# Spring Cloud Config Server 工程构建说明

Spring Cloud Config Server为外部配置（名称 - 值对或等效的YAML内容）提供基于HTTP资源的API。 通过使用@EnableConfigServer注释，Spring-Cloud-Config 服务器就可嵌入Spring Boot应用程序中。

# application.yml.bc1 -------- 简单基础使用方式

## 登陆码云构建自己的 git 仓库

* 访问 https://gitee.com/projects/new，新建自己的microservice-config-repo git仓库，然后 git 克隆到本地
* 新建文件application.yml，其内容profile: profile-default
* 新建文件foobar.yml，其内容profile: profile-dev
* Git 提交到码云仓库地址

## application.yml 文件中配置

* 与所有Spring Boot应用程序一样，它默认在端口8080上运行，但可以通过各种方式将其切换到更传统的端口8080。最简单的方法是设置默认配置存储库，方法是使用spring.config.name = configserver启动它（Config Server jar中有一个configserver.yml）。另一种方法就是使用自己的application.properties 或 application.yml 中配置，如以下示例所示：

server:

port: 8080

# 配置Spring Cloud Config Server 的存储库

spring:

cloud:

config:

server:

git:

uri: https://gitee.com/baiHoo/microservice-config-repo.git #后缀 .git 可以不要

* 其中${user.home}/config-repo 是一个包含YAML和属性文件的git存储库。
* 注意：在Windows上，如果文件URL是绝对的驱动器前缀，则需要额外的“/”（例如，file：///${user.home}/ config-repo）。

## 测试顺序

### 启动 microservice-config-server 模块服务，启动1个端口8080

* 新起网页页签，输入http://localhost:8080/ ，网页会显示错误
* 网页页签，输入http://localhost:8080/master/任意字符串-default.yml 或 http://localhost:8080/任意字符串-default.yml，网页显示如下：

-----------------------------------------------------------

profile: profile-default

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库application.yml中内容

* 网页页签，输入http://localhost:8080/master/任意字符串-dev.yml 或 http://localhost:8080/任意字符串-dev.yml，网页显示如下：

-----------------------------------------------------------

profile: profile-default

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库application.yml中内容

* 网页页签，输入http://localhost:8080/master/foobar-dev.yml 或 http://localhost:8080/ foobar -dev.yml，网页显示如下：

-----------------------------------------------------------

profile: profile-dev

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库foobar.yml中内容

* [why?](#_登陆码云构建自己的_git_仓库)

因为这个 HTTP请求的资源来自：

/{application}/{profile}[/{babel}]

/{application}-{profile}.yml

/{label}/{application}-{profile}.yml

/{application}-{profile}.properties

/{label}/{application}-{profile}.properties

"label"是git操作标签"master"

# application.yml.bc2 -------- 通配符使用

## 登陆码云新构建 git 仓库

* 访问 https://gitee.com/projects/new，新建的两个命名为simple 和special 的git仓库，然后 git 克隆到本地
* simple仓库目录新建文件application.yml，其内容profile: simple
* special仓库目录新建文件application.yml，其内容profile: simple
* Git 提交到码云仓库地址

## git 通过应用名占位符方式，配置远程存储库

Spring Cloud Config Server支持带有{application}和{profile}（以及{label}的占位符的git存储库URL（如果需要），但请记住该标签仍然作为git标签应用）。 因此，您可以使用类似于以下的结构来支持“每个应用程序一个存储库”策略：

spring:

cloud:

config:

server:

git:

uri: https://github.com/myorg/{application**}**

# application.yml.bc3 -------- 模式匹配和多个存储库

Spring Cloud Config支持更复杂的需求，并在应用程序和配置文件名称上进行模式匹配。 模式格式是带有通配符的{application} / {profile}名称的逗号分隔列表（请注意，可能需要引用以通配符开头的模式），如以下示例所示：

spring:

cloud:

config:

server:

git:

uri: https://github.com/spring-cloud-samples/config-repo

repos:

simple: https://github.com/simple/config-repo

special:

pattern: special\*/dev\*,\*special\*/dev\*

uri: https://github.com/special/config-repo

* special仓库目录新建文件special-dev.yml，其内容profile: special-dev
* special仓库目录新建文件special-test.yml，其内容profile: special-test
* Git 提交到码云仓库地址

## 启动 microservice-config-server 模块服务，启动1个端口8080

* 新起网页页签，输入http://localhost:8080/ ，网页会显示错误
* 网页页签，输入http://localhost:8080/master/任意字符串-default.yml 或 http://localhost:8080/任意字符串-default.yml，网页显示如下：

-----------------------------------------------------------

profile: profile-default

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库application.yml中内容

* 网页页签，输入http://localhost:8080/master/任意字符串-dev.yml 或 http://localhost:8080/任意字符串-dev.yml，网页显示如下：

-----------------------------------------------------------

profile: profile-default

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库application.yml中内容

* 网页页签，输入http://localhost:8080/master/foobar-dev.yml 或 http://localhost:8080/ foobar -dev.yml，网页显示如下：

-----------------------------------------------------------

profile: profile-dev

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库foobar.yml中内容

# application.yml.bc4 -------- 搜索路径

每个存储库还可以选择将配置文件存储在子目录中，搜索这些目录的模式可以指定为searchPaths。以下示例显示了顶级的配置文件：

spring:

cloud:

config:

server:

git:

uri: https://github.com/spring-cloud-samples/config-repo

searchPaths: foo,bar\*

* microservice-config-repo仓库目录新建目录foo,foo目录下新建文件foo-dev.yml，其内容profile: foo-dev
* smicroservice-config-repo仓库目录新建目录bar,bar目录下新建文件bar-dev.yml，其内容profile: bar-dev
* Git 提交到码云仓库地址

## 启动 microservice-config-server 模块服务，启动1个端口8080

* 网页页签，输入http://localhost:8080/bar-dev.yml网页显示如下：

-----------------------------------------------------------

profile: bar-dev

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库bar目录下bar-dev.yml中内容

* 网页页签，输入http://localhost:8080/foo-dev.yml，网页显示如下：

-----------------------------------------------------------

profile: foo-dev

-----------------------------------------------------------

注意：网页命中的内容来自提交到仓库foo目录下foo-dev.yml中内容

# application.yml.bc5 -------- cloneOnStart

默认情况下，服务器在首次请求配置时克隆远程存储库。 可以将服务器配置为在启动时克隆存储库，如以下顶级示例所示：

spring:

cloud:

config:

server:

git:

uri: https://git/common/config-repo.git

repos:

team-a:

pattern: team-a-\*

cloneOnStart: **true**

uri: http://git/team-a/config-repo.git

team-b:

pattern: team-b-\*

cloneOnStart: **false**

uri: http://git/team-b/config-repo.git

team-c:

pattern: team-c-\*

uri: http://git/team-a/config-repo.git

# application.yml.bc6 -------- 账号密码