04 Spring Boot 的设计理念和简单实践

更新时间：2019-06-19 17:53:45

> 富贵必从勤苦得。
>
> ——杜甫

从前面的文章我们了解到 Spring Cloud 是依赖于 Spring Boot 构建的。本节内容会给大家详细介绍 Spring Boot 的来龙去脉，最后会用一个简单的例子让大家感受 Spring Boot 带来的便利性。

Spring Boot 介绍

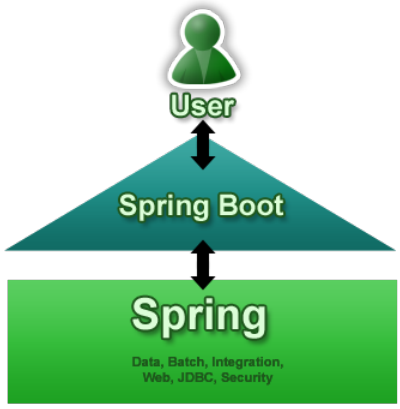
Spring Boot 是由 Pivotal 团队提供的全新框架。Spring Boot 设计之初就是为了以最少的配置、最快的速度来启动和运行 Spring 项目。Spring Boot 使用特定的配置来构建生产就绪型的项目。 Spring Boot 默认配置了很多框架的使用方式，就像 Maven 整合了所有的 Jar 包，Spring Boot 整合了所有的框架。

Spring Boot 的核心设计思想是：**约定优于配置**。Spring Boot 所有开发细节都是依据此思想进行实现，并且对已有的开发模式进行了大刀阔斧地改善，整体提升了研发效率。

> 约定优于配置（convention over configuration），也称作按约定编程，是一种软件设计范式，旨在减少软件开发人员需做决定的数量，获得简单的好处，而又不失灵活性。

Spring Boot 是一套全新的框架，它来自于 Spring 大家族，因此 Spring 所有具备的功能它都有并且更容易使用；Spring Boot 简化了基于 Spring 的应用开发，通过少量的代码就能创建一个独立的、产品级别的 Spring 应用。

下图展示出 Spring Boot 在 Spring 生态中的位置：



该项目主要的目的是：

- 让 Spring 的开发更快更广泛地上手
- 使用默认方式实现快速开发
- 提供大多数项目所需的非功能特性，诸如：嵌入式服务器、安全、心跳检查、外部配置等

Spring Boot 特性

- 使用 Spring 项目引导页面可以在几秒构建一个项目
- 方便对外输出各种形式的服务，如 REST API、WebSocket、Web、Streaming、Tasks
- 非常简洁的安全策略集成
- 支持关系数据库和非关系数据库
- 支持运行期内嵌容器，如 Tomcat、Jetty
- 强大的开发包，支持热启动
- 自动管理依赖
- 自带应用监控
- 支持各种 IDE，如 IntelliJ IDEA 、NetBeans

Spring Boot 诞生的背景

多年以来，Spring 平台饱受非议的一点就是大量的 XML 配置以及复杂的依赖管理。

随着使用 Spring 进行开发的个人和企业越来越多，Spring 也慢慢从一个单一简洁的小框架变成一个大而全的开源软件，Spring 的边界不断地进行扩充，到了后来 Spring 几乎可以做任何事情了，市面上主流的开源软件、中间件都有 Spring 对应组件支持。人们在享用 Spring 的便利之后，也遇到了一些问题。

Spring 每集成一个开源软件，就需要增加一些基础配置，慢慢地随着人们开发的项目越来越庞大，往往需要集成很多开源软件，后期使用 Spring 开发大型项目需要引入很多配置文件，太多的配置非常难以理解，并容易配置出错，到了后来人们甚至称 Spring 为配置地狱。

在 2013 年的 SpringOne 2GX 会议上，Pivotal 的 CTO Adrian Colyer 回应了这些批评，并且特别提到该平台将来的目标之一就是实现免 XML 配置的开发体验。Spring Boot 所实现的功能超出了这个任务的描述，开发人员不仅不再需要编写 XML，而且在一些场景中甚至不需要编写繁琐的 import 语句。

