

说说SpringBoot的配置文件(*)

author: hushuang

email : hd1611756908@gmail.com

adress: www.dajiangdahe.com

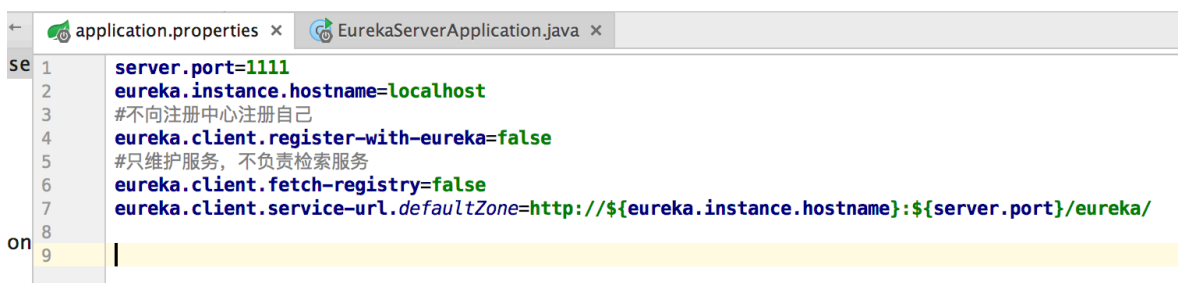
说说SpringBoot的配置文件，配置文件不论在什么项目中都很重要，项目的初始化参数设置大多都是在配置文件中完成。

SpringBoot的配置文件分为两种方式：

1. application.properties方式
2. application.yml 方式

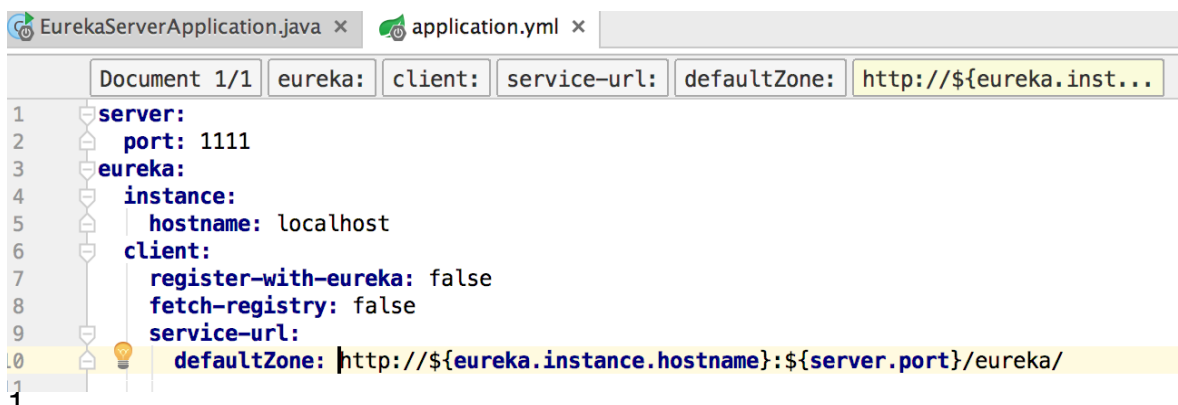
这两种配置文件的方式用哪种都是可以的，只是一般推荐采用YAML的方式，他在语法方面简洁更易读，不过有时候有特殊设置的时候这种语法是不支持的，以后再说。

1. properties文件的方式我们比较熟悉，就是一个键值对key:value的方式，看下面的例子

A screenshot of an IDE window showing the 'application.properties' file. The file contains the following properties:

```
server.port=1111
eureka.instance.hostname=localhost
#不向注册中心注册自己
eureka.client.register-with-eureka=false
#只维护服务，不负责检索服务
eureka.client.fetch-registry=false
eureka.client.service-url.defaultZone=http://${eureka.instance.hostname}:${server.port}/eureka/
```

