# SpringBoot | 番外:使用小技巧合集

謝謝同学 ImportNew 今天

(点击上方公众号,可快速关注)

来源: oKong,

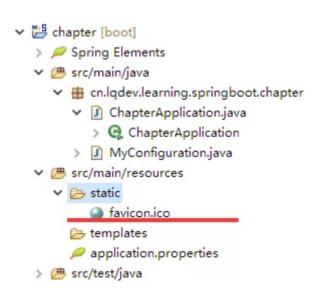
blog.lqdev.cn/2018/08/11/springboot/springboot-tips/

#### 前言

最近工作比较忙,事情也比较多。加班回到家都十点多了,洗个澡就想睡觉了。所以为了不断更太多天,偷懒写个小技巧合集吧。之后有时间都会进行文章更新的。原创不易,码 字不易,还希望大家多多支持!话不多说,开始今天的技巧合集吧~

#### 设置网站图标

原来我们在使用tomcat开发时,设置网站图片时,即icon图标时,一般都是直接替换root包下的favicon.ico替换成自己的,或者在网页的头部设置link的ref为icon然后设置其href值。而在SpringBoot中,替换图片也是很简单的,只需要将自定义图片放置在静态资源目录下即可,即默认有static、public、resources、/META-INF/resources或者自定义的静态目录下即可。





# Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Aug 11 10:20:16 CST 2018
There was an unexpected error (type=Not Found, status=404).
No message available

#### 允许跨域访问

CORS是一个W3C标准,全称是"跨域资源共享"(Cross-origin resource sharing)。它允许浏览器向跨源(协议 + 域名 + 端口)服务器,发出XMLHttpRequest请求,从而克服了AJAX只能同源使用的限制。

简单来说,跨域问题是可以通过nginx来解决的,或者通过jsonp(只支持get请求)来解决。 而SpringBoot中也提供了配置方法。

**0.**利用@CrossOrigin注解,可放至在类上或者方法上。类上代表整个控制层所有的映射方法都支持跨域请求。

```
@CrossOrigin(origins = "http://blog.lqdev.cn", maxAge = 3600)
@RestController
public class demoController{

@GetMapper("/")
public String index(){
   return "hello,CORS";
}
```

1.配置全局CORS配置。官网也有给出实例,具体如下:

```
@Configuration
public class MyConfiguration {
    @Bean
```

# 独立Tomcat运行

讲解了这么久,一般上我们都是通过jar包的方式进行启动的应用的。所以部署在独立的 tomcat时,需要如何解决呢?其实也简单,只需要将项目打包方式修改为war包,然后修 改下启动类配置即可。

0.修改pom打包方式为war, 同时排除了内置的tomcat。

1.改造下启动类,使其继承SpringBootServletInitializer,同时覆盖configure方法。

```
@SpringBootApplication
@SIf4j
public class ChapterApplication extends SpringBootServletInitializer{
```

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

// new SpringApplicationBuilder().sources(ChapterApplication.class).web(false).run(args);

//之后这里设置业务逻辑 比如挂起一个线程 或者设置一个定时任务。保证不退出

//不然它就是一个启动类,启动后就停止了。
    log.info("jar,chapter启动!");

}

@Override
protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
    log.info("外部tomcat,chapter启动!");
    return application.sources(ChapterApplication.class);
}
```

2.maven打包成war(mvn clean install),然后放入tomcat中,启动运行即可。

```
≜ Tomcat
11-Aug-2018 10:28:08.703 信息 [main] org. apache. tomcat. util. net. NioSelectorPool. ^
getSharedSelector Using a shared selector for servlet write/read
11-Aug-2018 10:28:08.713 信息 [main] org. apache. coyote. AbstractProtocol. init Ini tializing ProtocolHandler ["ajp-nio-8009"] 11-Aug-2018 10:28:08.716 信息 [main] org. apache. tomcat. util. net. NioSelectorPool. getSharedSelector Using a shared selector for servlet write/read 11-Aug-2018 10:28:08.717 信息 [main] org. apache. catalina. startup. Catalina. load I
nitialization processed in 1187 ms
11-Aug-2018 10:28:08.758 信息 [main] org. apache. catalina. core. StandardService. st
artInternal Starting service [Catalina]
11-Aug-2018 10:28:08.759 信息 [main] org. apache. catalina. core. StandardEngine. sta
rtInternal Starting Servlet Engine: Apache Tomcat/8.5.27
11-Aug-2018 10:28:08.786 信息 [localhost-startStop-1] org. apache. catalina. startu
p. HostConfig. deployWAR Deploying web application archive [D:\apache-tomcat-8.5.2
7\webapps\chapter-0.0.1-SNAPSHOT.war]
11-Aug-2018 10:28:11.693 信息[localhost-startStop-1]org.apache.jasper.servlet.
TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no TLDs.
 Enable debug logging for this logger for a complete list of JARs that were scan
ned but no TLDs were found in them. Skipping unneeded JARs during scanning can i
mprove startup time and JSP compilation time.
10:28:12.363 [localhost-startStop-1] INFO cn. lqdev. learning. springboot. chapter. C
hapterApplication - 外部tomcat,chapter启动!
```

其实这样设置的话, 在开发时直接运行启动类也还是可以直接运行的, 方便.

