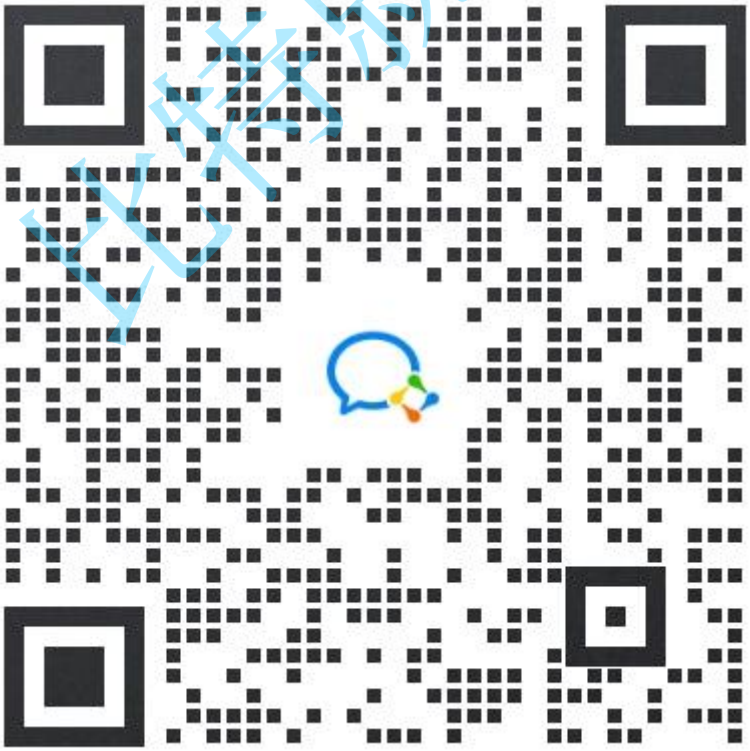
SpringBoot 微服务镜像制作

版权说明

对比特项目感兴趣，可以联系这个微信。



基础知识

微服务

微服务架构（通常简称为微服务）是指开发应用所用的一种架构形式。通过微服务，可将大型应用分解成多个独立的组件，其中每个组件都有各自的责任领域。在处理一个用户请求时，基于微服务的应用可能会调用许多内部微服务来共同生成其响应。

容器是微服务架构的绝佳示例，因为它们可让您专注于开发服务，而无需担心依赖项。现代云原生应用通常使用容器构建为微服务。

SpringBoot

Spring Boot 是 Spring 家族的成员，它是一个全新的框架，它的设计目的是尽可能简单和快速的开发、运行 Spring 应用程序，简化配置。它为开发者快捷的使用 Spring 及相关开发框架提供了便利，但是它并不是微服务的框架，它只是为微服务框架的使用也提供了很好的脚手架。

