第 5 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

- SpringBoot 概述
- 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。为了解决上述问题，Spring Boot 应运而生。在编写本书时，Spring Boot 的最新正式版是 3.2.4，建议读者在测试本书示例代码时使用相同的版本。

5.1 Spring Boot 概述

▶5.1.1 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。

▶5.1.2 Spring Boot 的优点

Spring Boot 具有如下优点。

（1）使编码变得简单：推荐使用注解。

（2）使配置变得快捷：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。

▶5.1.3 Spring Boot 的主要特性

Spring Boot 的主要特性如下。

❶ 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。

❷ 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)。

❸ 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 包的形式部署应用。

❹ 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。

❺ 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。

❻ 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。

❼ 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，非常方便、快捷。

扫一扫

视频讲解

5.2　第一个 Spring Boot 应用

▶5.2.1　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。仓库主要包括中央仓库和本地仓库。

❶ 中央仓库

中央仓库存放开发过程中的所有 JAR 包，例如 JUnit，可以通过互联网从中央仓库中下载，仓库的地址为 http://mvnrepository.com。

❷ 本地仓库

本地仓库指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。

Maven 项目首先从本地仓库中获取所需要的 JAR 包，当无法获取指定 JAR 包时，本地仓库将从远程仓库(中央仓库)中下载 JAR 包，并存入本地仓库以备将来使用。

▶5.2.2　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：

❶ properties 元素

在<properties></properties>之间可以定义变量，以便在<dependency></dependency>中

引用，示例代码如下：

```
<properties>
    <!-- Spring 版本号 -->
    <spring.version>6.0.0</spring.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>
</dependencies>
```

❷ dependencies 元素

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

❸ dependency 元素

＜dependency＞＜/dependency＞元素内部通过＜groupId＞＜/groupId＞、＜artifactId＞＜/artifactId＞、＜version＞＜/version＞三个子元素确定唯一的依赖，也可以称为三个坐标。其示例代码如下：

```
<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>
```

❹ scope 子元素

在＜dependency＞＜/dependency＞元素中有时使用＜scope＞＜/scope＞子元素管理依赖的部署。＜scope＞＜/scope＞子元素可以使用以下 5 个值。

(1) compile(编译范围)：compile 是默认值，即默认范围。如果依赖没有提供范围，那么该依赖的范围就是编译范围。编译范围的依赖在所有的 classpath 中可用，同时也会被打包发布。

(2) provided(已提供范围)：provided 表示已提供范围，只有当 JDK 或者容器已提供该依赖时才可以使用。已提供范围的依赖不具有传递性，也不会被打包发布。

(3) runtime(运行时范围)：runtime 范围依赖在运行和测试系统时需要，但在编译时不需要。

(4) test(测试范围)：test 范围依赖在一般的编译和运行时都不需要，它们只有在测试编译和测试运行阶段可用，不会随项目发布。

(5) system(系统范围)：system 范围与 provided 范围类似，但需要显式地提供包含依赖的 JAR 包，Maven 不会在 Repository 中查找它。

▶5.2.3 使用 IntelliJ IDEA 快速构建 Spring Boot 应用

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

❶ 新建 Spring Project

打开 IDEA，通过选择 File→New→Project 打开新建项目窗口。在新建项目窗口的左侧选中 Spring Initializr 选项，打开如图 5.1 所示的窗口输入项目信息。

