## Spring Boot 项目搭建与Redis结合与应用

### Spring Boot 简介

微服务是一种架构风格，是一种使用一套微小服务来开发单个应用的方式途径，每个服务运行在自己的进程中，通过轻量的通讯机制联系，经常是基于HTTP资源API，这些服务基于业务能力构建，能够通过自动化部署方式独立部署。微服务，简单的说就是将一个大服务（项目）划分为多个子（服务）项目，如下图所示。

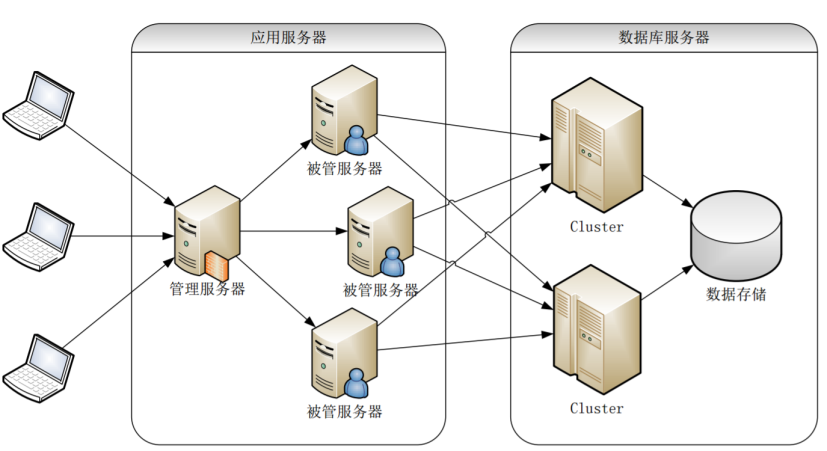


图 8‑1 典型微服务架构

从上图可以看出微服务的好处，每个微服务组件都是简单灵活的，能够独立部署。微服务应用不需要一个庞大的应用服务器来支撑。微服务之间是松耦合的，内部是高内聚的，每个微服务很容易按需扩展。微服务架构与语言工具无关，可以自由选择合适的语言和工具，高效的完成业务目标。

Spring Boot是 Spring 的一套快速配置脚手架，可以基于Spring Boot快速开发单个微服务，Spirng Boot 本身并不提供 Spring 框架的核心特性以及扩展功能，只是用于快速、敏捷地开发新一代基于 Spring 框架的应用程序，如下图所示。也就是说，它并不是用来替代 Spring 的解决方案，而是和 Spring 框架紧密结合用于提升 Spring 开发者体验的工具。

Spring Boot在开发过程中大量使用“约定优先配置”的思想来摆脱Spring框架中各类繁复纷杂的配置。采用 Spring Boot 可以大大的简化开发模式，同时它集成了大量常用的第三方库配置（例如 Redis、MongoDB、Jpa、RabbitMQ、Quartz 等等），Spring Boot 应用中这些第三方库几乎可以零配置的开箱即用，大部分的 Spring Boot 应用都只需要非常少量的配置代码，开发者能够更加专注于业务逻辑。

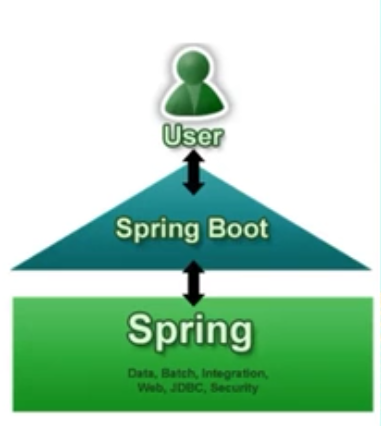


图 8‑2 Spring Boot Spring的关系

Spring Boot不等于微服务，它只是一套开源框架，跟SSM（Spring+Struts2+MyBatis）差不多，只是基于Spring Boot来开发微服务相当方便，所以这两个词一般都是成对出现的。当我们的服务越来越多时，就可以通过Spring Cloud来统一管理这些服务了，Spring Cloud才算是真正的微服务框架，Spring Boot是J2EE一站式解决方案。使用SpringBoot可以提供微服务开发小型的web项目。

