

Spring Boot3 简介及快速搭建

Spring Boot 简介及快速搭建

1.简介

2.Why SpringBoot?

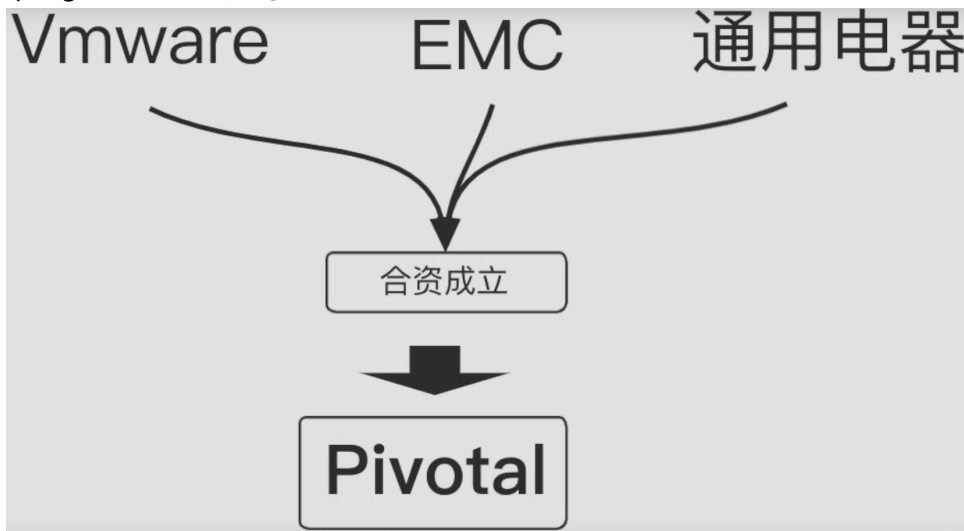
2.1微服务

3.快速开始 SpringBoot Hello World

4.代码说明

1.简介

SpringBoot它基于Spring4.0设计，是由 **Pivotal** 公司提供的框架。



2014年 Spring Boot

2015年 Spring Cloud

2018年 敲钟上市

2014 年 4 月发布 Spring Boot 1.0 基于Spring4.0

2018 年 3 月 Spring Boot 2.0发布 基于Spring 5.0。

SpringBoot 基于 Spring 开发。不仅继承了Spring框架原有的优秀特性，它并不是用来替代 Spring 的解决方案，而和 Spring 框架紧密结合进一步简化了Spring应用的整个搭建和开发过程。**其设计目的是用来简化 Spring 应用的初始搭建以及开发过程**怎么简化的呢？就是通过提供默认配置等方式让我们更容易使用。

关于 SpringBoot 有一句很出名的话就是**约定大于配置**。采用 Spring Boot 可以大大的简化开发模式，它集成了大量常用的第三方库配置，所有你想集成的常用框架，它都有对应的组件支持，例如 Redis、MongoDB、Dubbo、kafka，ES等等。SpringBoot 应

用中这些第三方库几乎可以零配置地开箱即用，大部分的 SpringBoot 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。另外SpringBoot通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