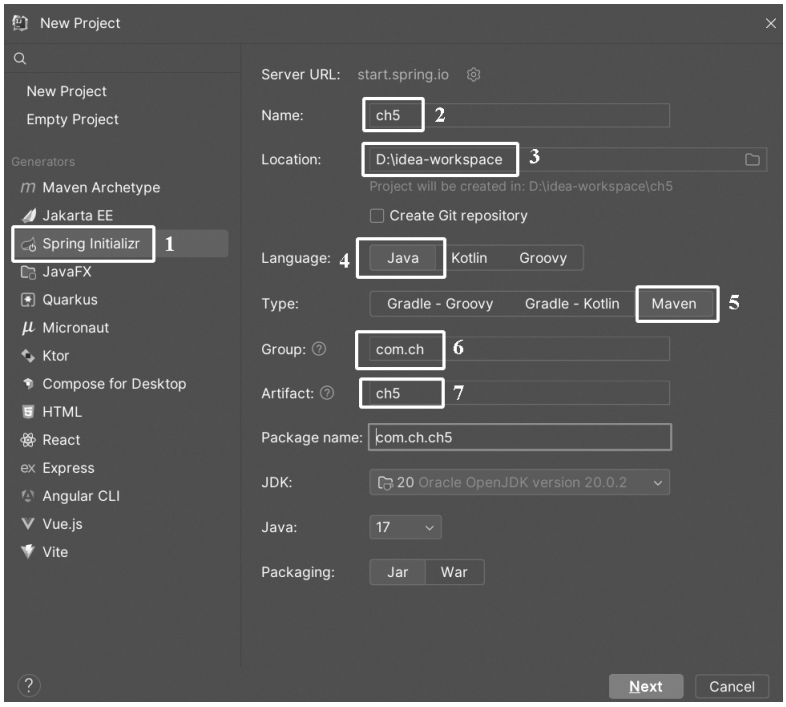

图 5.1　输入项目信息

❷ 选择项目依赖

在图 5.1 中输入项目信息后，单击 Next 按钮，打开如图 5.2 所示的选择项目依赖窗口。在图 5.2 中选择项目依赖（如 Spring Web）后，单击图 5.2 中的 Create 按钮，即可完成 Spring Boot Web 应用的创建。

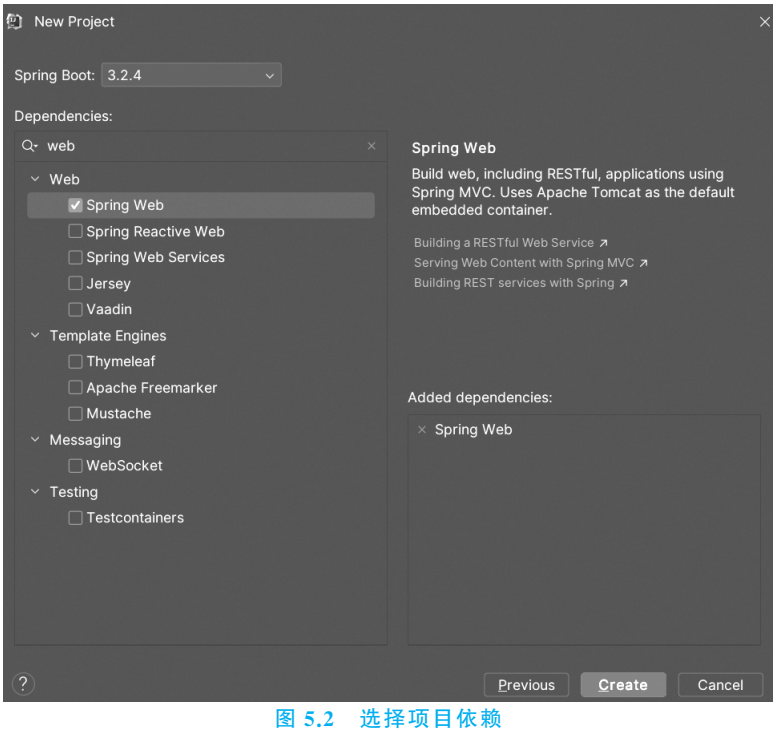

图 5.2　选择项目依赖

❸ 编写测试代码

在应用 ch5 的 src/main/java 目录下创建 com.ch.ch5.test 包，并在该包中创建 TestController 类，代码如下：

```
package com.ch.ch5.test;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class TestController {
    @GetMapping("/hello")
    public String hello() {
        return "您好，Spring Boot!";
    }
}
```

在上述代码中使用的@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、HTML 等视图，返回的内容就是 return 的内容。

（2）如果需要返回到指定页面，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@ResponseBody 注解。

❹ 应用程序的 App 类

在应用 ch5 的 com.ch.ch5 包中自动生成了应用程序的 App 类 Ch5Application。Ch5Application 的代码略。

❺ 运行 main 方法启动 Spring Boot 应用

运行 Ch5Application 类的 main 方法后，控制台信息如图 5.3 所示。

图 5.3 启动 Spring Boot 应用后的控制台信息

从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 3 内嵌 Tomcat 10，因此 Spring Boot Web 应用不需要开发者配置与启动 Tomcat。

❻ 测试 Spring Boot 应用

启动 Spring Boot 应用后，默认访问地址为 http://localhost:8080/，将项目路径直接设置为根路径，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@GetMapping("/hello")对应），测试效果如图 5.4 所示。

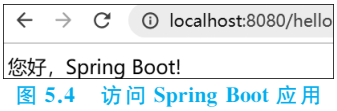

图 5.4 访问 Spring Boot 应用

本章小结

本章首先简单地介绍了 Spring Boot 应运而生的缘由，然后演示了如何使用 IDEA 快速构建 Spring Boot 应用。IDEA 在业界被公认为较好的 Java 开发工具，开发者可根据实际工程需要选择合适的 IDE 构建 Spring Boot 应用。

扫一扫

自测题

习题 5

1. Spring、Spring MVC、Spring Boot 三者有什么联系？为什么要学习 Spring Boot？
2. 在 IDEA 中如何快速构建 Spring Boot 的 Web 应用？

第 6 章 Spring Boot 核心

学习目的与要求

本章将详细介绍 Spring Boot 的核心注解、基本配置、自动配置原理以及条件注解。通过本章的学习，掌握 Spring Boot 的核心注解与基本配置，理解 Spring Boot 的自动配置原理与条件注解。