2. YAML的方式取数据的时候是一个数组的方式，不管是什么方式Spring都会自己去读取和解析，咱们只是遵循他的语法格式去配置就好。

A screenshot of an IDE window showing the 'application.yml' file. The file contains the following YAML configuration:

```
server:
  port: 1111
eureka:
  instance:
    hostname: localhost
  client:
    register-with-eureka: false
    fetch-registry: false
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

YAML语法有一个比较特殊的地方

```
instance:
  hostname: localhost
```

就是：冒号后面要有一个空格如果没有空格就会发现冒号前面的属性不会变颜色，所以hostname：后面加一个空格然后在赋值，这是一个语法格式问题。

后面我会全部采用YAML方式配置

下面我说一下配置文件的获取，这才是SpringBoot配置文件的重中之重，不是什么属性配置知道配置文件中都会自动被加载的，下面就看看哪个配置可以自动加载。

其实如果你在写上面配置文件的时候他要是提示的，这样的配置文件一般都是你加了相对应的依赖默认Spring就会去加载。如果是自定义的配置，名称什么都是自定义的那么这样的配置文件我们就要自己去读取，虽然是自己读取但是Spring也提供了非常好的方式。

下面分别说配置文件的读取：

1.系统提示的配置文件的读取，首先得知道为什么系统会提示，肯定是加了依赖，加了依赖之后系统才会提示，但是也不是加什么依赖都会提示，要是仔细看SpringBoot的依赖管理会发现SpringBoot的依赖有一个共同的特点，它里面都会带一个starter这样的字样，SpringBoot就会根据这样的依赖去猜测你要建的是一个什么样类型的项目，再启动的时候就会直接进行初始化配置。

```
spring-boot-starter-test
spring-cloud-starter-eureka-server
```

也不是加什么样子的依赖都可以，必须是按照Spring的标准来的才可以，如果我随便加一个连接池依赖Durid (<https://github.com/alibaba/druid/wiki/%E5%B8%B8%E8%A7%81%E9%97%AE%E9%A2%98>) 官网没有为Spring提供类似于starter这样的依赖所以配置他的时候需要定制化配置，配置需要我们自定义。

2.说说自定义配置信息的获取

—>下面是我随便自定义的几个配置信息：



The screenshot shows an IDE with three tabs: application.yml, SpringbootDemo02Application.java, and TestHandler.java. The application.yml file is open, showing the following content:

```

1
2 address: 中国
3 company: 百度
4 number: 10
5
6 #还可以这样获取拼接配置信息
7 content: ${address}:${company}:${number}

```

配置完之后怎么取出来呢，下面就是获取配置信息需要的注解和获取的格式，用@value注解可以获取配置文件中的配置信息语法格式见下图



The screenshot shows a Java class TestHandler with the following code:

```

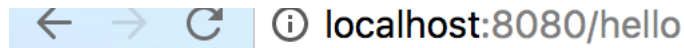
@RestController
public class TestHandler {

    @Value("${address}")
    private String address;
    @Value("${company}")
    private String company;
    @Value("${number}")
    private Integer number;
    @Value("${content}")
    private String content;

