微服务基础框架之

Spring boot 快速入门

普元：国内领先的软件基础平台与解决方案提供商

普元 云计算&SOA

平台 · 让创新无限

PRIMETON · 普元

课程简介

微服务框架众多，而基于Spring boot的Spring Cloud应用最广，国内基于Spring开发开发项目最多。而Spring Boot 由于其简单易用和Spring用户的基础，得以快速的普及。本系列微课程，将会系统详细的介绍Spring boot开发使用、配置、部署，扩展机制以及功能特性等，让我们在工作和项目中能够灵活的应用和驾驭它！

目录

Spring boot与微服务

微服务

- 是一种架构风格
- 灵活多样
- 松散耦合
- 细粒度服务
- 轻量级协议
- …

Spring Boot

- 是一个基于Spring的框架
- 简单易用，扩展性强
- 按需依赖
- HTTP、RESTFul
- …

准备工作

1. 首先需要安装java jdk 1.6以上的版本
2. Maven或者Gradle二选一
3. 自己顺手的IDE（如Eclipse，IntilliJ IDEA）
4. 安装Spring Boot Cli(非必须，可以使用sdkman进行安装)

注:

1. Spring boot cli 工具不是必须的，本次只是用它来演示，如何创建一个最简单的Spring boot 应用
2. 如果使用Eclipse作为开发工具，强烈推荐使用Spring的官方IDE工具Spring tools suite

使用Spring boot cli创建最简单的Spring应用

1. 创建一个Groovy文件app.groovy，内容右图所示:

2. 执行Spring boot cli命令,如图所示

3. 再访问http://127.0.0.1:8080/

app.groovy

```groovy
@RestController
class ThisWillActuallyRun {

    @RequestMapping("/")
    String home() {
        "Hello World!"
    }
}
```

```
14:58:08 ~/tmp/spring
$ spring run app.groovy

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.4.2.RELEASE)

2017-04-28 14:59:10.411  INFO 70045 --- [       runner-0] o.s.boot.SpringApplication
dom/tmp/spring)
2017-04-28 14:59:10.419  INFO 70045 --- [       runner-0] o.s.boot.SpringApplication
2017-04-28 14:59:10.741  INFO 70045 --- [       runner-0] ationConfigEmbeddedWebApplicatio
ApplicationContext@55de94d: startup date [Fri Apr 28 14:59:10 CST 2017]; root of context h
2017-04-28 14:59:12.752  INFO 70045 --- [       runner-0] s.b.c.e.t.TomcatEmbeddedServletC
```

使用Maven创建一个Spring boot应用

1. 使用自己钟爱的一款IDE，创建一个Maven项目
2. 添加spring-boot-starter-parent为父项目
3. 设置编译所要使用的java版本
4. 添加Spring boot的应用依赖
5. 添加Spring boot构建可执行jar包的插件依赖

```xml
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.primeton.sample</groupId>
    <artifactId>boot-sample</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <properties>
        <!-- 默认使用1.7编译，生成1.7的target -->
        <maven.compiler.source>1.7</maven.compiler.source>
        <maven.compiler.target>1.7</maven.compiler.target>
    </properties>
    <!-- 添加项目继承Spring boot starter parent -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.2.RELEASE</version>
    </parent>
    <!-- 添加项目的依赖 -->
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- 创建一个可执行的jar包的插件 -->
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

使用Maven创建一个Spring boot应用

1. 如果你的项目是一个已经有一个父项目，不能继承spring-boot-starter-parent
2. 则可以添加如右图所示的配置
3. 在Spring boot的插件依赖中

```
<dependencies>
    <!-- 添加项目依赖 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>${spring-boot.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

使用Maven创建一个Spring boot应用

4. 注意：如果使用dependencyManagement的方式引入Spring boot，则需要需要添加创建executable jar的配置

```
<!--Spring Boot 插件-->
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <version>${spring-boot.version}</version>
    <configuration>
        <executable>true</executable>
    </configuration>
    <executions>
        <execution>
            <goals>
                <!--使用Spring boot重新打包-->
                <goal>repackage</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

添加应用的启动代码程序

1. 创建一个包(**必须**)
2. 创建一个Java入口类（如右图所示）
3. 可以直接在IDE里面进行执行这个main方法
4. 也可以在命令行执行
   mvn spring-boot:run
5. 还可以打成jar包进行执行
   mvn package
   java –jar app.jar

```java
package com.primeton.sample;

import ...