简化Spring应用开发的一个框架；

对整个企业级开发技术栈的一个大整合build anything；

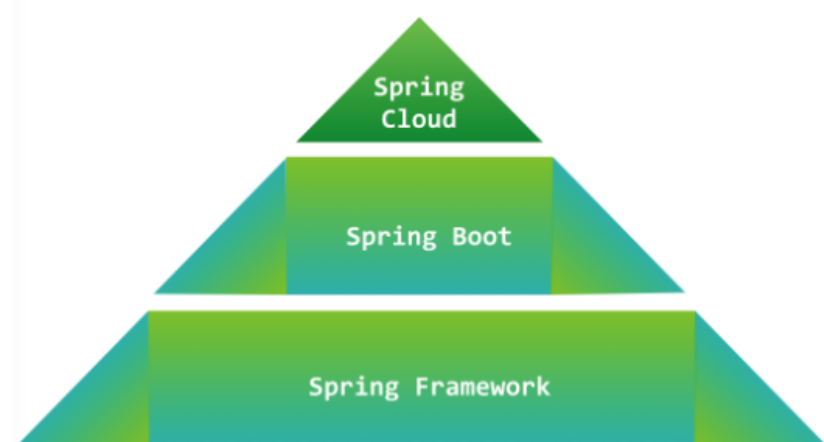
J2EE开发的一站式解决方案；

优点

- 快速构建一个独立的 Spring 应用程序；
- 嵌入的 Tomcat 、 Jetty 或者 Undertow，无须部署 WAR 文件；
- 提供starter POMs来简化Maven配置和减少版本冲突所带来的问题；
- 对Spring和第三方库提供默认配置，也可修改默认值，简化框架配置；
- 提供生产就绪型功能，如指标、健康检查和外部配置；
- 无需配置XML--JavaConfig，无代码生成，开箱即用；

2.Why SpringBoot?

刚才说 SpringBoot 简化了基于 Spring 开发，这只是最直观的一方面；还有一个方面：也更得力于各微服务组件的支持,这也是谈 SpringBoot 必谈微服务的原因。（起初是Netflix移植到Spring),可以说是Spring Cloud带动了SpringBoot，SpringBoot成就了SpringCloud。



SpringBoot和微服务的火热程度是同步的



2.1微服务

2014年一个叫 Martin Fowler（同时也是经典著作《重构：改善既有代码的设计》一书的作者）发表了一篇关于[微服务的博客](#)，比较形象生动地介绍了什么是微服务，然后微服务才慢慢被人所熟知。



他说微服务其实是一种架构风格，我们在开发一个应用的时候这个应用应该是由一组小型服务组成，每个小型服务都运行在自己的进程内；小服务之间通过HTTP的方式进行互联互通。

和微服务相对应的就是我们之前的，**单体应用**，就是大名鼎鼎的 all in one 的风格。这种风格把所有的东西都写在一个应用里面，比如我们熟悉的OA，CRM，ERP系统，所有的页面，所有的代码都放在一起，打成打成一个war包，然后把war包放在Tomcat容器中运行。

这种传统web开发的架构模式当然也有它的优势，比如它测试部署比较简单，因为不涉及到多个服务的互联互通，只需要把一个包上传到服务器就行了，可以说是一人吃饱全家不饿。同样也不会给运维带来麻烦，方便水平扩展，只需要又把相同的应用复制多份放在不同的服务器中就达到了扩展的目的。

单体应用的缺点也显而易见，容易牵一发而动全身，比如要更改一个小小的功能，就可能需要重新部署整个应用。当然，更大的挑战就是日益增长的用户需求。

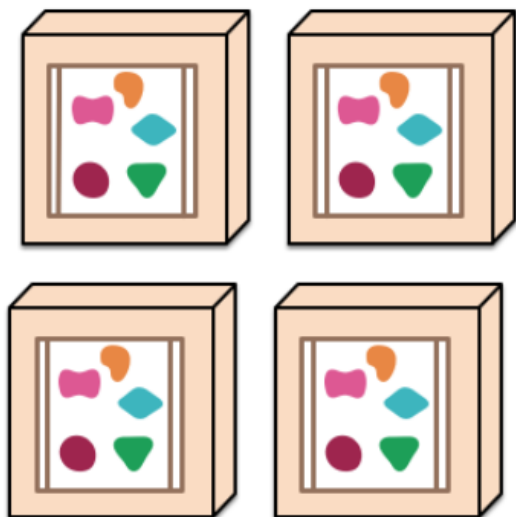
一个单体应用程序把它所有的功能放在一个单一进程中...