### 使用Spring Initalizr新建项目

本节介绍在Windows下使用Spring Boot, Spring Boot 依赖的开发环境如表所示。

|  |  |
| --- | --- |
| 操作系统 | Windows 10 64位平台 |
| JDK | 1.8 |
| Maven | 3.6.0 |

新建一个Spring Boot项目，可以使用Spring Initalizr方式，这种方式很简单，通过在线网页设置项目的选项，自动生成Spring Boot项目。

首先在 <http://start.spring.io/> 上创建一个简单的SpringBoot应用。类型为Maven Project、开发语言为Java和Spring Boot版本为2.2.0M6，以及填写项目的基本信息Group和Artifact，如下图所示。最后点击”Generate the Project”按钮下载springbootDemo.zip压缩包。

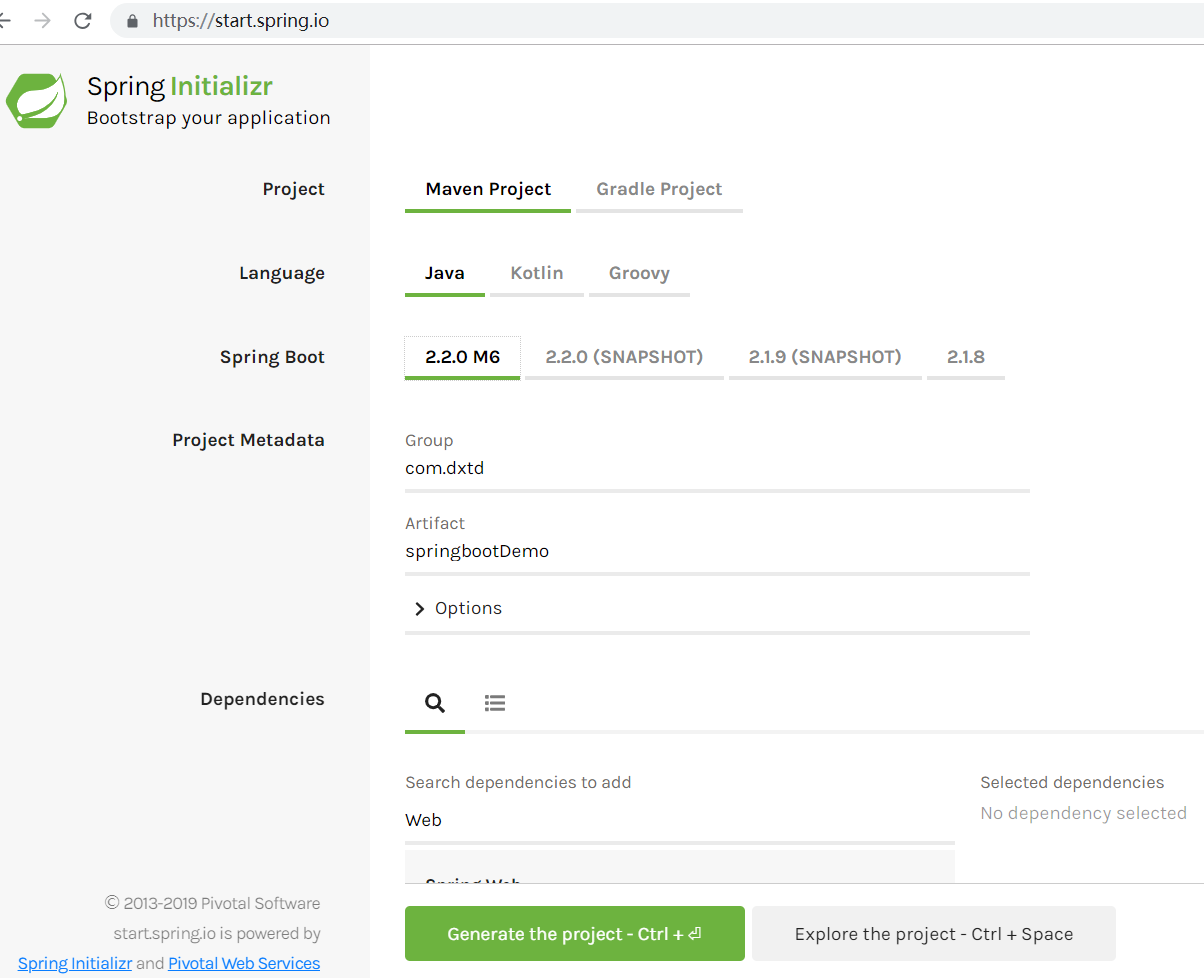


图 8‑3 创建Spring Boot 应用

本书主要以Intellij IDEA进行实验，把springbootDemo.zip进行解压缩到本地硬盘，然后使用Intellij IDEA导入工程，File-> open -> 选择springbootDemo在本地硬盘路径，如果是第一次配置 Spring Boot 的话可能需要等待一会儿 IDEA 下载相应的依赖包，默认创建好的项目结构如下图所示。

