glmapper

SpringBoot 实践系列-外部化配置优先级问题

#springboot

字数统计: 1.3k 阅读时长: 6 min

2020/01/03 9 Share

本文主要针对 `spring.profiles.active` 、 `spring.config.location` 以及 `spring.config.additional-location` 的作用机制及优先级问题进行实践对比。

> 本文案例工程已上传 github 仓库： https://github.com/glmapper/springboot-series-guides/tree/master/guides-properties

spring.profiles.active

除了 application.properties 文件之外，profile-specific 配置也可以通过以下命名方式来定义:application-{profile}.properties。在没有使用 active 指定 profiles 的情况下，Environment 会指定一组默认的 profiles（默认情况下是[default]），换句话说就是，如果没有显示的激活 profiles 配置文件，则默认加载的是 application-default.properties 配置文件。

profile-specific 配置文件的属性与标准 application.properties 从相同的位置加载（一般是 classpath 下）； profile-specific 指定的 properties 配置文件始终覆盖默认配置。

在案例工程中(guides-properties)，resources 下面包括 application.properties 和 application-dev.properties 两份配置文件

- application.properties 文件配置

```
spring.application.name=appNameInner
testKey=key-default
```

- application-dev.properties 文件配置

```
testKey=key-dev
```

通过以下代码在启动时将配置值输出：

```

 @Value("${testKey}")
 private String testKey;
```

```
4
5  @PostConstruct
6  private void init(){
7      System.out.println("--------------------------------");
8      System.out.println(testKey);
9      System.out.println("--------------------------------");
10 }
```


不指定 spring.profiles.active 时

通过 `java -jar guides-properties/target/guides-properties-0.0.1-SNAPSHOT.jar` 启动工程，console 输出如下:

```
1  2020-01-04 00:08:47.279  INFO 11050 --- [           main] com.glmapper.bridge.boot.BootStrap       : No activ
2  --------------------------------
3  key-default
4  --------------------------------
```

结论是，如果不显示指定 profiles，则使用默认的。

指定 spring.profiles.active 时

通过 `java -jar -Dspring.profiles.active=dev guides-properties/target/guides-properties-0.0.1-SNAPSHOT.jar` 启动工程，console 输出如下:

```
1  2020-01-04 00:08:14.426  INFO 11040 --- [           main] com.glmapper.bridge.boot.BootStrap       : The foll
2  --------------------------------
3  key-dev
4  --------------------------------
```

结论是，在显示指定 profiles 的情况下，会覆盖默认 application.properties 中的配置值。

spring.config.location

在 SpringBoot 2.x 中 spring.config.location 的语义发生了变更(此项配置会导致 classpath 中的 application.properties 不再生效)。原因如下:

```
1  private Set<String> getSearchLocations() {
2      // spring.config.location 直接使用此份文件，不会再处理其他配置文件
3      if (this.environment.containsProperty(CONFIG_LOCATION_PROPERTY)) {
4          return getSearchLocations(CONFIG_LOCATION_PROPERTY);
5      }
6      Set<String> locations = getSearchLocations(CONFIG_ADDITIONAL_LOCATION_PROPERTY);
7      locations.addAll(
8              asResolvedSet(ConfigFileApplicationListener.this.searchLocations, DEFAULT_SEARCH_LOCATIONS));
9      return locations;
10 }
```

在工程的根目录的 `conf` 目录下新建一个 `application-conf.properties` 配置文件，内容如下:

```
1  testKey=key-spring.config.location
```

通过 `java -jar -Dspring.config.location=conf/application-conf.properties guides-properties/target/guides-properties-0.0.1-SNAPSHOT.jar` 启动工程，发现启动报错，原因是因为 application-conf.properties 中没有 配置 `spring.application.name` ，而 `spring.application.name` 是在 resources 目录下的 application.properties 中的，所以也间接说明前面提到的，会使 classpath 下的配置失效。新增 `spring.application.name` 之后，重新启动工程，

```
1  spring.application.name=guides-properties
2  testKey=key-spring.config.location
```

输出结果如下:

```
1  2020-01-04 00:19:12.225  INFO 11147 --- [           main] com.glmapper.bridge.boot.BootStrap       : No activ
2  --------------------------------
```

```
key-spring.config.location
------------------------------
```


所以在使用 `spring.config.location` 指定外部配置文件时，需要此份配置文件需全量满足当前工程运行时所需，因为它不会去与 resources 目录下的配置文件去做 merge 操作。

spring.config.additional-location

在使用 `spring.config.additional-location` 这种方式自定义 locations 时，除了默认 locations 之外，还会使用 `spring.config.additional-location` 指定的。

> additional-location：言外之意就是增量的配置

在工程的根目录的 `conf` 目录下新建一个 `application-addition.properties` 配置文件，内容如下：

```
testKey=key-addition
```

通过 `java -jar -Dspring.config.additional-location=conf/application-addition.properties guides-properties/target/guides-properties-0.0.1-SNAPSHOT.jar` 启动工程，输出结果如下：

```
2020-01-04 00:28:30.048  INFO 11384 --- [           main] com.glmapper.bridge.boot.BootStrap       : No activ
------------------------------
key-addition
------------------------------
```

结论是，会覆盖默认 application.properties 中的配置值。

spring.config.additional-location 与 spring.profiles.active 配置加载关系

`spring.config.location` 不用多说，它就是独立的一份，使用它就不能使用其它的。所以这里只分析 `spring.config.additional-location` 与 `spring.profiles.active` 配置加载关系。

同时指定两个配置

通过 `java -jar -Dspring.profiles.active=dev -Dspring.config.additional-location=conf/application-addition.properties guides-properties/target/guides-properties-0.0.1-SNAPSHOT.jar` 启动工程，输出如下：

```
2020-01-04 00:32:59.044  INFO 11451 --- [           main] com.glmapper.bridge.boot.BootStrap       : The foll
------------------------------
key-dev
------------------------------
```

为了排除与 -D 参数顺序有关，也使用如下方式再执行一次： `java -jar -Dspring.config.additional-location=conf/application-addition.properties -Dspring.profiles.active=dev guides-properties/target/guides-properties-0.0.1-SNAPSHOT.jar` ，输出结果与前面相同，所以可以得出，`spring.profiles.active` 的优先级比 `spring.config.additional-location` 要高。

`spring.config.additional-location` 指定差异增量配置

在 `spring.config.additional-location` 中增加 additionKey

```
testKey=key-addition
additionKey=testAddition
```

使用 `java -jar -Dspring.config.additional-location=conf/application-addition.properties -Dspring.profiles.active=dev guides-properties/target/guides-properties-0.0.1-SNAPSHOT.jar` 启动工程，输出如下：

```
2020-01-04 11:44:42.227  INFO 12821 --- [           main] com.glmapper.bridge.boot.BootStrap       : The foll
------------------------------
key-dev
testAddition
------------------------------
```

结论是 `spring.config.additional-location` 可以用于提供出 profiles 机制或者默认方式之外的增量配置。

小结

在使用外部化配置文件时，执行顺序为：

`spring.config.location` > `spring.profiles.active` > `spring.config.additional-location` > 默认的 `application.proerties` 。

其中通过 `spring.profiles.active` 和 `spring.config.additional-location` 指定的配置文件会与 默认的 `application.proerties` merge 作为最终的配置， `spring.config.location` 则不会。

< Next Post
SpringBoot 源码系列-自动配置及 starter 机制解析

Previous Post >
SpringBoot 源码系列-配置解析