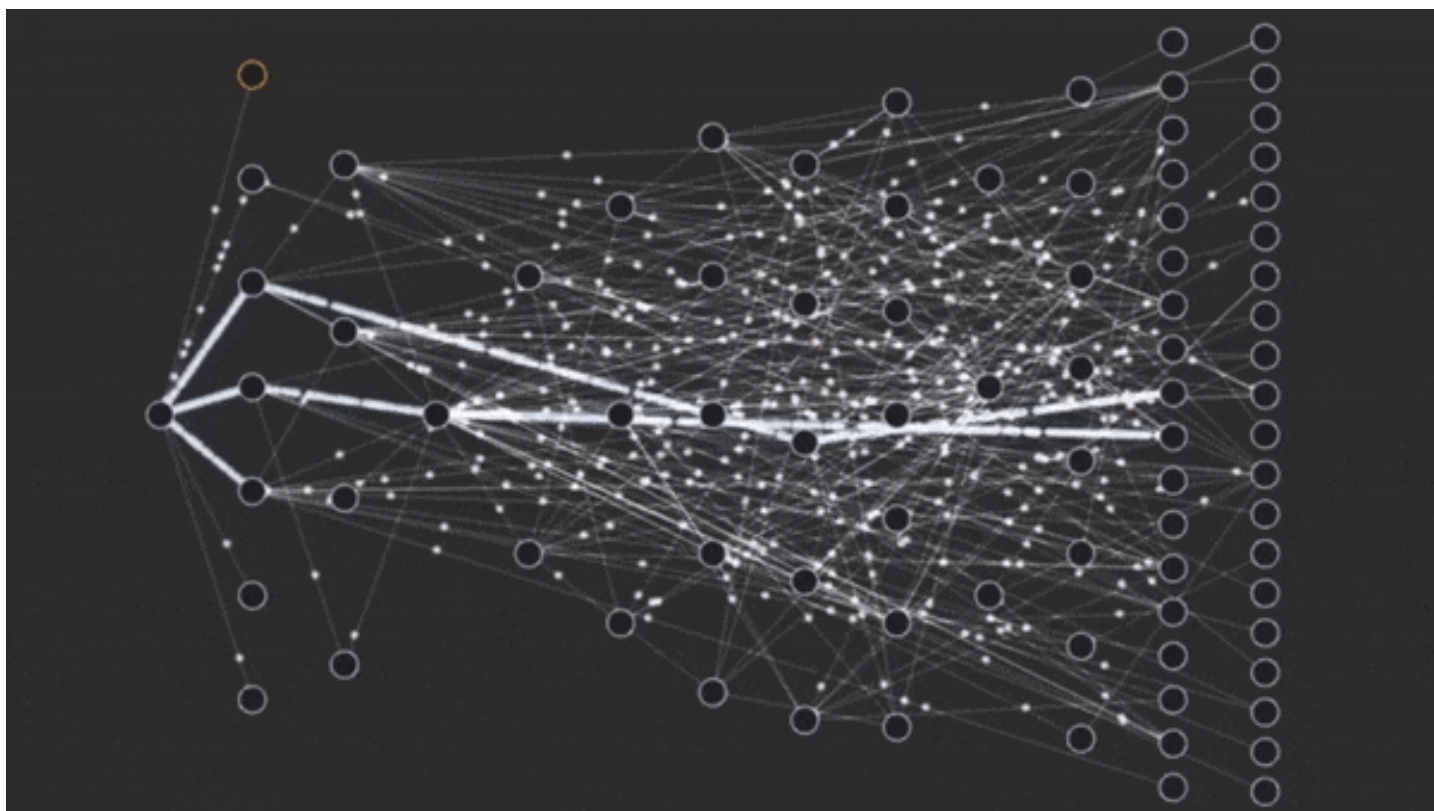
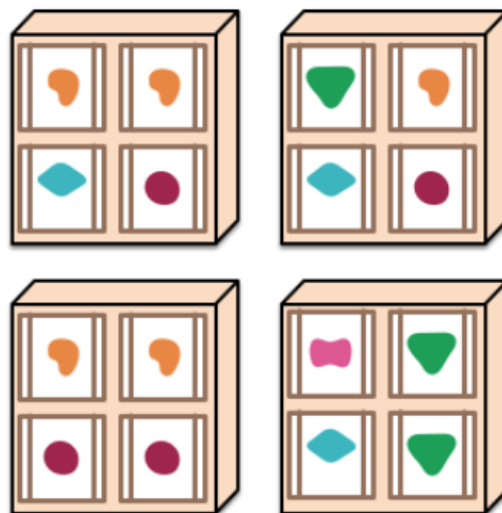
一个微服务架构把每个功能元素放进一个独立的服务中...



