# 微服务大量重复配置

大量的微服务工程,application全局配置文件,非常非常多的重复配置的.并且不好管理---这些全局配置文件的问题可以交给spring cloud config.

# springcloud config结构

服务端和客户端

服务端：连接公共网络资源读取公用的配置属性

客户端：连接服务端,按照需求读取他使用的配置

|  |
| --- |
| 计算机生成了可选文字: 商 品 系 统 config-client  mybatis  datas  哪 个 版 本  用 户 系 统 config-client  config—server  配 置 服 务 端  u S e r  pass  公 用 配 置 github/git/gitee  不 同 的 配 置 技 术  mybatis.yml  datasource-druid.ym  datasource-h3.ymI  datasource-c3pO.ymI  redis ． 泖 讠 式 环 境 1 ． 、 ml  redis- 测 试 环 境 2.ymI  rabbitmq-testl .yml |

# client从server获取配置的方式

可以通过访问config-server配置中心，决定拿到git上哪些properties/yml的配置文件.

配置中心提供了多种不同的接口地址(mapping映射地址)，让config-client选择性的获取远程git的不同配置文件.

例：

localhost:8098/redis-test1.properties//读取 redis-test1.properties一个文件

localhost:8098/redis/test1// (一次读取到多个文件,包括 redis-test1.properties)

### 访问格式：

|  |
| --- |
| /application/profile  /application-profile.[json/yml/properties] |

**application：**

应用名称：使用应用名字区分各种技术,

例如：redis.properties, mybatis.properties

**profile：**

配合application,给同一个应用提供不同的配置内容.因为即使都是redis的配置,也区分测试环境和线上生产环境.

例如：redis-test1.properties,redis-test2.properties,redis-product.properties

所以为了迎合这样的格式,准备git上的公用配置文件是一般都是

application-profile.yml/json/properties

### 特殊情况

#### 只有application没有profile的配置文件

读取的特点，只要application同名就能读到里面的内容.

做同一个应用同一个技术的公用配置

#### 名称不是严格的application-profile格式

例如：demo01-test02-version01.properties

下例都能访问该配置文件：

demo01-test02==可解析为application

demo01==可解析为application

### 结论

配置中心一旦搭建 config-server就会按照他提供的接口接收我们的请求,解析请求中的参数,读取git的不同配置文件.

# 搭建config-server

可以通过访问这个config-server配置中心，决定拿到git上哪些.properties/.yml的配置文件.

配置中心提供了多种不同的接口地址(mapping映射地址)，让config-client选择性的获取远程git的不同配置文件。

* maven项目
* pom.xml
  + 继承:springboot
  + 导入:springcloud
  + 依赖:
    - config-server
  + application.properties
    - 端口:8098
    - git服务器仓库地址
    - git仓库下的文件夹

server.port=8098

### 导入依赖

|  |
| --- |
| <!--config服务器--> <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-config-server</artifactId> </dependency> |

### application.properties

|  |
| --- |
|  |
| server.port=8098 #git服务器的仓库 gitee配置 spring.cloud.config.server.git.uri=https://gitee.com/easymall-test/easymall-config #私有仓库账号密码 公有仓库不需要 spring.cloud.config.server.git.username=15723468981 spring.cloud.config.server.git.password=Sxs411051  #git的某个仓库下是否有多个文件夹，可以指定多个或一个 #spring.cloud.config.server.git.search-paths=configtest #**TODO 高可用，微服务注册** |

### 启动类

|  |
| --- |
| @SpringBootApplication @EnableConfigServer public class StarterConfig {  public static void main(String[] args) {  SpringApplication.*run*(StarterConfig.class,args);  } } |

# 搭建config-client

当一个web应用引入config-client之后,在web应用启动完整之前,最先做的事就是一个client进程读取config-server的所有配置属性，交给web应用去使用

* + maven项目
  + pom:
    - 继承:springboot
    - 导入:springcloud
    - 依赖:
      * config-client
      * starter-web
  + application.properties(不再使用)
  + bootstrap.properties (使用这个配置文件)

### application.properties与bootstrap.properties区别

计算机生成了可选文字:
config-client 
先 加 #)bootstrap 
读 取 
属 性 赋 值 
配 置 中 心 属 
web 应 用 
后 加 fiapplication 
rCe 
属 性 赋 值 
mybatis 
微 服 务 
配 置 读 取 配 置 中 心 的 辑 
oca 丨 host ： 8098 ， 哪 些 prop rties 
bootstrap.properties 
plicat 

### 导入依赖

|  |
| --- |
| <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-config</artifactId> </dependency> |

### bootstrap.properties

|  |
| --- |
| server.port=8099 #config-client继承指定读取远程公用配置内容 #redis-test1.properties #当前客户端想要读取的 redis datasource mybatia 应用名称 spring.cloud.config.name=redis spring.cloud.config.profile=test1 #发起的请求就是 localhost:8098/redis/test1 #指定一下config-server地址 spring.cloud.config.uri=http://localhost:8098 |

# config-server高可用

这个结构一旦config-server宕机,网络延迟,就无法让客户端访问获取必要的一些属性。

解决高可用访问思路：服务治理组件,只要让config-client通过服务调用的方式，访问多个配置中心服务端就行了。

### config-server作为euerka客户端在注册中心注册服务

#### 增加依赖eureka

|  |
| --- |
| <!--eureka 客户端--> <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-eureka</artifactId> </dependency> |

#### 增加配置

|  |
| --- |
| spring.application.name=config-server eureka.client.service-url.defaultZone=http://127.0.0.1:8761/eureka eureka.instance.prefer-ip-address=true |

### config-client作为eureka客户端在注册中心抓取服务'

#### 增加依赖eureka

|  |
| --- |
| <!--eureka 客户端--> <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-eureka</artifactId> </dependency> |

#### 完整配置

|  |
| --- |
| server.port=8099 #config-client继承指定读取远程公用配置内容 #redis-test1.properties #当前客户端想要读取的 redis datasource mybatia 应用名称 spring.cloud.config.name=redis spring.cloud.config.profile=test1 #发起的请求就是 localhost:8098/redis/test1 #指定一下config-server地址 #spring.cloud.config.uri=http://localhost:8098 #微服务调用 spring.application.name=config-client eureka.client.service-url.defaultZone=http://127.0.0.1:8761/eureka eureka.instance.prefer-ip-address=true #通过服务访问配置中心config-server spring.cloud.config.discovery.enabled=true spring.cloud.config.discovery.service-id=config-server |