理解独立的Spring应用

SpringBoot它内置了web容器并且可以直接生成可执行的jar包,这里主要讨论可执行jar包中的内容

创建SpringBoot应用可知行JAR

SpringBoot之所以可以使用Maven打包生成可执行JAR包是因为他其中包含了maven的工具插件

```
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
```

他会帮我们完成打包,这里对其执行的过程以及原理不做过多的赘述,接下来对打包后生成的目录结构进行分析

```
.
├── BOOT-INF
|   ├── classes
|   |   ├── application.properties
|   |   ├── thinking
|   |   |   └── in
|   |   |       └── spring
|   |   |           └── boot
|   |   |               ├── autoconfigure
|   |   |               |   └── WebAutoConfiguration.class
|   |   |               ├── config
|   |   |               |   └── WebConfiguration.class
|   |   |               └── firstappbygui
|   |   |                   └── FirstAppByGuiApplication.class
|   |   └── uml
|   |       └── Launcher.ucls
|   └── lib [60 entries exceeds filelimit, not opening dir]
├── META-INF
|   ├── MANIFEST.MF
```

```
|   ├── maven
|   |   └── thinking-in-spring-boot
|   |       └── first-app-by-gui
|   |           ├── pom.properties
|   |           └── pom.xml
|   └── spring.factories
└── org
    └── springframework
        └── boot
            └── loader [15 entries exceeds filelimit, not opening dir]
```

- BOOT-INF/classes存放编译后的class文件;
- BOOT-INF/lib目录存放应用依赖的JAR包;
- META-INF/目录存放应用相关的元信息,如MANIFEST.MF文件;
- org/目录存放SpringBoot相关的class文件.

经过