/**
 * @author <a href='mailto:huzd@primeton.com'>虎振东</a>
 * @date 2017/5/16 09:56.
 */
@RestController
@SpringBootApplication
public class SampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(SampleApplication.class, args);
    }

    @GetMapping("/hello")
    public Object hello(String name) {
        return String.format("Hello %s !", name);
    }

}
```

使用Gradle创建一个Spring boot应用

1. 还是使用自己钟爱的一款IDE，创建一个Gradle项目(也可以直接在前面的Maven项目里创建build.gradle文件)
2. build.gradle文件的内容如右图所示
3. 在项目目录下执行:
    1. gradle buid
4. 使用 命令gradle bootRun执行项目
5. 当然也可以直接使用java –jar命令进行执行

```
plugins {
    id 'org.springframework.boot' version '1.5.2.RELEASE'
    id 'java'
}

jar {
    baseName = 'spring-boot-sample'
    version =  '1.0.0-SNAPSHOT'
}

repositories {
    mavenLocal()
    mavenCentral()
}

dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    testCompile("org.springframework.boot:spring-boot-starter-test")
}
```

Spring Boot 构建项目配置详解-准备

首先我们在项目的依赖中添加spring-boot-actuator用做我们部分功能的测试或者验证

Maven依赖:

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

Gradle依赖

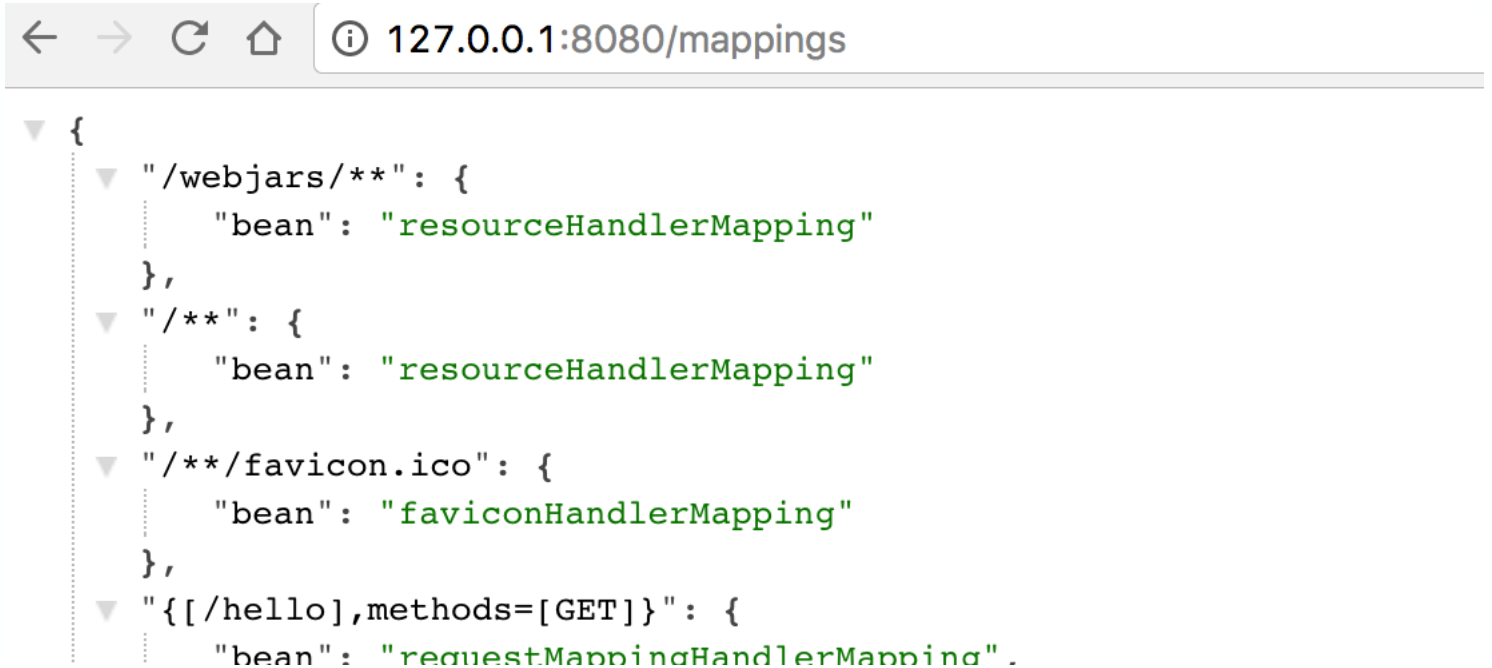
```
dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    compile("org.springframework.boot:spring-boot-starter-actuator")
    testCompile("org.springframework.boot:spring-boot-starter-test")
}
```

Spring Boot 构建项目配置详解-准备

在src/main/resources目录下创建application.yml配置文件,并添加如下配置跳过安全验证：