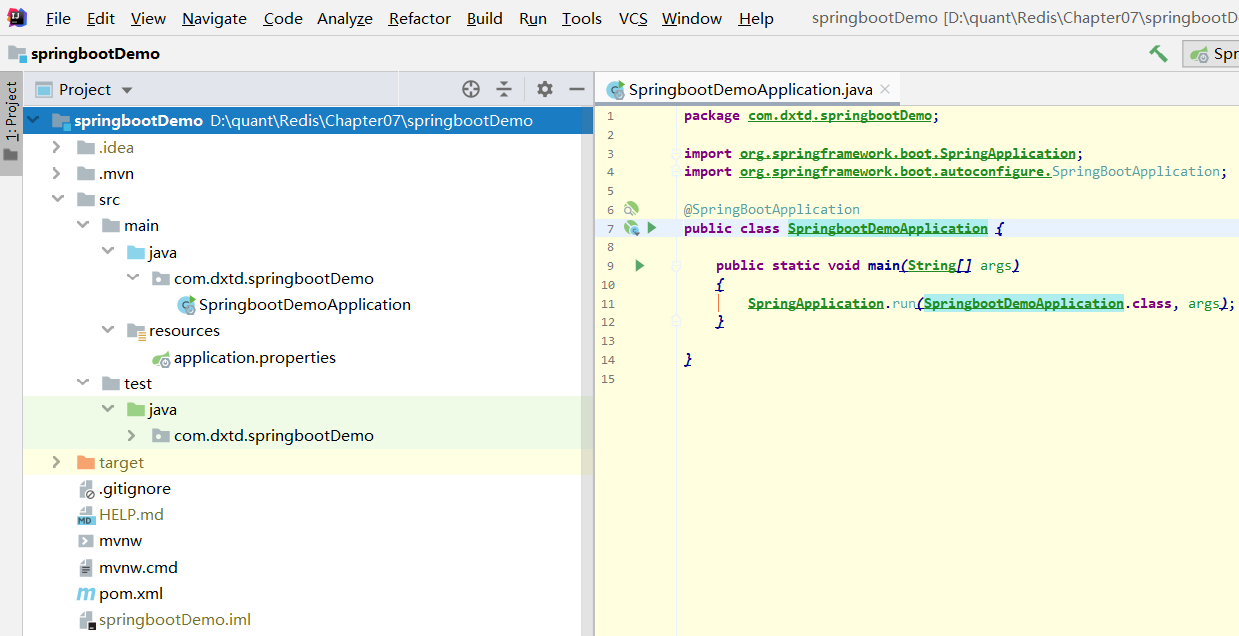


图 8‑4 Intellij Idea导入SpringBoot工程

springbootDemo的项目结构很清晰，相比Java EE项目少了很多配置文件，默认生成的文件主要有4个。

* SpringbootDemoApplication.java ： 一个带有 main() 方法的类，用于启动应用程序。
* SpringbootDemoApplicationTests.java：一个空的 Junit 测试类，它加载了一个使用。 Spring Boot 字典配置功能的 Spring 应用程序上下文。
* application.properties：一个空的 properties 文件，可以根据需要添加配置属性。
* pom.xml： Maven 构建项目的说明文件。

最后引入web模块，传统基于Java Web应用需要配置 web.xml,讲web应用打成war包放入应用服务器(Tomcat,Weblogic等)中运行。如果基于SPring Boot开发为服务，将变得简单，以Maven项目为例，需要在pom.xml中引入Spring Boot的开发依赖模块  。