本章主要内容

- SpringBoot 的基本配置
- 读取应用配置
- SpringBoot 的自动配置原理
- Spring Boot 的条件注解

在 Spring Boot 产生之前，Spring 项目会存在多个配置文件，例如 web.xml、application.xml，应用程序自身也需要多个配置文件，同时需要编写程序读取这些配置文件。现在 Spring Boot 简化了 Spring 项目配置的管理和读取，仅需要一个 application.properties 文件，并提供了多种读取配置文件的方式。本章将学习 Spring Boot 的基本配置与运行原理。

扫一扫

视频讲解

6.1 Spring Boot 的基本配置

▶6.1.1 启动类和核心注解@SpringBootApplication

Spring Boot 应用通常都有一个名为 * Application 的程序入口类，该入口类需要使用 Spring Boot 的核心注解@SpringBootApplication 标注为应用的启动类。另外，该入口类有一个标准的 Java 应用程序的 main 方法，在 main 方法中通过“SpringApplication.run(* Application.class, args);”启动 Spring Boot 应用。

Spring Boot 的核心注解@SpringBootApplication 是一个组合注解，主要组合了@SpringBootConfiguration、@EnableAutoConfiguration 和@ComponentScan 注解。其源代码可以从 spring-boot-autoconfigure-x.y.z.jar 依赖包中查看 org/springframework/boot/autoconfigure/SpringBootApplication.java。

❶ @SpringBootConfiguration 注解

@SpringBootConfiguration 是 Spring Boot 应用的配置注解，该注解也是一个组合注解，源代码可以从 spring-boot-x.y.z.jar 依赖包中查看 org/springframework/boot/SpringBootConfiguration.java。在 Spring Boot 应用中推荐使用@SpringBootConfiguration 注解代替@Configuration 注解。

❷ @EnableAutoConfiguration 注解

@EnableAutoConfiguration 注解可以让 Spring Boot 根据当前应用项目所依赖的 JAR 包自动配置项目的相关配置。例如，在 Spring Boot 项目的 pom.xml 文件中添加了 spring-boot-starter-web 依赖，Spring Boot 项目会自动添加 Tomcat 和 Spring MVC 的依赖，同时对 Tomcat 和 Spring MVC 进行自动配置。打开 pom.xml 文件，单击窗口右侧的 Maven 即可查看 spring-boot-starter-web 的相关依赖，如图 6.1 所示。

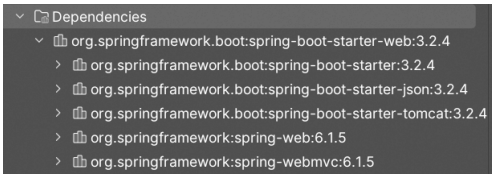

图 6.1　spring-boot-starter-web 的相关依赖

❸ @ComponentScan 注解

该注解的功能是让 Spring Boot 自动扫描@SpringBootApplication 所在类的同级包以及其子包中的配置，所以建议将@SpringBootApplication 注解的入口类放置在项目包（Group Id＋Artifact Id 组合的包名）中，并将用户自定义的程序放置在项目包及其子包中，这样可以保证 Spring Boot 自动扫描项目所有包中的配置。

▶6.1.2　Spring Boot 的全局配置文件

Spring Boot 的全局配置文件（application.properties 或 application.yml）位于 Spring Boot 应用的 src/main/resources 目录下。

❶ 设置端口号

全局配置文件主要用于修改项目的默认配置。例如，在 Spring Boot 应用 ch6 的 src/main/resources 目录下找到名为 application.properties 的全局配置文件，添加如下配置内容：

```
server.port=8888
```

可以将内嵌的 Tomcat 的默认端口改为 8888。

❷ 设置 Web 应用的上下文路径

如果开发者想设置一个 Web 应用程序的上下文路径，可以在 application.properties 文件中配置如下内容：

```
server.servlet.context-path=/XXX
```

这时应该通过“http://localhost:8080/XXX/testStarters”访问如下控制器类中的请求处理方法：

```
@GetMapping("/testStarters")
public String index() {
}
```

❸ 配置文档

在 Spring Boot 的全局配置文件中可以配置与修改多个参数，如果读者想了解参数的详细说明和描述，可以查看官方文档说明（https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#appendix.application-properties）。

▶6.1.3　Spring Boot 的 Starters

Spring Boot 提供了许多简化企业级开发的“开箱即用”的 Starters。Spring Boot 项目只要使用所需要的 Starters，Spring Boot 即可自动关联项目开发所需要的相关依赖。例如，在应用的 pom.xml 文件中添加如下依赖配置：