```
: Mapping filter: 'httpPutFormContentFilter' to: [/*]
.b.w.servlet.FilterRegistrationBean
.b.w.servlet.FilterRegistrationBean
                                     : Mapping filter: 'requestContextFilter' to: [/*]
.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.emb
s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEn
.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframe
.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.spring
.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type [class org.springframewo:
.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.sp
.j.e.a.AnnotationMBeanExporter
                                     : Registering beans for JMX exposure on startup
.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
.1.s.chapter.ChapterApplication
                                     : Started ChapterApplication in 3.153 seconds (JVM running for 4.312)
                                     : jar,chapter启动!
.l.s.chapter.ChapterApplication
```

#### 启动不设置端口

对一些定时任务服务项目,其本身只是提供一个定时调度功能,不需要其他服务调用,只是去调度其他服务。像这样的服务,正常也就不需要设置端口了。这时候SpringBoot也是支持的。只需要改下启动方式:

```
new SpringApplicationBuilder().sources(ChapterApplication.class).web(false).run(args);
//之后这里设置业务逻辑 比如挂起一个线程 或者设置一个定时任务。保证不退出
//不然它就是一个启动类,启动后就停止了。
```

#### 或者修改配置文件的属性:

spring.main.web-environment=false

## 最后效果,是不是没有看见端口了:

```
ition : Starting ChapterApplication on Kong-pc with PID 6840 (H:\okongWorksplace\spring-boot\chapter\t
ition : No active profile set, falling back to default profiles: default
ionContext : Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@28c47llc:
iter : Registering beans for JMX exposure on startup
ition : Started ChapterApplication in 1.043 seconds (JVM running for 1.975)
ition : chapter启动!
ionContext : Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@28c47llc: st
:ter : Unregistering JMX-exposed beans on shutdown
```

# 启动完成前进行业务逻辑

利用CommandLineRunner或者ApplicationRunner可实现在SpringApplication的run() 完成前执行一些业务逻辑

0.修改启动类,实现CommandLineRunner接口,ApplicationRunner类似,只是run的入参不同而已。

#### @Override

public void run(String... args) throws Exception {

```
log.info("CommandLineRunner运行");
}
```

1.运行应用、注意查看控制台输出:

```
main] o.s.w.s.nander.Simpleorinandernapping : napped okt path [/~/lavicon.ico] onto nander of type [class org.springframework.web.servicentain] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
main] c.l.l.s.chapter.ChapterApplication : CommandLineRunner)运行
main] c.l.l.s.chapter.ChapterApplication : Started ChapterApplication in 2.806 seconds (JVM running for 3.87)
main] c.l.l.s.chapter.ChapterApplication : jar,chapter启动完成:
```

当然,直接申明一个bean也是可以的。

若多个时,可设置@Order来确定执行的顺序。

#### 动态修改日志级别

通过org.springframework.boot.logging.LoggingSystem提供的api即可。

loggingSystem.setLogLevel(null, LogLevel.DEBUG);

如,默认时是info模式,未修改时,debug模式是不会输出的。