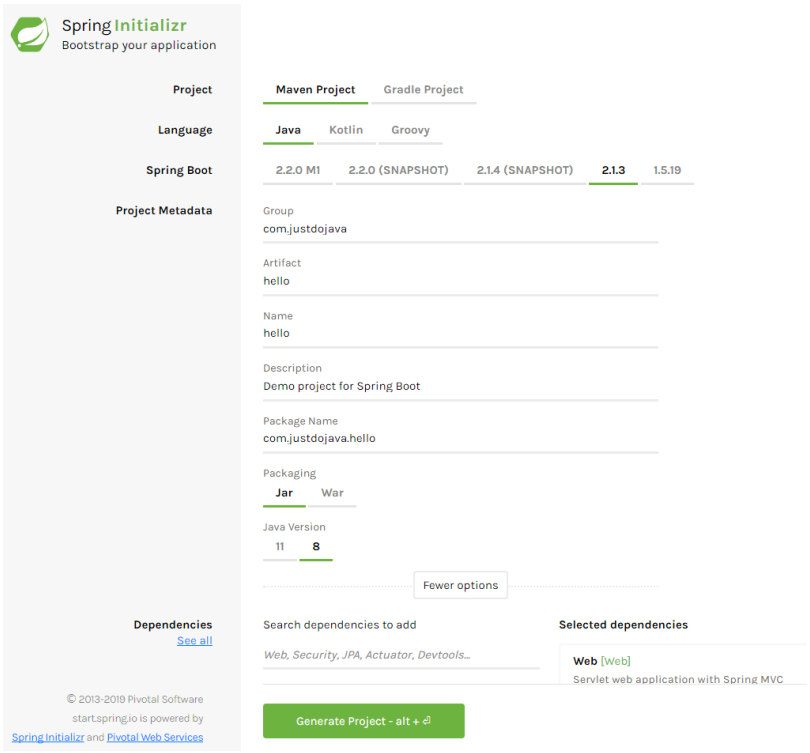
2013 年，微服务的概念也慢慢兴起，快速开发微小独立的应用变得更为急迫，Spring 刚好处在这么一个交叉点上。于 2013 年初开始的 Spring Boot 项目的研发，到2014 年，Spring Boot 伴随着 Spring 4.0 诞生发布了第一个正式版本。

简单实践

接下来我们通过一个很简单的例子，来感受 Spring Boot 快速开发的魅力。

构建项目模板

Spring Boot 官网提供了快速构建项目模板的网站：https://start.spring.io/，只需要在页面进行简单的选择即可构建一个 Spring Boot 模板项目。



页面中配置的信息如上图，下面会一一解释：

- Project，有两个选项，代表了使用 Maven 或者是 Gradle 来构建项目，默认使用 Maven；
- Language，Spring Boot 支持 Java\Kotlin\Groovy 编程语言构建项目，默认使用 Java；
- Spring Boot ，这里选择 Spring Boot 的版本，这里我们使用 2.X 系列的最新稳定版本；
- Project Metadata，配置项目的相关信息，配置信息有包名、项目名、描述信息等。需要注意的是 Packaging 代表了项目的打包方式，Jar 就是 Jar 包的形式来运行，或者选择 War 包的形式，Jdk 默认支持 1.8；
- Dependencies，添加项目依赖，可以通过页面输入关键字的形式来添加依赖组件。

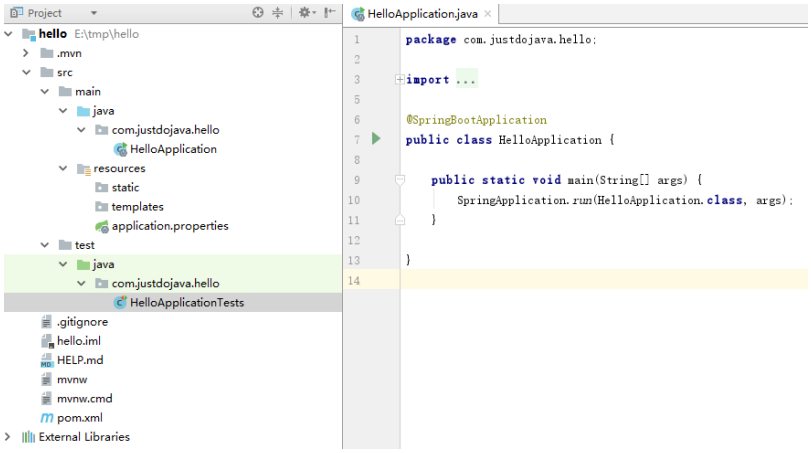
当上述信息配置完成之后，在页面点击 “Generate Project” 按钮或者使用快捷键 `alt + ⏎` 就会将配置好的项目模板下载下来。

> 在 IDEA 中使用 Spring Initalizr 插件也可以进行配置。

导入项目

上面下载的 Spring Boot 项目模板是一个压缩包，解压后导入 IDEA 中进行开发，IDEA 中选择 `File -> New -> Model from Existing Source.. -> 选择解压后的文件夹 -> OK`，选择 Maven 一路点击 Next，项目导入完成。

项目结构如下：



如上图所示，Spring Boot 的基础结构共三个文件:

- `src/main/java` 项目启动类和主要代码
- `src/main/resources` 配置信息目录
- `src/test/java` 测试程序

resources 目录下：

- static 目录存放 Web 访问的静态资源，如：js、css、图片等
- templates 目录存放页面模板
- `application.properties` 项目配置信息
- `pom.xml` 用于配置项目依赖包以及其它配置
- mvnw 文件 ， mvnw 全名是 Maven Wrapper。作用是当运行环境中找不到正确的 Maven 版本时，会自动下载 Maven 信息，之后再运行项目

开发 **Hello World** 服务

在页面构建的时候我们已经选择了添加 Web 依赖，因此在 pom 包会存在 `spring-boot-starter-web` 依赖信息：

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

接下来我们只需要编辑输出一个 Controller 代码即可，在目录 `src\main\java\com\justdojava\hello` 下创建 `HelloController`：

```
@RestController
public class HelloController {

    @RequestMapping("/hello")
    public String hello() {
        return "hello world";
    }

}
```

- `@RestController` 的意思是 Controller 里面的方法都以 Json 格式输出，不需要有其他额外的配置；如果配置为 `@Controller`，代表输出内容到页面
- `@RequestMapping("/hello")` 提供路由信息，"/hello"路径的 HTTP Request 都会被映射到`hello()`方法进行处理

这样一个 hello 服务就创建完成了。

启动测试

右键单击项目中的 HelloApplication | run 命令，就可以启动项目了，若出现以下内容表示启动成功：

```
"C:\Program Files\Java\jdk1.8.0_101\bin\java" ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.3.RELEASE)
...
2019-03-25 11:53:46.535  INFO 20936 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engin
e: [Apache Tomcat/9.0.16]
...
2019-03-25 11:53:46.962  INFO 20936 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring em
bedded WebApplicationContext
2019-03-25 11:53:46.963  INFO 20936 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationCon
text: initialization completed in 2804 ms
2019-03-25 11:53:47.798  INFO 20936 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port
(s): 8080 (http) with context path ''
2019-03-25 11:53:47.803  INFO 20936 --- [           main] com.justdojava.hello.HelloApplication    : Started HelloApplicati
on in 4.813 seconds (JVM running for 8.705)
```

根据控制台打印的信息，我们也可以得到一些关键信息：使用的是 2.1.3.RELEASE 版本，默认使用了 Tomcat 9 内嵌容器，项目的启动端口为 8080，启动时间 2804 ms 。

启动成功后，打开浏览器输入网址：`http://localhost:8080/hello`，就可以看到以下内容：

```
hello world
```

有没有发现构建一个应用是如此的简单。

Spring Boot 开发项目的优势

对比了上面的操作过程，我们可以明显地感触到使用 Spring Boot 开发项目的便利性，让我们在几分钟之内轻松构建一个简单服务。相比传统项目的开发流程，使用 Spring Boot 可以省掉很多中间环节。

使用 Spring Boot 开发项目的优势，又不仅仅局限于我们上面所看到的这些，它对研发的整个流程都进行了重构，在开发、部署、测试、运维方面均有优化，我们可以简单地提炼出几个核心优势：

- Spring Boot 使开发变简单，Spring Boot 提供了丰富的解决方案，快速集成各种解决方案提升开发效率；
- Spring Boot 使配置变简单，Spring Boot 提供了丰富的 Starters，集成主流开源产品往往只需要简单的配置即可；
- Spring Boot 使部署变简单，Spring Boot 本身内嵌启动容器，仅仅需要一个命令即可启动项目，结合 Jenkins 、Docker 自动化运维非常容易实现；
- Spring Boot 使监控变简单，Spring Boot 自带监控组件，使用 Actuator 轻松监控服务各项状态。

从软件发展的角度来讲，越简单的开发模式越会流行，简单的开发模式解放出更多生产力，让开发人员可以将精力集中在业务上，而不是各种配置、语法所设置的门槛上。Spring Boot 就是尽可能地简化应用开发的门槛。

Spring Boot 所集成的技术栈，几乎都是各互联网公司在使用的技术，跟着 Spring Boot 的路线去学习，基本可以了解国内外互联网公司的技术特点。

小结

Spring Boot 诞生一方面是因为 Spring 自身发展所遇到的问题，另一方面在微服务思想诞生之际，急需要一款快速开发工具来实现微服务技术落地，在这样的背景下诞生了 Spring Boot。

Spring Boot 是一套快速开发框架，按照约定优于配置的思想，整合了 Spring 生态以及周边产品，让大家以统一的编程体验来开发项目。因此它带来的开发效率提升是全方面的，Spring Cloud 正是依托于 Spring Boot 的这些特性构建的微服务体系。

本文作者：纯洁的微笑、江南一点雨

03 Spring Cloud 介绍以及发展前景

05 Spring Cloud 和 Spring Boot 之间的密切关系