<dependency>

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

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

</dependency>

pom.xml文件中默认有两个模块：

* spring-boot-starter：核心模块，包括自动配置支持、日志和YAML；
* spring-boot-starter-test：测试模块，包括JUnit、Hamcrest、Mockito。

### 运行与发布

SpringbootDemoApplication是Spring Boot项目的入口类，它的关键源代码如下：

@SpringBootApplication

public class SpringbootDemoApplication {

public static void main(String[] args)

{

SpringApplication.run(SpringbootDemoApplication.class, args);

}

}

其中，@SpringBootApplication注解是SpringBoot启动项目时需要加上的。需要说明的是SpringbootDemoApplication是整个Spring Boot应用启动的初始点，因此如果在应用启动格式需要进行某些资源初始化处理，那么最好都在该类完成。

双击SpringbootDemoApplication，在代码区域鼠标右键运行“Run ‘SpringbootDemoApplic’” 启动Spring Boot应用，如下图所示。

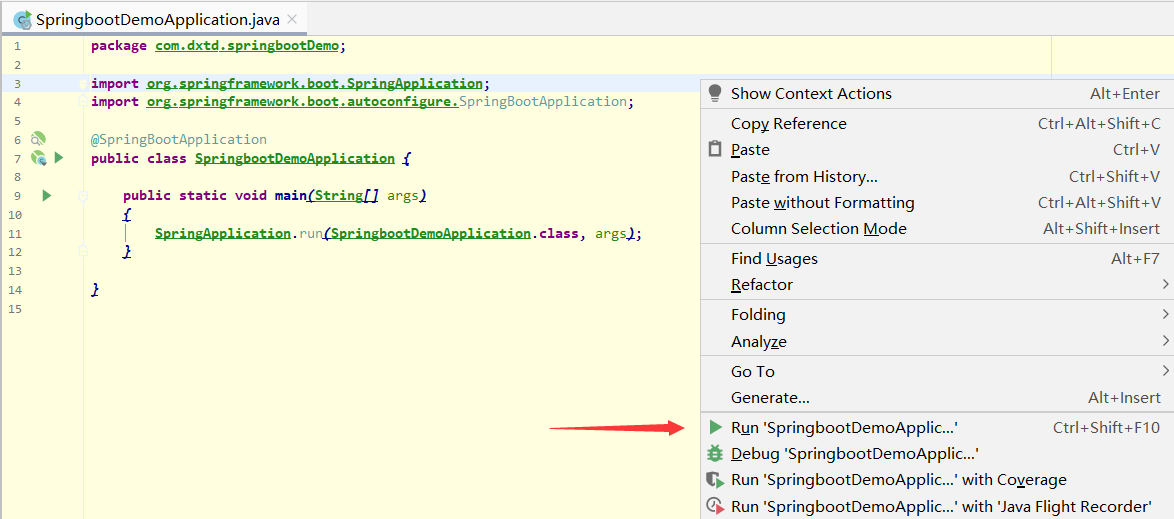


图 8‑5 运行Spring Boot 的入口类

我们可以看到后台日志在IDEA的控制台打印出如下内容。

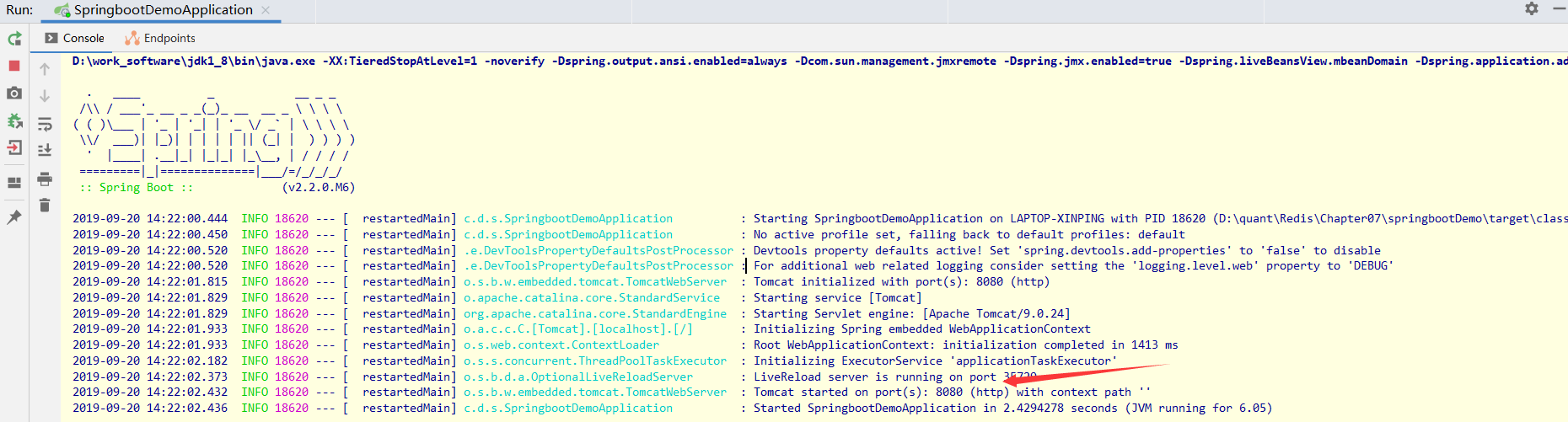


图 8‑6 IDEA控制台

从上图的输出可以看出 Tomcat服务器 默认开启了8080端口，要访问这个应用提供的服务，可以在浏览器的地址栏中输入 <http://localhost:8080/> ,我们在项目中没有手动的去配置 Tomcat 服务器，是因为Spring Boot 内置了Tomcat服务器。

如果要把应用部署到服务器上，还需要对项目进行发布，通过Maven的mvn命令对整个Spring Boot 项目进行项目编译，单元测试和打包，然后运行打包后的jar文件。我们切换到项目所在的目录然后运行mvn package命令，如下图所示。

