Spring Cloud Config 分享

ScienJus 2017.7

主要内容

- 背景介绍
- 快速入门
- 实现原理
- 高级特性

背景介绍

Spring Cloud 是什么

一套构建分布式应用的基础组件，包括并不限于：

- 配置管理
- 服务注册与发现
- 断路器
- ...

其中 Spring Cloud Config 就是配置管理的组件，用于微服务场景下使用的统一配置中心。

为什么需要配置中心

- 代码与配置分离
- 安全性
- 配置共享

在整个服务由一个单体应用拆分成数十个微服务后，配置管理的成本成为一个不可忽视的问题。

为什么选择 Spring Cloud Config

- 客户端与 Spring Boot 无缝结合
- 依托于 Git 管理配置，管理复杂度低，有着天生的版本控制和读写权限
- 官方维护，社区活跃度良好
- 去中心化，高可用的集群

快速入门

客户端集成指南

配置文件的对应关系

Spring Cloud Config 支持 application, label 和 profile 三个维度的配置管理，也就是说可以通过这三个配置的值获取到一个唯一的配置文件。

例如：给 demo 项目的 dev 环境创建一个 master 配置。

在 Git 中创建配置文件

在 Git 中，application 对应于一个根目录的文件夹，profile 和本地的配置文件一样以 application-$profile.yml 作为文件名，label 为 Git 的分支名。

所以需要创建 /demo/application-dev.yml 文件，并提交到 master 分支。

在 Server 端预览配置

提交配置后，就可以实时的在 Server 端预览到配置内容了，访问地址的对应关系为：

```
http://host/$application/$profile/$label
```

也就是：

```
http://configserver/demo/dev/master
```

在本地应用加载配置

引入依赖:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-config-client</artifactId>
</dependency>
```

增加 Spring Cloud Config 配置：

```yaml
spring:
  application:
    name: demo
  profiles:
    active: dev
  cloud:
    config:
      uri: http://configserver:8888
      label: master
```

其中 spring.application.name、spring.profiles.active 和 spring.cloud.config.label 分别对应 application、profile 和 label。

像本地配置一样注入

```
@Value("${foo.bar}")
private String bar;
```

或是：

```
@ConfigurationProperties(prefix = "foo")
public class FooConfig {

  private String bar;
}
```

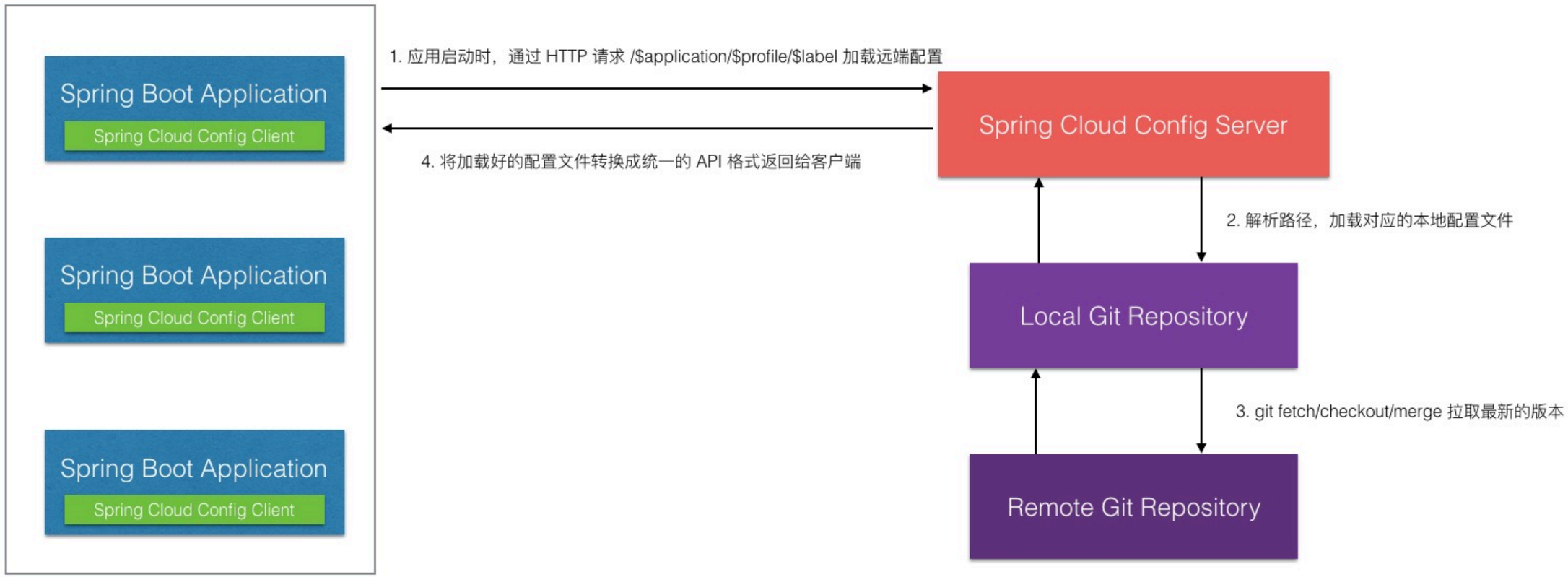
亦或是：

```
@Autowired
private Environment env;

env.getProperty("foo.bar", "empty");
```

实现原理

配置加载流程


Spring Boot Application
Spring Cloud Config Client
Spring Boot Application
Spring Cloud Config Client
Spring Boot Application
Spring Cloud Config Client
1. 应用启动时，通过 HTTP 请求 /$application/$profile/$label 加载远端配置
4. 将加载好的配置文件转换成统一的 API 格式返回给客户端
Spring Cloud Config Server
2. 解析路径，加载对应的本地配置文件
Local Git Repository
3. git fetch/checkout/merge 拉取最新的版本
Remote Git Repository

Server 端实现

Web：`ResourceController#resolve`

AutoConfiguration：`GitRepositoryConfiguration`

Service：`JGitEnvironmentRepository#getLocations`

Client 端实现

BootstrapConfiguration：`ConfigServiceBootstrapConfiguration`

PropertySourceLocator：`ConfigServicePropertySourceLocator`

高级特性

- 鉴权
- 多仓库
- 配置项加密
- 非 Spring 配置文件
- 热更新