...并且通过在多个服务器上复制这个单体进行扩展



...并且通过跨服务器分发这些服务进行扩展，只在需要时才复制。



3.快速开始 SpringBoot Hello World

前置知识要求:

- SSM框架的使用经验
- 熟练使用Maven进行项目构建和依赖管理
- 熟练使用Idea

环境要求

Spring Boot 3.1.5需要[Java 17](#), 还需要[Spring Framework 6.x](#)版本。

构建工具的支持

| Build Tool | Version |
|------------|----------------------------|
| Maven | 3.6.3 or later |
| Gradle | 7.x (7.5 or later) and 8.x |

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!--
4  Licensed to the Apache Software Foundation (ASF) under one
5  or more contributor license agreements. See the NOTICE file
6  distributed with this work for additional information
7  regarding copyright ownership. The ASF licenses this file
8  to you under the Apache License, Version 2.0 (the
9  "License"); you may not use this file except in compliance
10 with the License. You may obtain a copy of the License at
11
12     http://www.apache.org/licenses/LICENSE-2.0
13
14 Unless required by applicable law or agreed to in writing,
15 software distributed under the License is distributed on an
16 "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
17 KIND, either express or implied. See the License for the
18 specific language governing permissions and limitations
19 under the License.
20 -->
21
22 <!--
23 | This is the configuration file for Maven. It can be specified at two levels:
24 |
25 | 1. User Level. This settings.xml file provides configuration for a single user,
26 |    and is normally provided in ${user.home}/.m2/settings.xml.
27 |
28 |    NOTE: This location can be overridden with the CLI option:
29 |
30 |        -s /path/to/user/settings.xml
31 |
32 | 2. Global Level. This settings.xml file provides configuration for all Maven
```

```

33 |         users on a machine (assuming they're all using the same Maven
34 |         installation). It's normally provided in
35 |         ${maven.conf}/settings.xml.
36 |
37 |         NOTE: This location can be overridden with the CLI option:
38 |
39 |         -gs /path/to/global/settings.xml
40 |
41 | The sections in this sample file are intended to give you a running start at
42 | getting the most out of your Maven installation. Where appropriate, the default
43 | values (values used when the setting is not specified) are provided.
44 |
45 | -->
46 <settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"
47     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
48     xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0 https://maven.apache.org/xsd/settings-1.2.0.xsd"
49 <!-- localRepository
50 | The path to the local repository maven will use to store artifacts.
51 |
52 | Default: ${user.home}/.m2/repository
53 <localRepository>/path/to/local/repo</localRepository>
54 -->
55 <localRepository>C:\repmvn</localRepository>
56
57 <!-- interactiveMode
58 | This will determine whether maven prompts you when it needs input. If set to false,
59 | maven will use a sensible default value, perhaps based on some other setting, for
60 | the parameter in question.
61 |
62 | Default: true
63 <interactiveMode>true</interactiveMode>
64 -->
65
66 <!-- offline
67 | Determines whether maven should attempt to connect to the network when executing a build.
68 | This will have an effect on artifact downloads, artifact deployment, and others.
69 |
70 | Default: false
71 <offline>false</offline>
72 -->
73
74 <!-- pluginGroups
75 | This is a list of additional group identifiers that will be searched when resolving plugins by their prefix, i.e.
76 | when invoking a command line like "mvn prefix:goal". Maven will automatically add the group identifiers
77 | "org.apache.maven.plugins" and "org.codehaus.mojo" if these are not already contained in the list.
78 | -->
79 <pluginGroups>
80     <!-- pluginGroup
81     | Specifies a further group identifier to use for plugin lookup.
82     <pluginGroup>com.your.plugins</pluginGroup>
83     -->

```

