3 Spring Boot 快速构建

在 3.2.4 节使用 Maven 方便快捷地手工构建了一个 Spring Boot 应用，但还可以使用更便捷的方法构建 Spring Boot 应用。本节将讲解两个方便快捷的构建方法：http://start.spring.io 和 Spring Tool Suite(STS)。

## 3.3.1 http://start.spring.io

使用 http://start.spring.io 快速构建 Spring Boot 应用的具体步骤如下。

**1. 打开 spring.io**

在浏览器地址栏中输入“http://start.spring.io”，打开如图 3.11 所示的界面。

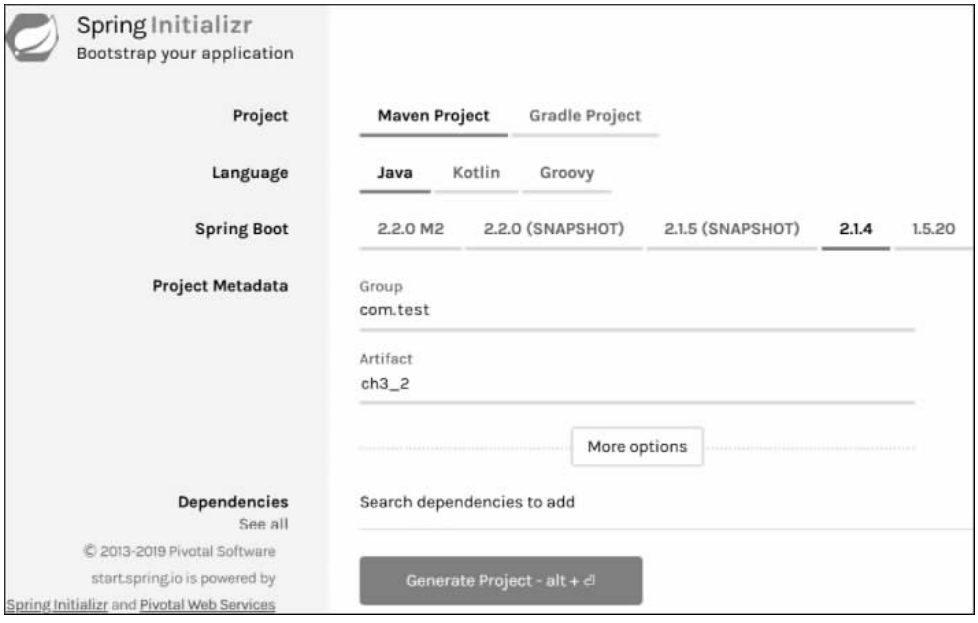



图 3.11 打开 spring.io

### 2. 填写项目信息

在图 3.11 中，Project 默认选择 Maven Project，Language 默认选择 Java，Spring Boot 默认选择最新正式版。Project Metadata 的 Group 输入“com.test”，Artifact 输入“ch3_2”，单击 More options 按钮，Java Version 选择 11。Project Metadata 的信息如图 3.12 所示。