```yaml
management:
  security:
    enabled: false
```

执行应用，我们就可以通过端点来访问到应用信息，如图所示：我们可能查看到所有的RequestMappings信息



Spring Boot 构建项目配置详解-添加项目构建信息

我们可以将我们的项目构建信息打包到jar包中去，通过Spring boot 的InfoEndpoint就可以访问到我们的项目相关信息

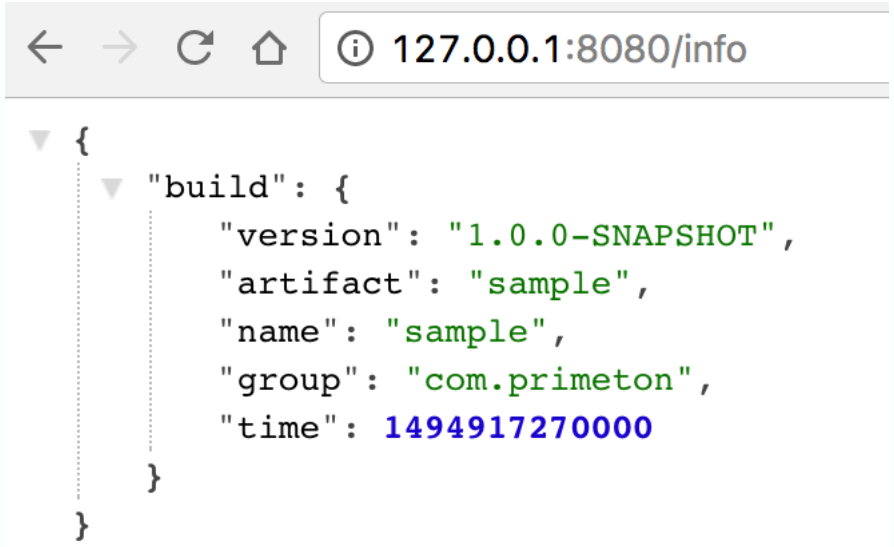
```
<!--Spring Boot 插件-->
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <version>${spring-boot.version}</version>
    <configuration>
        <executable>true</executable>
    </configuration>
    <executions>
        <execution>
            <goals>
                <!--使用Spring boot重新打包-->
                <goal>repackage</goal>
                <!--打包应用构建信息-->
                <goal>build-info</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Gradle配置

```
springBoot {
    buildInfo()
}
```

Spring Boot 构建项目配置详解-添加项目构建信息

通过端点来访问项目构建信息



Spring Boot 构建项目配置详解-GIT版本信息

还可以生成Git 版本信息

在Maven添加插件:

```
<plugin>
    <groupId>pl.project13.maven</groupId>
    <artifactId>git-commit-id-plugin</artifactId>
</plugin>
```

在gradle里添加插件:

```
plugins {
    id 'org.springframework.boot' version '1.5.2.RELEASE'
    id "com.gorylenko.gradle-git-properties" version "1.4.17"
    id 'java'
}
```

打包运行后的结果：


```
{
  "git": {
    "commit": {
      "time": 1494915074000,
      "id": "d6d596d"
    },
    "branch": "master"
  },
  "build": {
    "version": "1.0.0-SNAPSHOT",
    "artifact": "sample",
    "name": "sample",
    "group": "com.primeton",
    "time": 1494917907000
  }
}
```