```
34
 330
         @Override
         public void run(String... args) throws Exception {
 35
             log.info("CommandLineRunner运行,info输出");
 36 //
             loggingSystem.setLogLevel(null, LogLevel.DEBUG);
             log.debug("CommandLineRunner运行:debug输出");
 37
 38
                                                          ☑ Console 🏻 🔛 Progress 👭 Servers 🔝 Problems Ju JUnit
chapter - ChapterApplication [Spring Boot App] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (2018年8月11日 上午
gistrationBean
                 : Mapping filter:
                                    characterEncodingFilter' to: [/*]
                                    hiddenHttpMethodFilter' to: [/*]
gistrationBean
                 : Mapping filter:
gistrationBean
                 : Mapping filter:
                                    'httpPutFormContentFilter' to: [/*]
gistrationBean : Mapping filter: 'tequestContextFilter' to: [/*]
agHandlerAdapter : Looking for @ContiollerAdvice: org.springframework.boot.context.emb
ngHandlerMapping : Mapped "{[/error] " onto public org.springframework.http.ResponseEn
agHandlerMapping : Mapped "{[/error], roduces=[text/html]}" onto public org.springfram
lHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.sprin
lHandlerMapping : Mapped URL path [/ ] onto handler of type [class org.springframewo
lHandlerMapping : Mapped URL path [/* /favicon.ico] onto handler of type [class org.s
nExporter : Registering beans for JMX exposure on startup
ServletContainer : Tomcat started on port(s): 8080 (http)
                 : CommandLineRunner运行,info输出
                 : Started ChapterApplication in 1.809 seconds (JVM running for 2.571)
oplication
                 : jar,chapter启动完成!
oplication
```

#### 动态设置后

```
goverride
 334
        public void run(String... args) throws Exception {
 35
             log.info("CommandLineRunner运行,info输出");
 36
             loggingSystem.setLogLevel(null, LogLevel.DEBUG);
             log.debug("CommandLineRunner运行:debug输出");
 37
 38
                                                 🕒 🥏 🗶 🔏 🔳 🗟 🚮 🚱 🖅
■ Console 🛭 🤛 Progress 🦚 Servers 📳 Problems Ju JUnit
chapter - ChapterApplication [Spring Boot App] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (2018年8月11日
            : Mapping filter: "httpPutFormContentFilter' to: [/*]
ationBean
ationBean : Mapping filter: 'requestContextFilter' to: [/*]
dlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embed
dlerMapping : Mapped "{[/error]] " onto public org.springframework.http.ResponseEnt:
dlerMapping : Mapped "{[/error] produces=[text/html]}" onto public org.springframet
lerMapping : Mapped URL path [ webjars/**] onto handler of type [class org.spring:
lerMapping : Mapped URL path [ **] onto handler of type [class org.springframeworl
lerMapping : Mapped URL path [ */favicon.ico] onto handler of type [class org.sp:
            : Registering beans for JMX exposure on startup
etContainer : Tomcat started on port(s): 8080 (http)
            CommandLineRunner运行,info输出
ation
             CommandLineRunner运行:debug输出
ation
            : Returning cached instance of singleton bean 'springApplicationAdminRe
Factory
            : Started ChapterApplication in 1.828 seconds (JVM running for 2.626)
ation
            : jar,chapter启动完成!
```

#### 执部署

前面讲了这么多章节,因为功能都很单一,所以一般上都是直接重启服务来进行更新操作。但当服务功能一多,启动速度缓慢时,还是配置个热部署比较方便。在SpringBoot中,只需要加入一个spring-boot-devtools即可