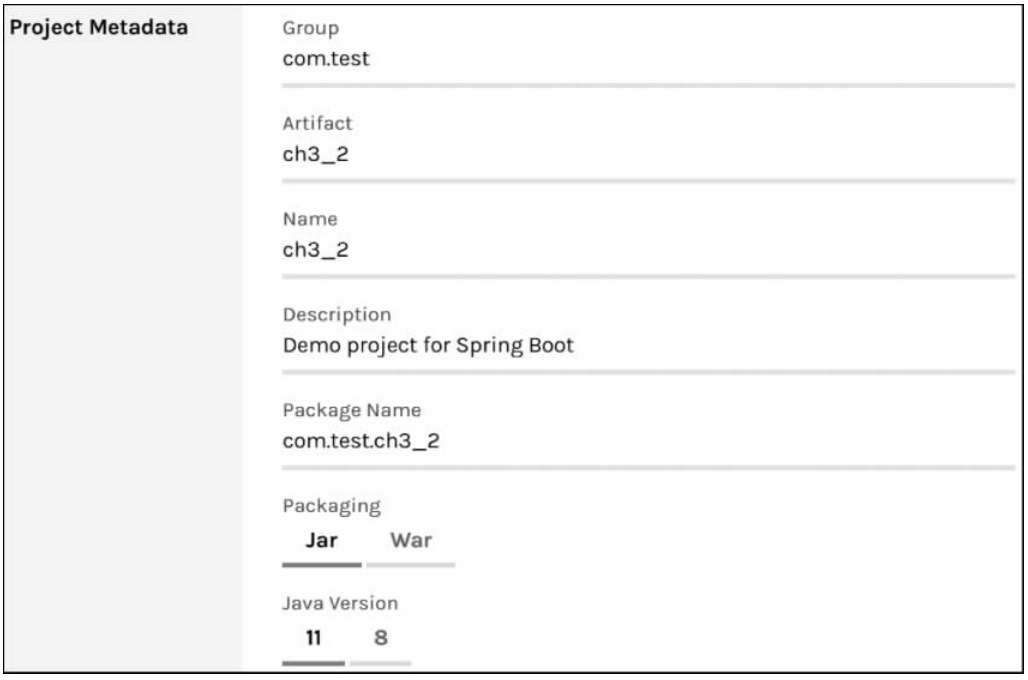



图 3.12　Project Metadata

在 Dependencies 处，输入“web”进行搜索，如图 3.13 所示。

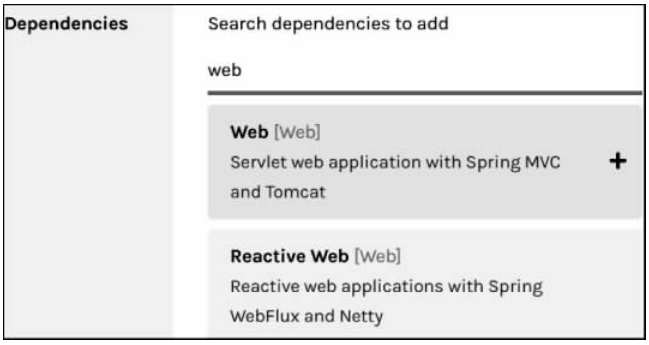



图 3.13　Search dependencies to add

单击图 3.13 中的“+”号添加 Web 依赖。

### 3. 创建应用并下载源代码

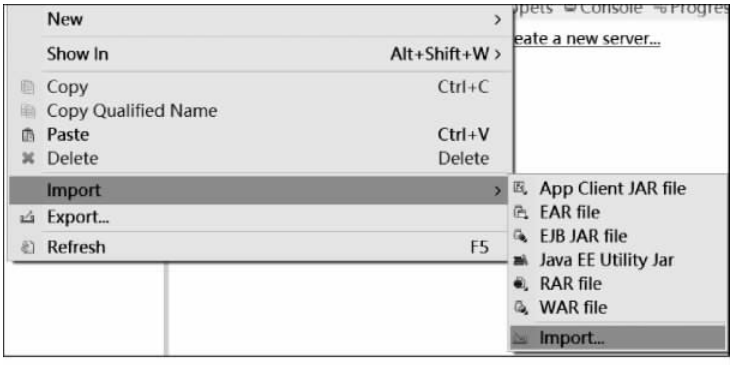
在图 3.11 中单击 Generate Project 按钮，创建 Spring Boot 应用，并下载源代码。此处