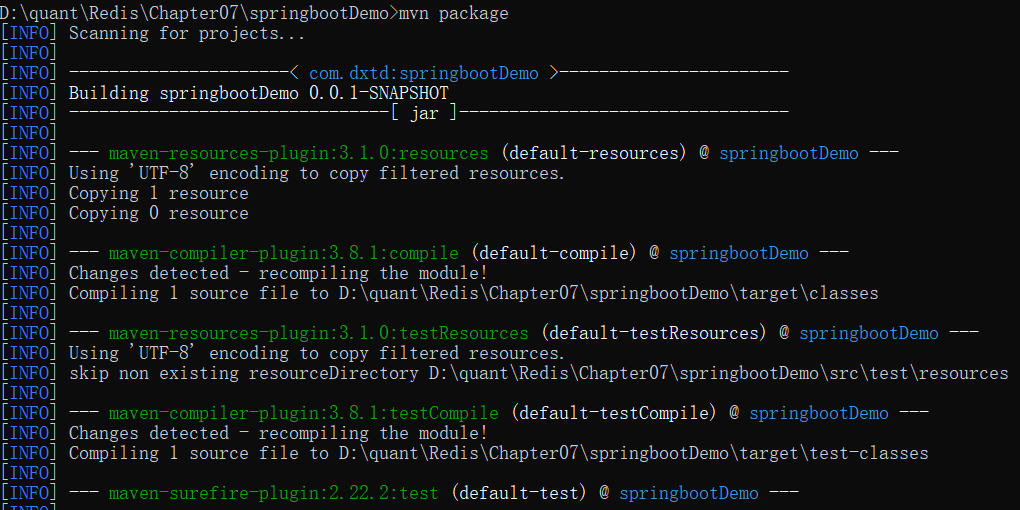


图 8‑7 编译Spring Boot项目

打包成功后，在工程的target目录中会生成jar文件springbootDemo-0.0.1-SNAPSHOT.jar，如下图所示。

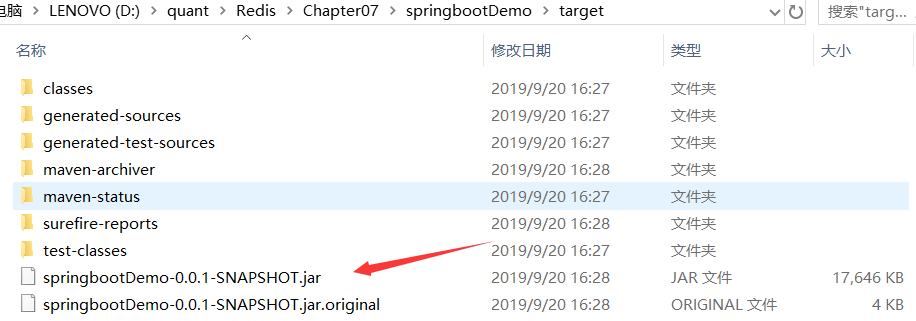


图 8‑8 打包成功后的target目录

在命令行窗口切换到项目的target目录中，运行如下命令就可以启动应用，这样应用就发布成功了。启动的效果如下图所示。

java -jar springbootDemo-0.0.1-SNAPSHOT.jar

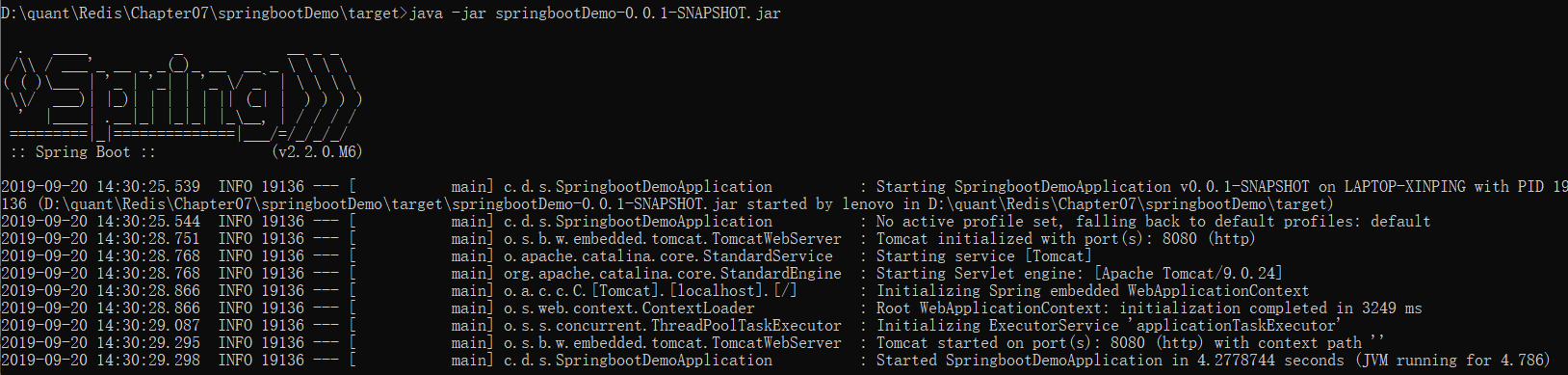


图 8‑9 通过命令行启动SpringBoot应用

### 编写控制器Controller

Spring Boot 应用可以提供很多服务，不同的服务对应不同的URL，对于不通的URL通常需要不同的方法进行处理并返回不通的内容，编写HelloWorldController类，源码的具体内容如下。

package com.dxtd.springbootDemo.controller;

import org.springframework.web.bind.annotation.RequestMapping;

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

@RestController

public class HelloWorldController {

@RequestMapping("/hello")

public String index() {

return "Hello spring boot";

}

}

服务提供者的代码非常简单，就是一个普通的RestController,当用户请求/hello时，就返回一个Hello Spring boot的字符串。

@RestController注解是控制器(controller)里面的方法都以json格式输出，不用在写jackjson配置。

@RequestMapping注解用来处理请求地址映射，可用于类或方法上。@RequestMapping注解用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径，@RequestMapping注解用于方法上，表示对外提供服务的地址。

启动主程序SpringbootDemoApplication，打开浏览器访问http://localhost:8080/hello，就可以看到效果了，如下图所示。