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

Spring Boot 将自动关联 Web 开发的相关依赖，如 Tomcat、spring-webmvc 等，进而对 Web 开发支持，并将相关技术的配置实现自动配置。

通过访问“https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using.build-systems.starters”官网，可以查看 Spring Boot 官方提供的 Starters。

除了 Spring Boot 官方提供的 Starters 外，还可以通过访问“https://github.com/spring-projects/spring-boot/blob/master/spring-boot-project/spring-boot-starters/README.adoc”网站查看第三方为 Spring Boot 贡献的 Starters。

扫一扫

视频讲解

6.2 读取应用配置

Spring Boot 提供了三种方式读取项目的 application.properties 配置文件的内容。这三种方式分别为使用 Environment 类、@Value 注解以及@ConfigurationProperties 注解。

▶6.2.1 Environment

Environment 是一个通用的读取应用程序运行时的环境变量的类，可以通过 key-value 方式读取 application.properties、命令行输入参数、系统属性、操作系统环境变量等。下面通过一个实例来演示如何使用 Environment 类读取 application.properties 配置文件的内容。

【例 6-1】 使用 Environment 类读取 application.properties 配置文件的内容。

其具体实现步骤如下。

❶ 创建 Spring Boot Web 应用 ch6

使用 IDEA 快速创建 Spring Boot Web 应用 ch6，同时给应用 ch6 添加如图 6.2 所示的依赖。在 6.2.3 节使用@ConfigurationProperties 注解读取配置时需要 Spring Configuration Processor 依赖，在此一起添加。

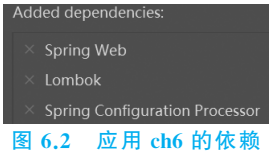

图 6.2 应用 ch6 的依赖

❷ 添加配置文件内容

在应用 ch6 的 src/main/resources 目录下找到全局配置文件 application.properties，并添加如下内容：

```
test.msg=read config
```

❸ 创建控制器类 EnvReaderConfigController

在应用 ch6 的 src/main/java 目录下创建名为 com.ch.ch6.controller 的包（com.ch.ch6 包的子包，保障注解全部被扫描），并在该包下创建控制器类 EnvReaderConfigController。在控制器类 EnvReaderConfigController 中使用@Autowired 注解依赖注入 Environment 类的对象，核心代码如下：

```
@RestController
public class EnvReaderConfigController{
    @Autowired
    private Environment env;
    @GetMapping("/testEnv")
    public String testEnv() {
        return "方法一:" + env.getProperty("test.msg") ;
        //test.msg 为配置文件 application.properties 中的 key
    }
}
```

❹ 启动 Spring Boot 应用

运行 Ch6Application 类的 main 方法，启动 Spring Boot 应用。

❺ 测试应用

在启动 Spring Boot 应用后，默认访问地址为 http://localhost:8080/，将项目路径直接设置为根路径，这是 Spring Boot 的默认设置。因此，可以通过 http://localhost:8080/testEnv 测试应用（testEnv 与控制器类 EnvReaderConfigController 中的@GetMapping("/testEnv")对应），测试效果如图 6.3 所示。

图 6.3　使用 Environment 类读取 application.properties 文件的内容

▶6.2.2　@Value

使用@Value 注解读取配置文件内容的示例代码如下：

```
@Value("${test.msg}")                    //test.msg 为配置文件 application.properties 中的 key
private String msg;                      //通过@Value 注解将配置文件中 key 对应的 value 赋值给变量 msg
```

下面通过实例讲解如何使用@Value 注解读取配置文件的内容。

【例 6-2】　使用@Value 注解读取配置文件的内容。

其具体实现步骤如下。

❶ 创建控制器类 ValueReaderConfigController

在 ch6 应用的 com.ch.ch6.controller 包中创建名为 ValueReaderConfigController 的控制器类，并在该控制器类中使用@Value 注解读取配置文件的内容，核心代码如下：

```
@RestController
public class ValueReaderConfigController {
    @Value("${test.msg}")
    private String msg;
    @GetMapping("/testValue")
    public String testValue() {
        return "方法二:" + msg;
    }
}
```

❷ 启动并测试应用

首先运行 Ch6Application 类的 main 方法，启动 Spring Boot 应用，然后通过 http://localhost:8080/testValue 测试应用。

▶6.2.3　@ConfigurationProperties

使用@ConfigurationProperties 首先建立配置文件与对象的映射关系，然后在控制器方法中使用@Autowired 注解将对象注入。

下面通过实例讲解如何使用@ConfigurationProperties 读取配置文件的内容。

【例 6-3】　使用@ConfigurationProperties 读取配置文件的内容。

其具体实现步骤如下。

❶ 添加配置文件内容

在 ch6 项目的 src/main/resources 目录下找到全局配置文件 application.properties，并添加