链滴

Spring Boot 特性 —— SpringApplication

作者：Ethan

原文链接：https://ld246.com/article/1524807352326

来源网站：链滴

使用SpringApplication引导项目启动

SpringApplication类为我们引导项目提供了一种便利的方式——通过main()方法直接启动。大多数况下，我们可以把项目启动这个任务直接委托给SpringApplication.run方法：

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

启动完成以后，你可以看到控制台输出如下：

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::  v1.4.1.RELEASE

2013-07-31 00:08:16.117  INFO 56603 --- [           main] o.s.b.s.app.SampleApplication            : Starting SampleApplication v0.1.0 on mycomputer with PID 56603 (/apps/myapp.jar started b pwebb)
2013-07-31 00:08:16.166  INFO 56603 --- [           main] ationConfigEmbeddedWebApplicatio Context : Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbedd ddWebApplicationContext@6e5a8246: startup date [Wed Jul 31 00:08:16 PDT 2013]; root of co text hierarchy
2014-03-04 13:09:54.912  INFO 41370 --- [           main] .t.TomcatEmbeddedServletContainerF ctory : Server initialized with port: 8080
2014-03-04 13:09:56.501  INFO 41370 --- [           main] o.s.b.s.app.SampleApplication            : Started SampleApplication in 2.992 seconds (JVM running for 3.658)```
Spring Boot 默认会输出`INFO`级别的日志，还包括一些相关的启动详细信息。

如果启动失败，Spring Boot 的`FailureAnalyzers`就会输出相关的错误信息，然后一个专用的处理器对这个错误进行相关的处理。Spring Boot 提供了很多这样的失败分析器，如果还不能满足要求，你可以自己通过实现`FailureAnalyzer`接口来构建自己的错误分析器。当然，如果你想看到更详细的异输出，你可以通过设置以下日志处理类的debug属性或者设置日志级别为`DEBUG`模式来达到这个目。

org.springframework.boot.autoconfigure.logging.AutoConfigurationReportLoggingInitializer

如果你使用`java -jar`命令来启动项目，那么你可以通过如下方式来设置debug属性：

```
$ java -jar myproject-0.0.1-SNAPSHOT.jar --debug
```

配置Banner

你可以通过添加一个`banner.txt`文件到你的classpath路径下，来改变在启动的时候打印的banner信，或者通过设置`banner.location`属性来设置该文件的位置，通过`banner.charset`来设置文件的编，你也可以添加`banner.gif`,`banner.jpg`,`` `banner.png``图片文件到classpath，或者通过设置`ban er.image.location`属性来作为banner信息，这些图片会被转换为有艺术感的ASCII，并且打印在文的顶部。 `banner.txt`中可以设置如下的占位符：
```

变量	描述
`${application.version}`	The version number of your application as declared in`MANIFEST.MF`. For example`Implementation-Version: 1.0`is printed as`1.0`.
`${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)`.
`${spring-boot.version}`	The Spring Boot version that you are using. For example`1.4.1.RELEASE`.
`${spring-boot.formatted-version}`	The Spring Boot version that you are using formatted for isplay (surrounded with brackets and prefixed with`v`). For example`(v1.4.1.RELEASE)`.
`${Ansi.NAME}`(or`${AnsiColor.NAME}`,`${AnsiBackground.NAME}`,`${AnsiStyle.NAME}`)	Where`NAME`is the name of an ANSI escape code. See[AnsiPropertySource](https://github.com/spring-projects/spring-boot/blob/v1.4.1.RELEASE/spring-boot/src/main/java/org/springframework/boot/ansi/AnsiPropertySource.java)for details.
`${application.title}`	The title of your application as declared in`MANIFEST.MF`. For example`Implementation-Title: MyApp`is printed as`MyApp`.

可以通过设置`spring.main.banner-mode`属性来控制输出，如下，在`application.properties`文件添加如下属性，将会覆盖SpringApplication中的默认配置：

打印到控制台

spring.main.banner-mode=console

打印到日志文件

spring.main.banner-mode=log

不打印

spring.main.banner-mode=off```

配置SpringApplication

如果SpringApplication无法满足要求，你可以自己创建一个局部实例，然后对其进行设置：

```
public static void main(String[] args) {

    SpringApplication app = new SpringApplication(MySpringConfiguration.class);
    //关闭Banner打印
    app.setBannerMode(Banner.Mode.OFF);
    //添加监听器
    app.addListeners(new MyListener());
    ...
    app.run(args);
}```
```

SpringApplication的相关配置将会被`@Configuration`注解的类，XML配置文件，以及Spring扫描包引用。更详细的配置选项，参见[SpringApplication](http://docs.spring.io/spring-boot/docs/1.41.RELEASE/api/org/springframework/boot/SpringApplication.html)Javadoc。

你也可以通过`SpringApplicationBuilder`来对SpringApplication的属性进行配置，这样的结构更有次感。`SpringApplicationBuilder`为构建[SpringApplication](http://docs.spring.io/spring-boot/ocs/1.4.1.RELEASE/api/org/springframework/boot/SpringApplication.html)和[ApplicationContxt](http://docs.spring.io/spring-framework/docs/4.3.3.RELEASE/javadoc-api/org/springframeork/context/ApplicationContext.html?is-external=true)实例提供了一套便利的流式API：

new SpringApplicationBuilder()

.sources(Parent.class)

.child(Application.class)

.bannerMode(Banner.Mode.OFF)

.listeners(new MyListener())

...

.run(args);```

SpringApplication将会根据需要创建一个ApplicationContext，默认情况下，如果是非web应用，会创建一个AnnotationConfigApplicationContext上下文，如果是web应用，则会创建一个AnnotatonConfigEmbeddedWebApplicationContext上下文。当然，你也可以通过setWebEnvironment(bolean webEnvironment)来覆盖默认的设置。

ApplicationRunner 和 CommandLineRunner

如果你希望在SpringApplication启动之前完成某些操作，你可以通过实现ApplicationRunner或者CmmandLineRunner接口来实现。这两个接口提供了一个run方法，会在SpringApplication.run(...)成之前被调用。适用于系统初始化配置的加载，启动检查等等。

```
import org.springframework.boot.*

import org.springframework.stereotype.*

@Component
public class MyBean implements CommandLineRunner {

    public void run(String... args) {
        // Do something...
    }

}```
```

如果有多个`CommandLineRunner`或者`ApplicationRunner`的实现，并且这些都需要按照一定的序来执行，你可以通过实现`org.springframework.core.Ordered`接口或者使用`org.springframewok.core.annotation.Order`注解来设置这些的执行顺序。