```
<dependencies>
  <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <optional>true</optional>
      </dependency>
</dependencies>
```

题外话:这里的<optional>true</optional>是表示依赖不会传递,依赖了此项目的需要额外引入此包,若需要使用的话。

若不生效,可试着在打包工具 spring-boot-maven-plugin下的 configuration 加入 <fork>true</fork>看看. 具体配置项如下:

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

# 自定义启动Banner

看烦了自带的Banner,动手修改一个属于自己的Banner,提现逼格的时候到了~哈哈,以下是官网给的配置指南:

#### 23.2 Customizing the Banner

The banner that is printed on start up can be changed by adding a banner.txt file to your classpath, or by setting banner.location to the location of such a file. If the file has an unusual encoding you can set banner.charset (default is UTF-8). In addition to a text file, you can also add a banner.gif, banner.jpg or banner.png image file to your classpath, or set a banner.image.location property. Images will be converted into an ASCII art representation and printed above any text banner.

#### 文字形式

其实,替换很简单,只需要在classpath路径下创建一个banner.txt即可。具体的一些变量官网也有给出,具体如下:

Variable	Description
{application.version}	The version number of your application as declared in MANIFEST.MF. For example Implementation-Version: 1.0 is printed as 1.0
{application.formatted-version}	The version number of your application as declared in MANIFEST.MF formatted for display (surrounded with brackets and prefixed with v). For example (v1.0)
{spring-boot.version}	The Spring Boot version that you are using. For example 1.5.15.RELEASE
{spring-boot.formatted-version}	The Spring Boot version that you are using formatted for display (surrounded with brackets and prefixed with $v$ ). For example $(v1.5.15.RELEASE)$
{Ansi.NAME} (or \${AnsiColor.NAME}, {AnsiBackground.NAME}, \${AnsiStyle.NAME})	Where NAME is the name of an ANSI escape code. See AnsiPropertySource for details.
\${application.title}	The title of your application as declared in MANIFEST.MF. For example  Implementation-Title: MyApp is printed as MyApp.

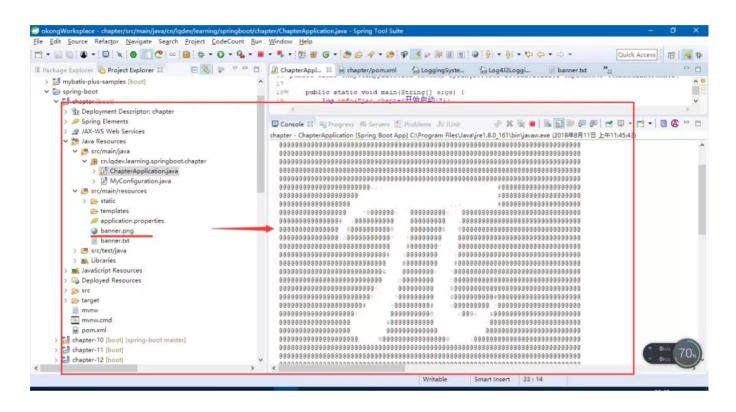
现在我们就定制一个自己的Banner。

题外话: 手输字符画是不太现实的, 大家可通过一些网站进行快速生成。可自行搜索下, 网上一搜一大把。

# 图片形式

若觉得使用文字不够酷炫,当然也可以将图片设置为启动的banner。目前支持的图片格式有qif、png、ipq。使用也很简单,只需要命名为banner即可。

如将头像放入目录中. 最后的效果如下:



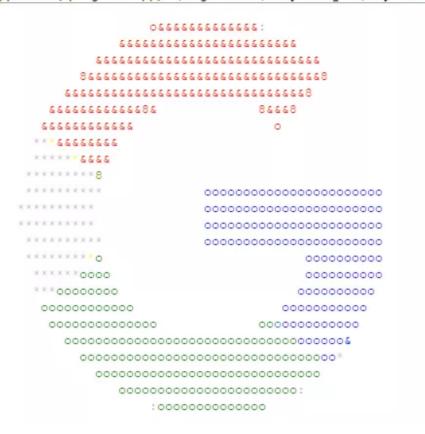
当然, 若图片是有色彩的, 也是可以的, 对于太复杂的图片显示效果就不佳了, 如下。

# 原图:



banner效果图:

chapter - ChapterApplication [Spring Boot App] C:\Program Files\Java\jre1.8.0\_161\bin\javaw.exe (2018年8月11日 上午1



是不是很酷炫~

#### 相关资料

1、https://docs.spring.io/spring-boot/docs/1.5.15.RELEASE/reference/htmlsingle

## 总结

本章节主要是简单的介绍了一些SpringBoot的一些小技巧,一般上也就一句话或者一个注释、一句配置就解决问题的。写这篇文章时,又去翻了翻官网的指南,很不错,每次都去看都有新发现。以上有部分就是看了写下的。确实,在看官网时,一般上是需要了解哪些知识点,就搜索直奔主题了,还没有哪次是从头看的。有时间还是耐心的看一看,就是全是英文看的有点头疼,好在代码是看的懂的, $\bigcirc_{\leftarrow}\bigcirc$  $\blacksquare$  $\blacksquare$ 

#### 一点吐槽

原本是想偷懒,发一点时间完成的。最后本着有图有真相且负责的原则,为了截图展现效果,实际操作了一遍,发现时间没有和写一篇正文来的少,好尴尬。。既然说了,就简单说下,接下来的章节会涉及的知识点吧。接下来还是web开发相关,会介绍下websocket