```

84 </pluginGroups>
85
86 <!-- TODO Since when can proxies be selected as depicted? -->
87 <!-- proxies
88 | This is a list of proxies which can be used on this machine to connect to the network.
89 | Unless otherwise specified (by system property or command-line switch), the first proxy
90 | specification in this list marked as active will be used.
91 |-->
92 <proxies>
93   <!-- proxy
94   | Specification for one proxy, to be used in connecting to the network.
95   |
96   <proxy>
97     <id>optional</id>
98     <active>true</active>
99     <protocol>http</protocol>
100    <username>proxyuser</username>
101    <password>proxypass</password>
102    <host>proxy.host.net</host>
103    <port>80</port>
104    <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
105  </proxy>
106  -->
107 </proxies>
108
109 <!-- servers
110 | This is a list of authentication profiles, keyed by the server-id used within the system.
111 | Authentication profiles can be used whenever maven must make a connection to a remote server.
112 |-->
113 <servers>
114   <!-- server
115   | Specifies the authentication information to use when connecting to a particular server, identified by
116   | a unique name within the system (referred to by the 'id' attribute below).
117   |
118   | NOTE: You should either specify username/password OR privateKey/passphrase, since these pairings are
119   |       used together.
120   |
121   <server>
122     <id>deploymentRepo</id>
123     <username>repouser</username>
124     <password>repopwd</password>
125   </server>
126   -->
127
128   <!-- Another sample, using keys to authenticate.
129   <server>
130     <id>siteServer</id>
131     <privateKey>/path/to/private/key</privateKey>
132     <passphrase>optional; leave empty if not used.</passphrase>
133   </server>
134   -->

```

```
135 </servers>
136
137 <!-- mirrors
138 | This is a list of mirrors to be used in downloading artifacts from remote repositories.
139 |
140 | It works like this: a POM may declare a repository to use in resolving certain artifacts.
141 | However, this repository may have problems with heavy traffic at times, so people have mirrored
142 | it to several places.
143 |
144 | That repository definition will have a unique id, so we can create a mirror reference for that
145 | repository, to be used as an alternate download site. The mirror site will be the preferred
146 | server for that repository.
147 |-->
148 <mirrors>
149 <!-- mirror
150 | Specifies a repository mirror site to use instead of a given repository. The repository that
151 | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are used
152 | for inheritance and direct lookup purposes, and must be unique across the set of mirrors.
153 |
154 <mirror>
155 <id>mirrorId</id>
156 <mirrorOf>repositoryId</mirrorOf>
157 <name>Human Readable Name for this Mirror.</name>
158 <url>http://my.repository.com/repo/path</url>
159 </mirror>
160 -->
161
162 <mirror>
163 <id>huaweicloud</id>
164 <mirrorOf>*</mirrorOf>
165 <url>https://repo.huaweicloud.com/repository/maven/</url>
166 </mirror>
167 <mirror>
168 <id>nexus-aliyun</id>
169 <name>Nexus aliyun</name>
170 <url>http://maven.aliyun.com/nexus/content/groups/public</url>
171 <mirrorOf>central</mirrorOf>
172 </mirror>
173 <mirror>
174 <id>central</id>
175 <name>Maven Repository Switchboard</name>
176 <url>http://repo.maven.org/maven2/</url>
177 <mirrorOf>central</mirrorOf>
178 </mirror>
179 <mirror>
180 <id>central</id>
181 <name>Maven Repository Switchboard</name>
182 <url>http://repo2.maven.org/maven2/</url>
183 <mirrorOf>central</mirrorOf>
184 </mirror>
185 </mirrors>
```