实战目的

掌握 Springboot 微服务如何制作成容器并在容器中启动。

实战步骤

编写 demo

使用 `Spring Boot` 创建一个简单的 `demo`， 在浏览器输出 `hello docker!`。

- 创建 maven 项目，选择 Spring Boot 2.x 的第一个发布版本 `2.0.2.RELEASE` 进行课程讲解, 如下在 pom.xml 中添加 spring-boot 依赖

```XML
<?xml version="1.0" encoding="UTF-8"?>
<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>org.example</groupId>
    <artifactId>springboot-demo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
```

```
        <maven.compiler.target>8</maven.compiler.target>
    </properties>

    <!-- 项目的 pom 定义继承自 SpringBoot 的父 pom -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
    </parent>

    <!-- Web 项目，添加 spring-boot-starter-web 依赖即可，版本号由父
pom 已经定义好了，此处省略 -->
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <!-- 添加 spring boot 项目构建插件 -->
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

- 编写主函数构建 Spring 容器

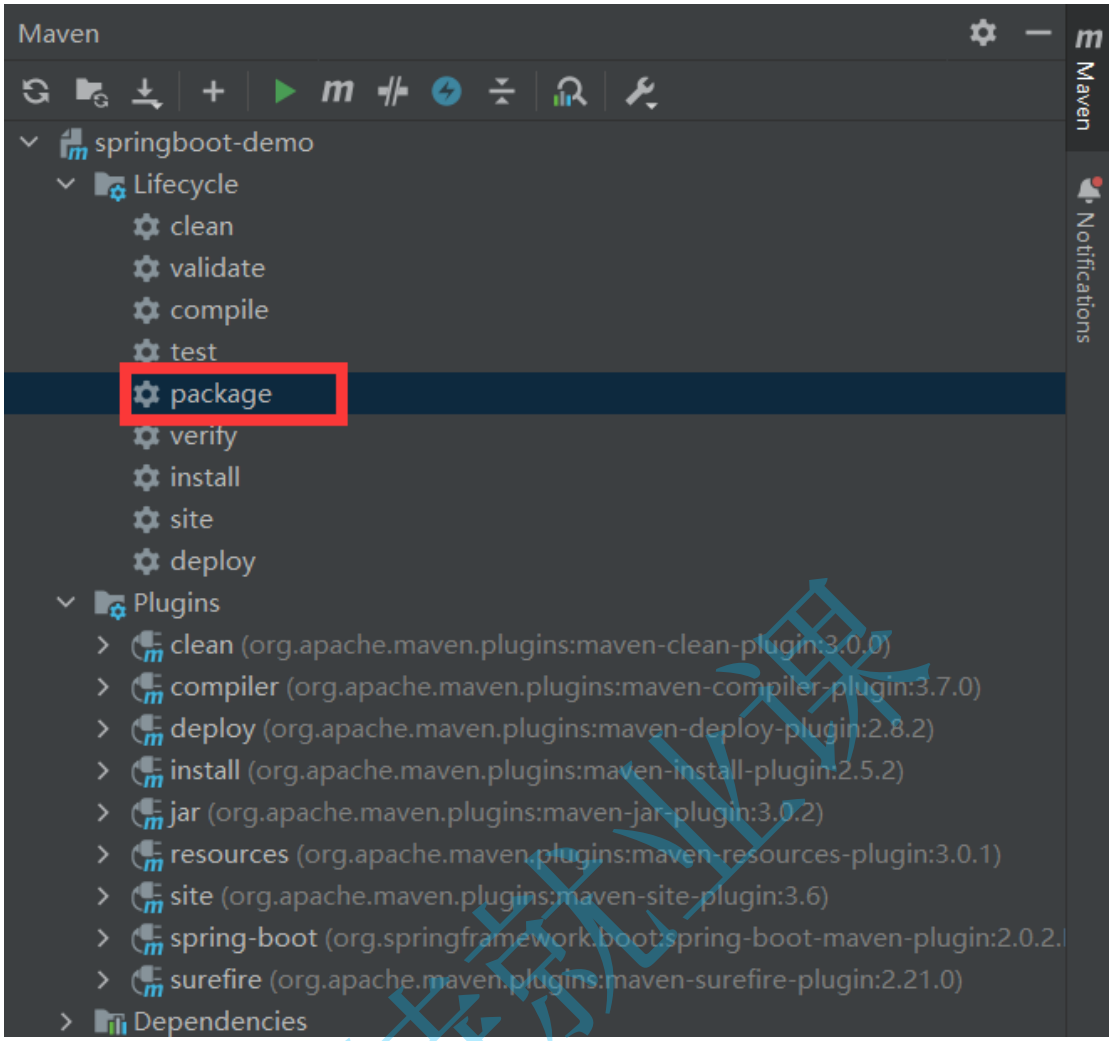
```
Java
package com.bittech.boot;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
public class ExampleApplication {
    public static void main(String[] args) {
        SpringApplication.run(ExampleApplication.class, args);
    }
}
```

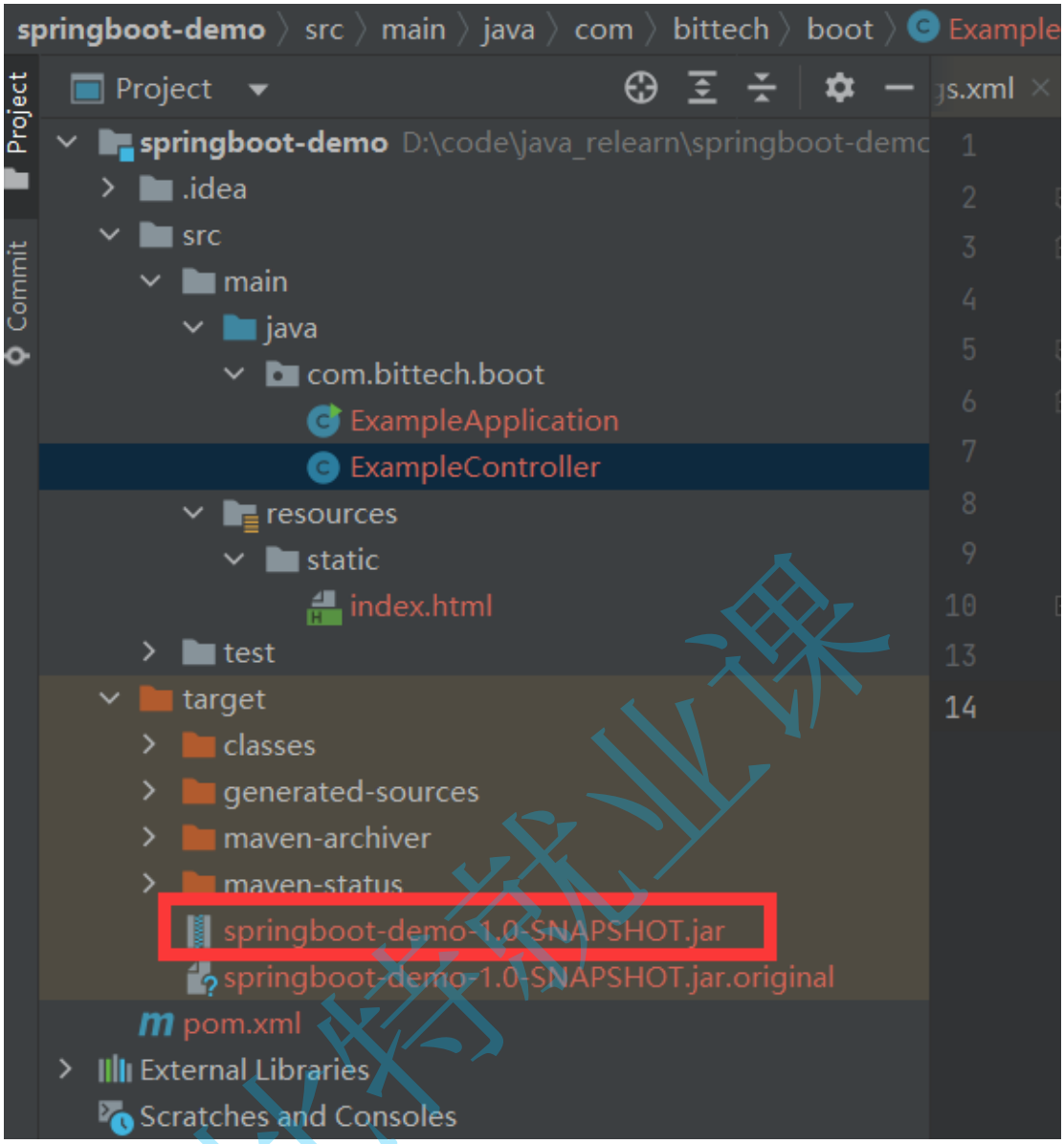
- 编写控制器