Spring Boot 构建项目配置详解-创建可执行jar

Maven配置：

如果是继承了spring-boot-starter-parent，那我们什么都不用做，默认打出来的就是可执行jar

如果是使用dependencyManagement引入依赖，则需要添加一下配置信息：

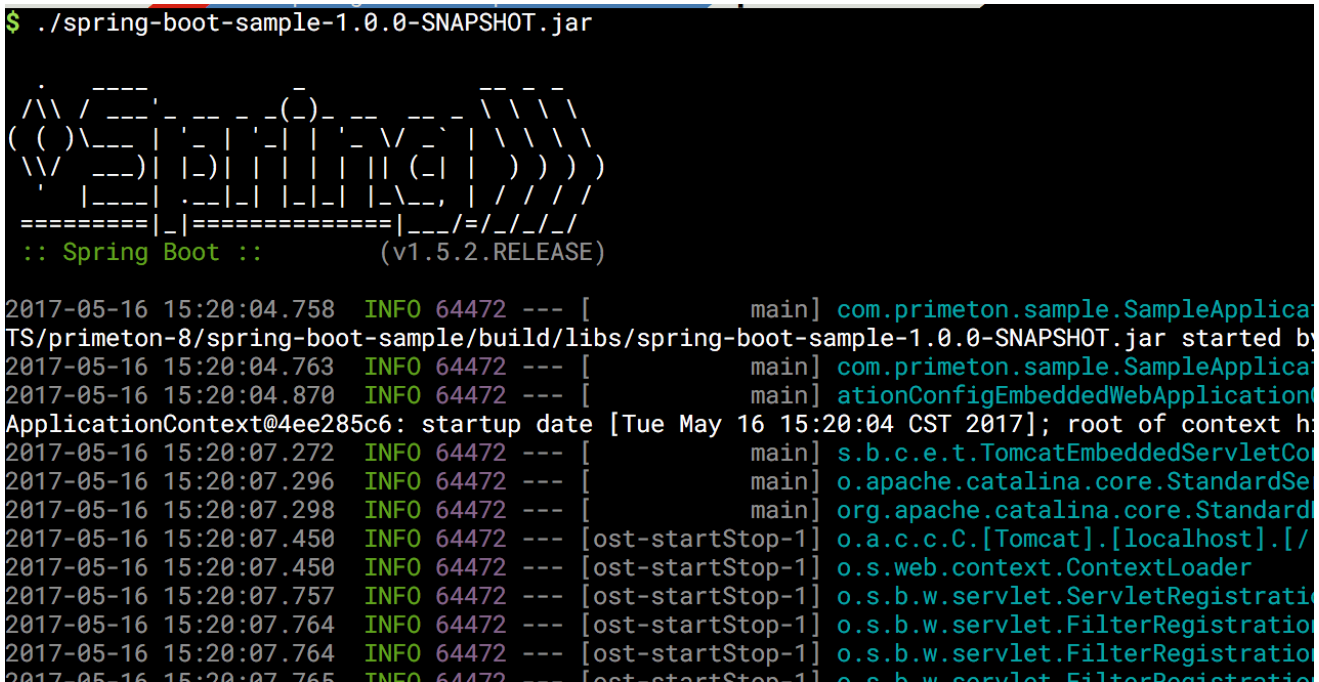
```
<!--Spring Boot 插件-->
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <version>${spring-boot.version}</version>
    <configuration>
        <executable>true</executable>
    </configuration>
    <executions>
        <execution>
            <goals>
                <!--使用Spring boot重新打包-->
                <goal>repackage</goal>
                <!--打包应用构建信息-->
                <goal>build-info</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Spring Boot 构建项目配置详解-创建可执行jar

Gradle的插件配置非常简单:

```
bootRepackage {
    executable = true
}
```

使用maven package或者gradle buid后，生成的jar可以这样执行：



使用Maven构建一个需要注意的地方

当我们使用Maven构建的时候，如果使用spring-boot-starter-parent作为项目parent的话，则插件的配置，只需要简单的导入即可

但是如果使用的是dependencyManagement做依赖的话，插件是没有办法进行继承的，所有，在我们使用dependencyManagement方式的时候，不仅要将插件引入进来，而且也需要对插件进行配置，例如：repackage

```
<!--Spring Boot 插件-->
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <version>${spring-boot.version}</version>
    <configuration>
        <executable>true</executable>
    </configuration>
    <executions>
        <execution>
            <goals>
                <!--使用Spring boot重新打包-->
                <goal>repackage</goal>
                <!--打包应用构建信息-->
                <goal>build-info</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Spring Boot部署到Tomcat等应用服务器

如果说，我们需要将我们Spring boot引用部署到像tomcat这样的应用服务器中，我们需要做以下几步：

1. 首先需要将我们的打包方式修改为war
2. 入口类需要继承SpringBootServletInitializer，并覆盖configrue方法
3. 这样我们打出的war包，就可以和普通的war包一样，部署到tomcat等应用服务器中

```
@RestController
@SpringBootApplication
public class SampleApplication extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(SampleApplication.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(SampleApplication.class, args);
    }

    @GetMapping("/hello")
    public Object hello(String name) {
        return String.format("Hello %s !", name);
    }

}
```

谢谢

平台 · 让创新无限

上海 · 北京 · 广州 · 西安 · 武汉 · 成都 | ☎ 400 820 5821 | 普元软件 | 普元信息

PRIMETON · 普元