```

186
187 <!-- profiles
188 | This is a list of profiles which can be activated in a variety of ways, and which can modify
189 | the build process. Profiles provided in the settings.xml are intended to provide local machine-
190 | specific paths and repository locations which allow the build to work in the local environment.
191 |
192 | For example, if you have an integration testing plugin - like cactus - that needs to know where
193 | your Tomcat instance is installed, you can provide a variable here such that the variable is
194 | dereferenced during the build process to configure the cactus plugin.
195 |
196 | As noted above, profiles can be activated in a variety of ways. One way - the activeProfiles
197 | section of this document (settings.xml) - will be discussed later. Another way essentially
198 | relies on the detection of a property, either matching a particular value for the property,
199 | or merely testing its existence. Profiles can also be activated by JDK version prefix, where a
200 | value of '1.4' might activate a profile when the build is executed on a JDK version of '1.4.2_07'.
201 | Finally, the list of active profiles can be specified directly from the command line.
202 |
203 | NOTE: For profiles defined in the settings.xml, you are restricted to specifying only artifact
204 | repositories, plugin repositories, and free-form properties to be used as configuration
205 | variables for plugins in the POM.
206 |
207 |-->
208 <profiles>
209 <!-- profile
210 | Specifies a set of introductions to the build process, to be activated using one or more of the
211 | mechanisms described above. For inheritance purposes, and to activate profiles via <activatedProfiles/>
212 | or the command line, profiles have to have an ID that is unique.
213 |
214 | An encouraged best practice for profile identification is to use a consistent naming convention
215 | for profiles, such as 'env-dev', 'env-test', 'env-production', 'user-jdcasey', 'user-brett', etc.
216 | This will make it more intuitive to understand what the set of introduced profiles is attempting
217 | to accomplish, particularly when you only have a list of profile id's for debug.
218 |
219 | This profile example uses the JDK version to trigger activation, and provides a JDK-specific repo.
220 <profile>
221 <id>jdk-1.4</id>
222
223 <activation>
224 <jdk>1.4</jdk>
225 </activation>
226
227 <repositories>
228 <repository>
229 <id>jdk14</id>
230 <name>Repository for JDK 1.4 builds</name>
231 <url>http://www.myhost.com/maven/jdk14</url>
232 <layout>default</layout>
233 <snapshotPolicy>always</snapshotPolicy>
234 </repository>
235 </repositories>
236 </profile>

```

```

237     -->
238
239     <!--
240     | Here is another profile, activated by the property 'target-env' with a value of 'dev', which
241     | provides a specific path to the Tomcat instance. To use this, your plugin configuration might
242     | hypothetically look like:
243     |
244     | ...
245     | <plugin>
246     |   <groupId>org.myco.myplugins</groupId>
247     |   <artifactId>myplugin</artifactId>
248     |
249     |   <configuration>
250     |     <tomcatLocation>${tomcatPath}</tomcatLocation>
251     |   </configuration>
252     | </plugin>
253     | ...
254     |
255     | NOTE: If you just wanted to inject this configuration whenever someone set 'target-env' to
256     |       anything, you could just leave off the <value/> inside the activation-property.
257     |
258     <profile>
259       <id>env-dev</id>
260
261       <activation>
262         <property>
263           <name>target-env</name>
264           <value>dev</value>
265         </property>
266       </activation>
267
268       <properties>
269         <tomcatPath>path/to/tomcat/instance</tomcatPath>
270       </properties>
271     </profile>
272     -->
273 </profiles>
274
275 <!-- activeProfiles
276 | List of profiles that are active for all builds.
277 |
278 <activeProfiles>
279   <activeProfile>alwaysActiveProfile</activeProfile>
280   <activeProfile>anotherAlwaysActiveProfile</activeProfile>
281 </activeProfiles>
282 -->
283 </settings>
284

```

3.1 创建一个maven工程; (jar)

3.2 项目中引入依赖

```
1  <!--继承springboot的父项目-->
2  <parent>
3      <groupId>org.springframework.boot</groupId>
4      <artifactId>spring-boot-starter-parent</artifactId>
5      <version>3.1.5</version>
6      <relativePath/> <!-- lookup parent from repository -->
7  </parent>
8
9  <dependencies>
10     <!--引入springboot的web支持-->
11     <dependency>
12         <groupId>org.springframework.boot</groupId>
13         <artifactId>spring-boot-starter-web</artifactId>
14     </dependency>
15 </dependencies>
```

3.3 建包并创建控制器

```
1  //在项目中创建指定的包结构
2  /*
3  cn
4      +| tulingxueyuan
5          +| controller */
6  @Controller
7  @RequestMapping("/hello")
8  public class HelloController {
9      @RequestMapping("/world")
10     @ResponseBody
11     public String hello(){
12         System.out.println("====hello world====");
13         return "hello";
14     }
15 }
16
```

3.4 编写启动类

```
1  //在项目中如下的包结构中创建启动类 Application
2  /*
3  cn
4      +| tulingxueyuan
5          +| Application.java */
6      @SpringBootApplication
7      public class Application {
8          public static void main(String[] args) {
9              SpringApplication.run(Application.class,args);
10          }
11      }
```

3.6 运行main启动项目