```
Java
package com.bittech.boot;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/hello")
public class ExampleController {

    @RequestMapping(value = "")
    public String index() {
        return "Hello Spring Boot for Dokcer World!";
    }
}
```

- 使用 maven 打包生成 jar 包



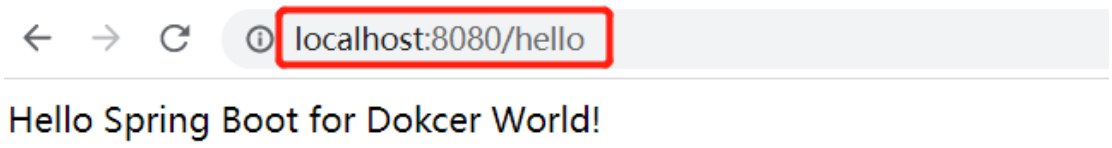
生成的 jar 包位于项目根路径下的 target 目录



- 测试 jar 包是否可用

```shell
# 在 cmd 中执行命令测试 jar 包是否可用
java -jar D:\code\java_relearn\springboot-demo\target\springboot-demo-1.0-SNAPSHOT.jar
```

启动容器后， 访问 `localhost:8080/hello` 查看运行结果， 确认 jar 包可用。



- 上传到 Linux 服务器备用

```
Shell
[zsc@VM-8-12-centos spring_demo]$ pwd
/home/zsc/dockerfile/spring_demo
[zsc@VM-8-12-centos spring_demo]$ ls
Dockerfile  springboot-demo-1.0-SNAPSHOT.jar
```

编写 Dockerfile 并构建生成镜像

- 编写 dockerfile

```
Dockerfile
# 指定 java 环境基础镜像
FROM openjdk:8
# 指定作者， 在公司内编写的时候最好写上 方便维护
MAINTAINER bitejiuyeke
# 拷贝 jar 包到容器中
ADD ./springboot-demo-1.0-SNAPSHOT.jar /app.jar
# 运行 jar 包
CMD ["java", "-jar", "/app.jar"]
```

- 构建生成镜像

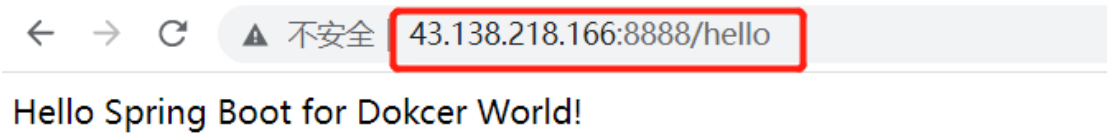
```
Shell
# 构建生成镜像 springboot-app
[zsc@VM-8-12-centos spring_demo]$ docker image build -t
springboot-app .
Sending build context to Docker daemon  16.14MB
Step 1/4 : FROM openjdk:8
8: Pulling from library/openjdk
001c52e26ad5: Pull complete
d9d4b9b6e964: Pull complete
2068746827ec: Pull complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
52a8c426d30b: Pull complete
8754a66e0050: Pull complete
Digest:
sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f54
52cb8
Status: Downloaded newer image for openjdk:8
 ---> b273004037cc
Step 2/4 : MAINTAINER bitejiuyeke
```

```
 ---> Running in f109b9bc266f
Removing intermediate container f109b9bc266f
 ---> 609966edeb24
Step 3/4 : ADD ./springboot-demo-1.0-SNAPSHOT.jar /app.jar
 ---> ce807632907c
Step 4/4 : CMD ["java", "-jar", "app.jar"]
 ---> Running in 1d81733ebd09
Removing intermediate container 1d81733ebd09
 ---> c5439c8549a9
Successfully built c5439c8549a9
Successfully tagged springboot-app:latest

# 查看镜像列表
[zsc@VM-8-12-centos spring_demo]$ docker image ls
REPOSITORY       TAG                 IMAGE ID       CREATED
SIZE
springboot-app   latest              c5439c8549a9   3 minutes ago
542MB
```

创建并运行容器

```
Shell
# 创建并运行容器，映射到宿主机的 8888 端口
[zsc@VM-8-12-centos spring_demo]$ docker container run -itd -p
8888:8080 springboot-app
aaa24b60d4dae1717acaec036a1b4064587428767b737be047ec8faa2eadb420
# 查看当前运行容器列表
[zsc@VM-8-12-centos spring_demo]$ docker container ls
CONTAINER ID   IMAGE           COMMAND               CREATED
STATUS             PORTS
NAMES
aaa24b60d4da   springboot-app   "java -jar app.jar"   About a
minute ago   Up About a minute   0.0.0.0:8888->8080/tcp, :::8888-
>8080/tcp   eager_swartz
```

访问 Linux 服务器的 `8888` 端口查看运行结果，可以看到通过浏览器已经可以访问到输出的日志了。



Hello Spring Boot for Dokcer World!

至此， `Spring Boot` 项目通过 Docker 容器部署完成。