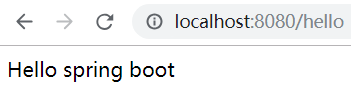


图 8‑10 访问Spring Boot的服务

### 开发环境的调试-热部署

热部署在开发项目时非常方便，使用热部署后可以使代码修改后，无须重启服务器，就可以加载更改的代码。只要在web页面中直接按F5键刷新，就可以显示新修改的内容了。

SpringBoot对热部署调试支持很好，修改Java代码后可以实时生效，只需要在pom.xml文件中添加spring-boot-devtools模块，spring-boot-devtools会自动部署新代码，监听classpath下的文件变动，并会立即重启应用，配置如下。

<dependency>

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

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

<scope>runtime</scope>

</dependency>

如果使用Intellij IDEA开发Spring Boot项目，在项目使用了spring-boot-devtools模块，此时修改了java文件，IDEA是不会自动热部署的。因为iIntellij IEDA和Eclipse不同，Eclipse一般设置了自动编译，而IDEA需要手工打开自动编译，否则热部署不会生效，步骤如下：

第一步： 在IDEA设置中打开自动编译。

在IDEA 界面点击File -> setting -> Build,Execution, Deployment -> Compiler 勾选 Build Project Automatically。

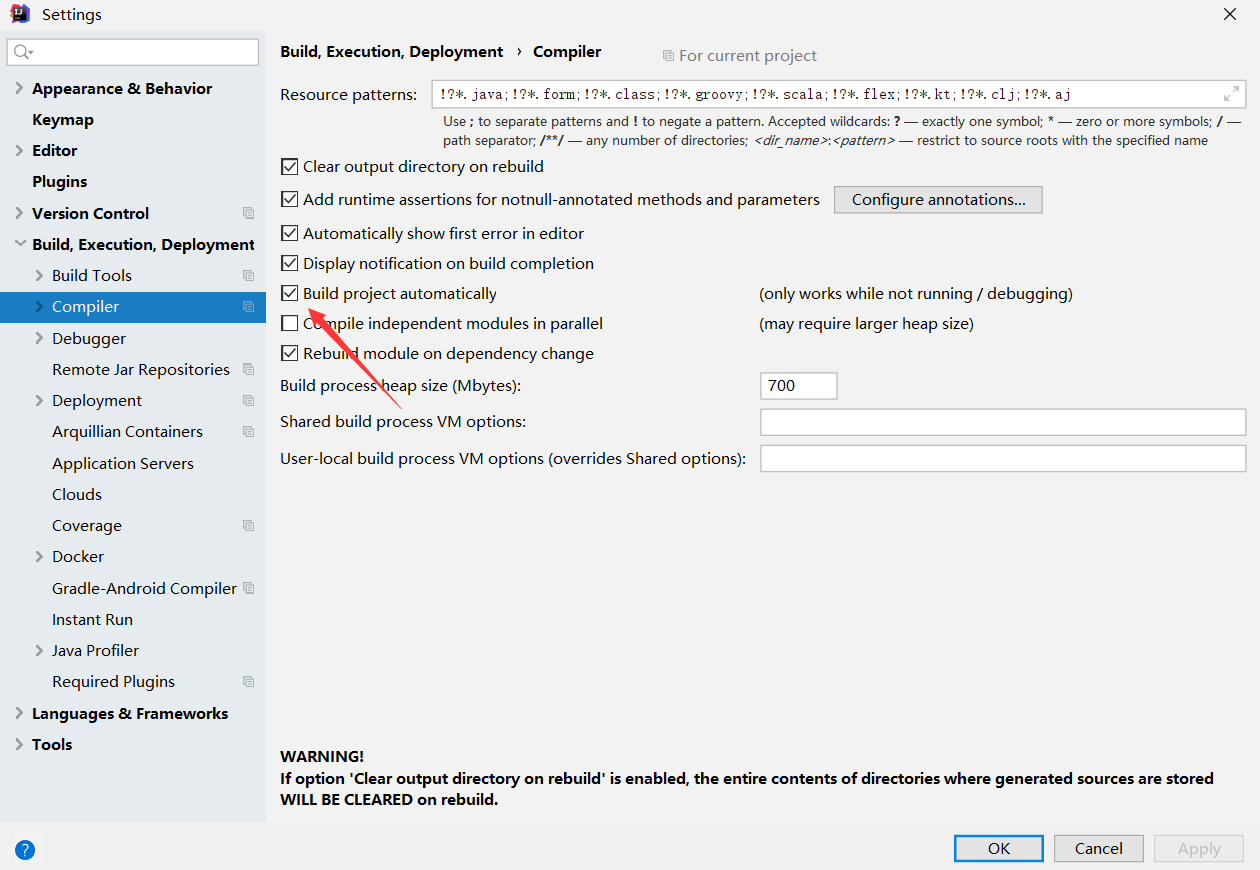


图 8‑11 Intellij IDEA自动编译项目

第二步：在IDEA中打开运行时编译。

在IDEA界面按Ctrl + Shift + Alt + / 调出Maintenance(维护)控制台选择Registry(登记)，如下图所示。

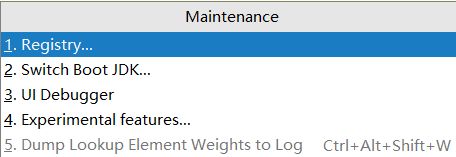


图 8‑12 调出Intellij IDEA的Maintenance控制台

勾选运行时自动编译(auto-making when app running)，然后重启IDEA。

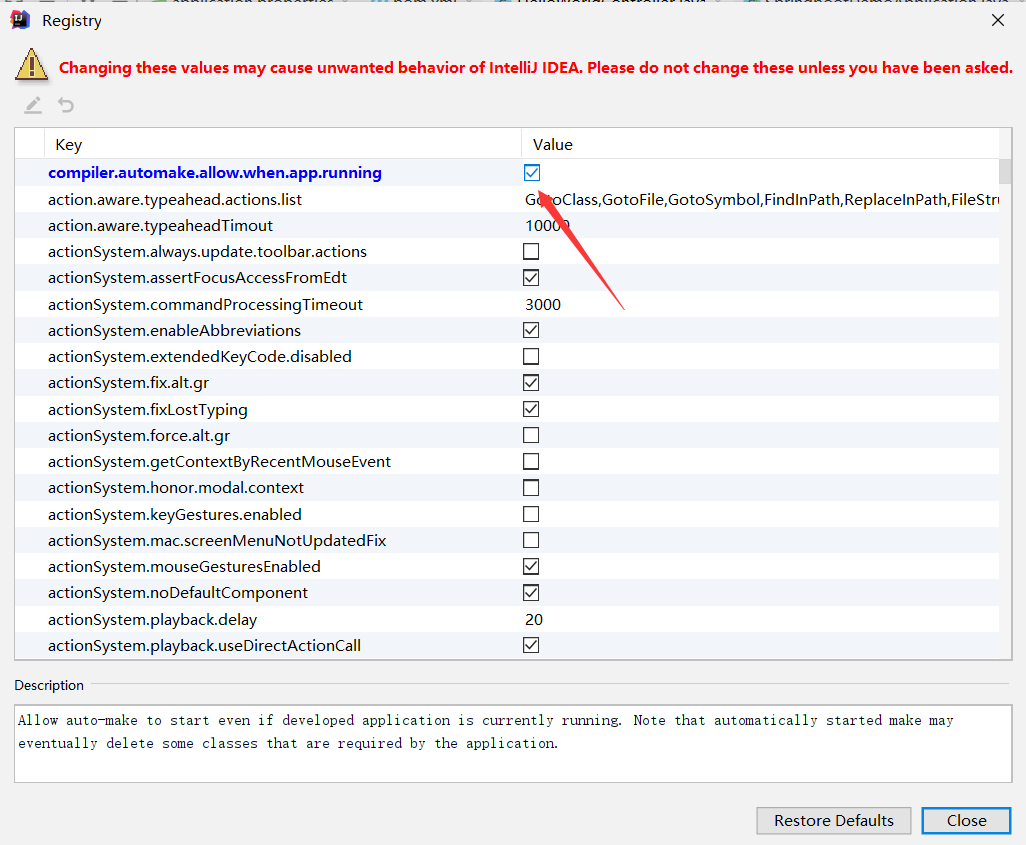


图 8‑13 勾选中运行时自动编译

设置完成后修改Java代码按Ctrl + F9, 就可以立即实现Spring Boot的热部署更新了。