```

  _ _ _ _ _
 / \ / \ _ _ _ _ _ \ \ \ \
 ( ( \ \ | ' _ | ' _ | ' _ \ \ | \ \ \ \
 \ \ _ \ | \ | | | | | | \ | | ) ) ) )
 ' | _ \ | _ | | | | | \ \ | / / / /
 =====|_|=====|_|_/_/_/_/

:: Spring Boot ::      (v2. 2. 7. RELEASE)

2020-11-04 22:33:37.003 INFO 30180 --- [main] com.tuling.SpringbootTestApplication : Starting SpringbootTestApplication
2020-11-04 22:33:37.008 INFO 30180 --- [main] com.tuling.SpringbootTestApplication : No active profile set, falling back to default profiles
2020-11-04 22:33:38.122 INFO 30180 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080
2020-11-04 22:33:38.137 INFO 30180 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-11-04 22:33:38.137 INFO 30180 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.27]
2020-11-04 22:33:38.139 INFO 30180 --- [main] o.a.catalina.core.AprLifecycleListener : An older version [1.2.16] of the APR library is
2020-11-04 22:33:38.139 INFO 30180 --- [main] o.a.catalina.core.AprLifecycleListener : Loaded APR based Apache Tomcat Native support
2020-11-04 22:33:38.139 INFO 30180 --- [main] o.a.catalina.core.AprLifecycleListener : APR capabilities: IPv6 [true], sendfile [true],
2020-11-04 22:33:38.140 INFO 30180 --- [main] o.a.catalina.core.AprLifecycleListener : APR/OpenSSL configuration: useAprConnector [false]
2020-11-04 22:33:39.364 INFO 30180 --- [main] o.a.catalina.core.AprLifecycleListener : OpenSSL successfully initialized [OpenSSL 1.1.1f]
2020-11-04 22:33:39.573 INFO 30180 --- [main] o.a.c.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-11-04 22:33:39.573 INFO 30180 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed
2020-11-04 22:33:39.891 INFO 30180 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'application'
2020-11-04 22:33:40.079 INFO 30180 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base URL [http://localhost:8080]
2020-11-04 22:33:40.140 INFO 30180 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http://localhost:8080)
2020-11-04 22:33:40.143 INFO 30180 --- [main] com.tuling.SpringbootTestApplication : Started SpringbootTestApplication
2020-11-04 22:33:41.351 INFO 30180 --- [3]-192.168.56.1 o.a.c.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-11-04 22:33:41.351 INFO 30180 --- [3]-192.168.56.1 o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2020-11-04 22:33:41.356 INFO 30180 --- [3]-192.168.56.1 o.s.web.servlet.DispatcherServlet : Completed initialization in 5 ms

```

出现以上日志说明启动成功

3.7 访问项目

注意：springboot的项目默认没有项目名

访问路径: `http://localhost:8080/hello/world`

3.7 修改端口

如果出现Web server failed to start. Port 8080 was already in use.说明端口正在使用，我们需要修改默认端口

项目中src/main/resources/application.properties

```
1 server.port=8088
```

访问路径: `http://localhost:8088/hello/world`

3.8 部署服务器

```
1 <!-- 这个插件，可以将应用打包成一个可执行的jar包，-->
2 <build>
3   <plugins>
4     <plugin>
5       <groupId>org.springframework.boot</groupId>
6       <artifactId>spring-boot-maven-plugin</artifactId>
```