下载的源代码是一个简单的基于 Maven 的应用，解压后的目录如图 3.14 所示。

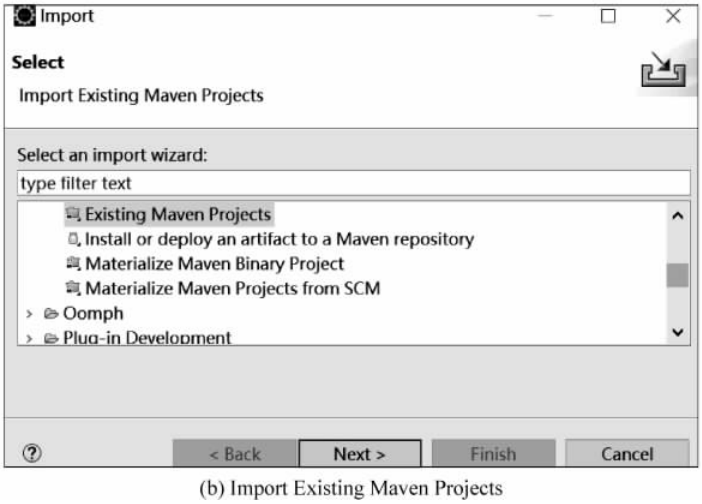

图 3.14 项目目录

### 4. 导入 Maven 项目到集成开发工具中

可将 ch3_2 这个 Maven 项目导入读者常用的集成开发工具中，如 Eclipse。Eclipse 导入 Maven 项目过程如图 3.15 所示。

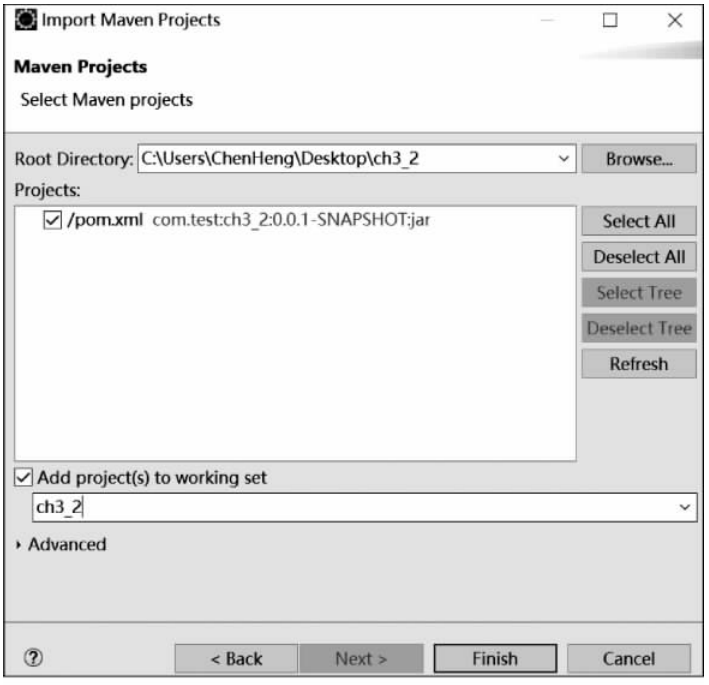



(a) 选择Eclipse的Import菜单项




(b) Import Existing Maven Projects

图 3.15 Eclipse 导入 Maven 项目过程




(c) Select Maven projects

图 3.15 （续）

成功导入 ch3_2 这个 Maven 项目后，在 Eclipse 中显示如图 3.16 所示的项目结构。

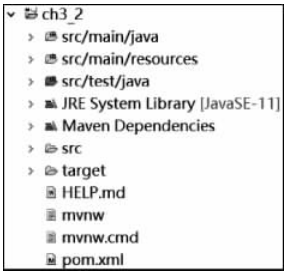



图 3.16 ch3_2 项目结构

这时，就可以在 ch3_2 项目中编写自己的 Spring Boot 应用程序了，如 Web 应用程序。

## 3.3.2 Spring Tool Suite

Spring Tool Suite（STS）是一个定制版的 Eclipse，专为 Spring 开发定制，方便创建、调试、运行、维护 Spring 应用。通过该工具，可以很轻易地生成一个 Spring 工程，例如 Web 工程，最令人兴奋的是工程中的配置文件都将自动生成，开发者再也不用关注配置文件的格式及各种配置了。可通过官网 https://spring.io/tools 下载 Spring Tools for Eclipse，本书采用的版本是 spring-tool-suite-4-4.1.1.RELEASE-e4.10.0-win32.win32.x86_64.zip。该版本与 Eclipse 一样，免安装，解压即可使用。

下面详细讲解如何使用 STS 集成开发工具快速构建一个 Spring Boot 应用，具体实现步骤如下。

**1. 新建 Spring Starter Project**

选择菜单 File|New|Spring Starter Project，打开如图 3.17 所示的 New Spring Starter Project 对话框。

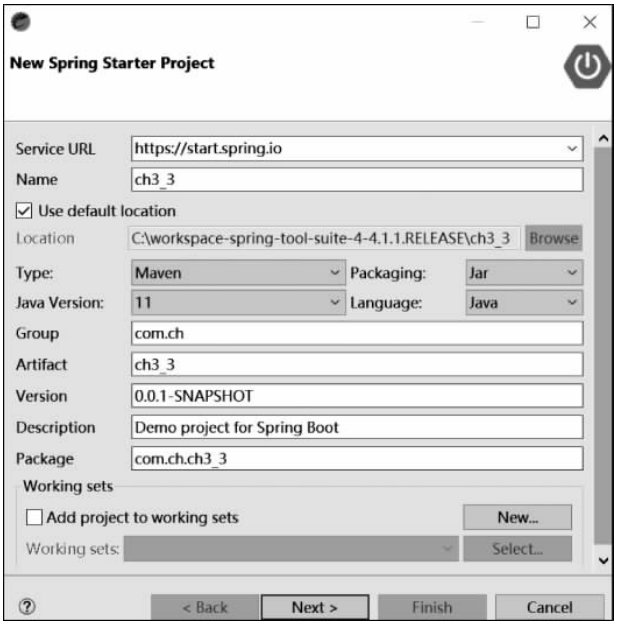



图 3.17 New Spring Starter Project

**2. 选择项目依赖**

在图 3.17 中输入项目信息后，单击 Next 按钮，打开如图 3.18 所示的 New Spring Starter Project Dependencies 对话框，并在图中选择项目依赖，如 Web。


图 3.18 New Spring Starter Project Dependencies

单击图 3.18 中的 Finish 按钮，即可完成 Web 应用的创建。ch3_3 的项目结构如图 3.19 所示。此时，就可以在项目 ch3_3 中编写自己的 Web 应用程序了。

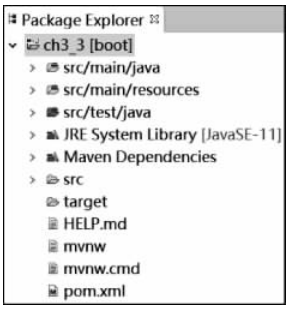



图 3.19　ch3_3 的项目结构

STS 是一款专为 Spring 定制的 Eclipse，并可以快速构建 Spring Boot 应用。因此，本书后续章节都采用 STS IDE 编写示例代码。

## 3.4　本章小结

本章首先简单介绍了 Spring Boot 应运而生的缘由，然后讲述了如何使用 Maven 手工构建一个 Spring Boot 应用，最后演示了如何使用 http://start.spring.io 或 Spring Tool Suite(STS)快速构建 Spring Boot 应用。开发者如何构建 Spring Boot 应用，可根据实际工程需要选择合适的 IDE。

## 习题 3

1. Spring、Spring MVC、Spring Boot 三者有什么联系？为什么还要学习 Spring Boot？
2. 在 Eclipse 中如何使用 Maven 手工构建 Spring Boot 的一个 Web 应用？