    @GetMapping("/hello")
    public String helloTest(){
        return address+" : "+company+" : "+number + " == "+content;
    }
}

```

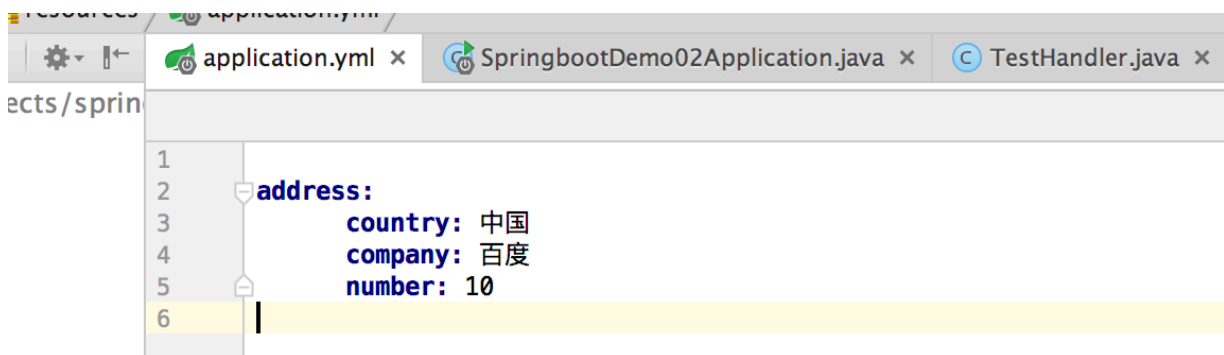
下面就是给前端返回的信息



中国：百度：10==中国:百度:10

上面是其中一种获取自定义配置文件的方式，这种获取配置文件的方式虽然方便，但是要是配置文件较多的话会比较麻烦，要一个个的获取。下面介绍另一种获取配置文件的方式

—>自定义配置文件信息



这样的配置文件怎么获取呢？

首先得定义一个POJO类随便取名字我就叫Address，定义三个属性，属性名称必须和配置文件里面的属性名称一一对应，否则会匹配不上，通过@ConfigurationProperties注解在配置上前缀就可以将配置信息匹配到这个Address类中，上面要再加一个@Component注解，因为要加到Spring容器进行管理，为什么加Component这个注解，因为我们用这个配置类时候需要进行注入，如果不加Component注解，就会在使用的时候注入失败，下面有例子解释为什么加Component注解。配置方式见下图。get/set方法，为了节省地方省略了。

```

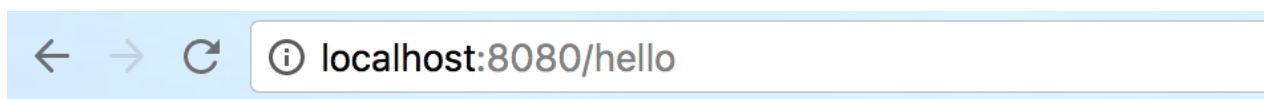
8  */
9  @Component
10 @ConfigurationProperties(prefix = "address")
11 public class Address {
12
13     private String country;
14     private String company;
15     private Integer number;
16

```

下面是将配置文件取出来的方式：是通过Autowired这个注解注入进来的。

```
Created by IntelliJ on 2017/5/25.  
*/  
@RestController  
public class TestHandler {  
  
    @Autowired  
    private Address address;  
  
    @GetMapping("/hello")  
    public String helloTest(){  
        return address.toString();  
    }  
}
```

运行现象：



A screenshot of a web browser's address bar. It features navigation icons (back, forward, refresh) on the left and an information icon on the right. The address text is 'localhost:8080/hello'.

Address{country='中国', company='百度', number=10}

以上就是SpringBoot配置文件的基本用法，这只是配置文件的一小部分，SpringBoot配置文件还支持多环境配置和CLI方式启动配置等。

下面在说一下多环境配置

一般的项目从开发到上线会分好几个环境，分别应用于不同的场景，比如我们开发的时候要加载开发环境的配置，测试的时候要加载测试环境的配置，线上的时候要加载线上的环境配置，如果没有分环境的配置，统一要用一个配置文件，那么从开发—测试—上线，会频繁的改配置文件，很容易出错，现在其他项目也会采用分环境配合，SpringBoot当然也不例外。

SpringBoot的多环境配置文件要遵循SpringBoot的文件命名格式

application-{profile}.properties 或者是 application-{profile}.yml

例如：

application-dev.properties ：开发环境

application-test.properties ： 测试环境

application-prod.properties ： 生产环境

然后在写一个总的配置文件用于加载各个环境的配置文件

application.properties

在这个总的配置文件中配置这个属性

```
spring.profiles.active=dev
```

来选择加载哪个配置文件

但是这样每次还是得修改配置文件中spring.profiles.active=dev 这个值还是比较麻烦。

所以SpringBoot还提供了命令行启动的方式

SpringBoot启动方式有三种我可以通过java -jar 的方式给他传入一个参数用于修改加载的配置文件。

例如：

```
java -jar xxx.jar --spring.profiles.active=dev
```

像这样就可以加载dev的配置文件了，如果其他环境就将dev修改就可以了。

以上就是简单的SpringBoot配置管理