```

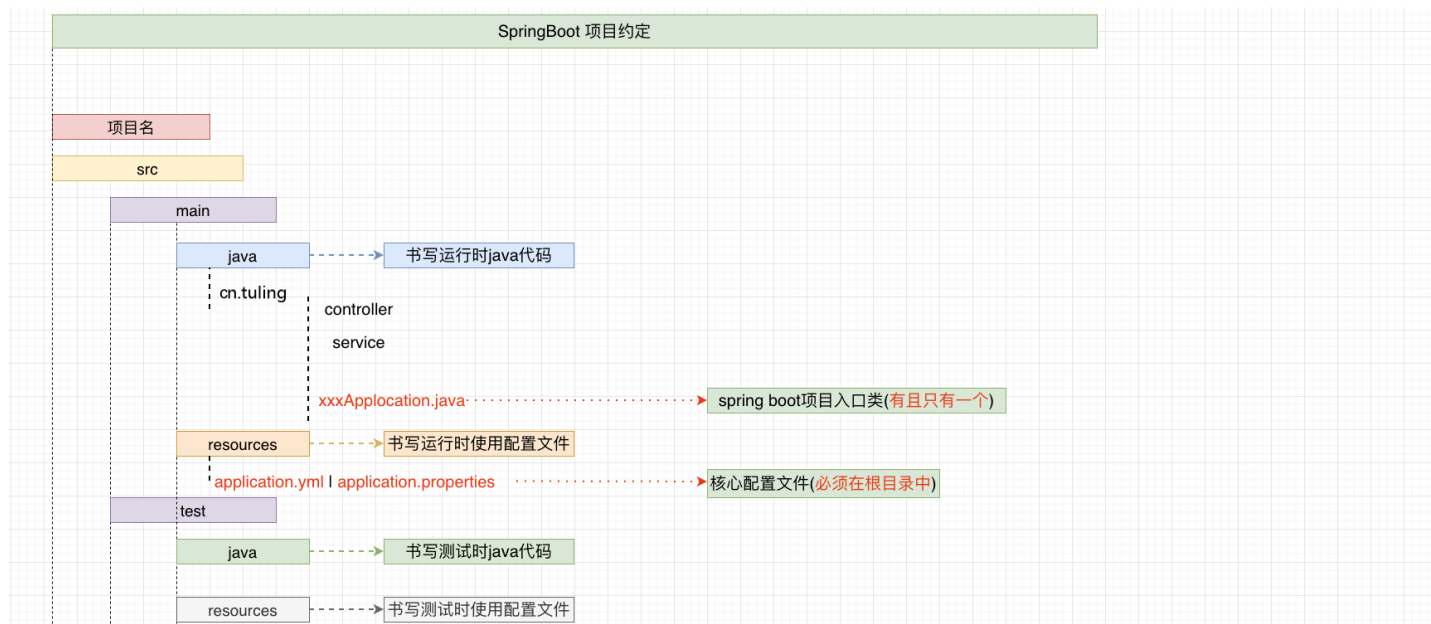
7     </plugin>
8 </plugins>
9 </build>

```

将这个应用打成jar包，直接使用java -jar的命令进行执行；

4.代码说明

文件说明



pom.xml

• spring-boot-starter-parent

```

1 <!--这是Spring Boot的父级依赖，这样当前的项目就是Spring Boot项目了。
2 它用来提供相关的Maven默认依赖。使用它之后，常用的包依赖可以省去version标签。-->
3 <parent>
4     <groupId>org.springframework.boot</groupId>
5     <artifactId>spring-boot-starter-parent</artifactId>
6     <version>3.1.5</version>
7     <relativePath/> <!-- lookup parent from repository -->
8 </parent>

```

```

1 <!--
2 spring-boot-starter-parent的父maven
3 它管理了Spring Boot应用里面的所有依赖版本；
4 -->
5 <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-dependencies</artifactId>
8     <version>3.1.5</version>
9 </parent>

```

- **spring-boot-starter-web**

spring-boot-starter: spring-boot场景启动器；帮我们导入了web模块正常运行所依赖的组件；

```
1 <dependencies>
2   <dependency>
3     <groupId>org.springframework.boot</groupId>
4     <artifactId>spring-boot-starter-web</artifactId>
5   </dependency>
6 </dependencies>
```

Spring Boot将所有的功能场景都抽取出来，做成一个个的starters（启动器），只需要在项目里面引入这些starter相关场景的所有依赖都会导入进来。要用什么功能就导入什么场景的启动器

- **Application 启动类**

```
1 /**
2  * @SpringBootApplication 来标注一个主程序类，说明这是一个Spring Boot应用
3  * 自动装配就是从这里开始的
4  */
5 @SpringBootApplication
6 public class Application {
7
8     public static void main(String[] args) {
9
10         // Spring应用启动起来
11         SpringApplication.run(Application.class,args);
12     }
13 }
```

描述一下SpringBoot的作用

SpringBoot有哪些特性？