博学谷
www.boxuegu.com

第七章 Spring Cloud Alibaba Nacos 分布式配置

一样的在线教育，不一样的教学品质

博学谷
www.boxuegu.com

目录 Contents

- Nacos 配置管理的应用
- 配置动态刷新与回滚
- 配置共享
- 配置优先级

Nacos 配置管理的应用

小节导学

Nacos 配置管理如何使用?

如何在管理控制台创建配置?

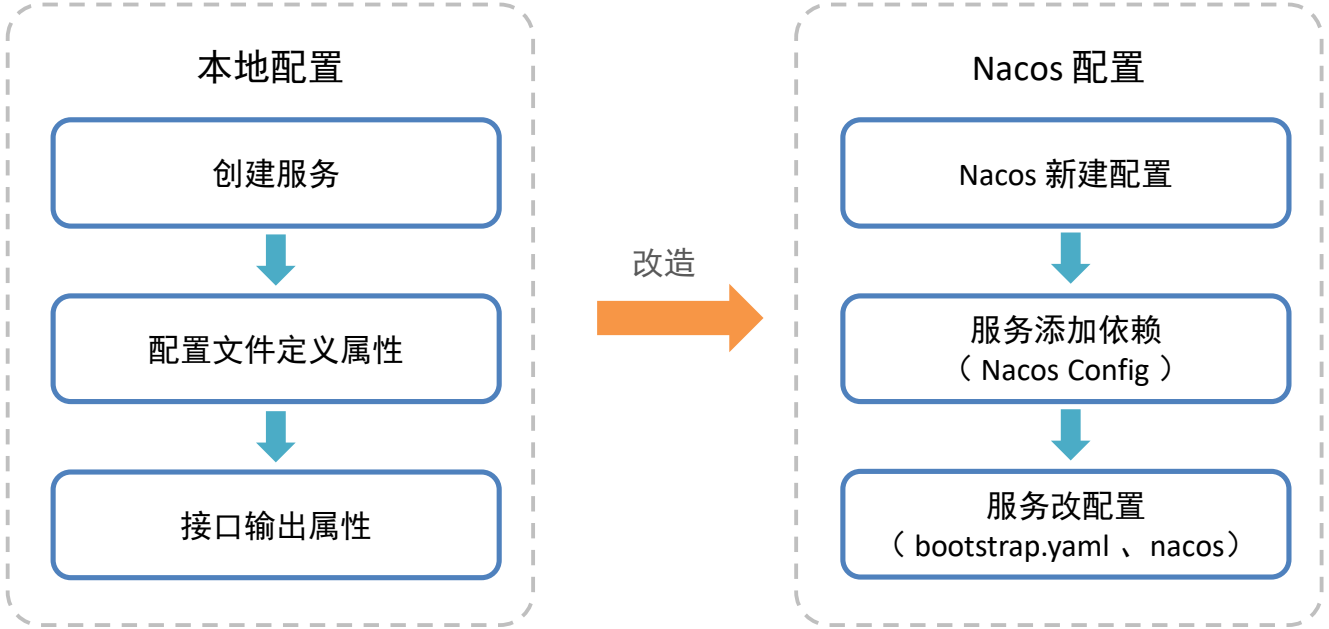
如何在项目中对接 Nacos?

本节我们的目标就是把一个本地配置改为 Nacos 配置方式。

- 本地配置
- Nacos 配置

Nacos 配置管理的应用

实践流程



1. 本地配置

目标

创建一个服务，使用本地配置的方式，在接口中把本地配置的属性值输出出来。

步骤

1. 属性配置

```
test:
  name: local-config
```

3. 启动 访问：`http://localhost:8001/testconfig`

   输出：`local-config`

2. 构建接口

```
@Value("${test.name}")
private String testName;

@GetMapping("/testconfig")
public String testConfig(){
    return testName;
}
```

2. Nacos 配置

目标

把本地属性配置改为 Nacos 配置。

步骤

1. Nacos 中创建配置

data id: service-config.yml

配置内容：

```
test:
  name: nacos-config
```

2. Nacos 配置

步骤

2. 添加 Nacos Config 依赖

```
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
</dependency>
```

2. Nacos 配置

步骤

3. 创建 bootstrap.yml

添加最核心的应用配置：应用名、nacos 连接信息、配置文件的后缀名

```yaml
spring:
  application:
    name: service-config
  cloud:
    nacos:
      config:
        server-addr: nacos.me:8848
        file-extension: yml
```

注意：

1. 必须使用 bootstrap.yml，优先级最高
2. nacos 中 Data ID 名字约定：

[application-name].[file-extension]

2. Nacos 配置

验证

启动服务

启动日志：

……

```
Located property source: CompositePropertySource {name='NACOS',
propertySources=[NacosPropertySource {name='service-config.yml'}, NacosPropertySource
{name='service-config'}]}
```

……

访问：`http://localhost:8001/testconfig`

输出：`nacos-config`

Nacos 配置管理的应用-总结

重难点

1. Nacos 管理界面中属性配置方式
2. Date ID 命名规则
3. 项目中整合 Nacos 配置的方式

目录 Contents

- Nacos 配置管理的应用
- 配置动态刷新与回滚
- 配置共享
- 配置优先级

配置动态刷新与回滚

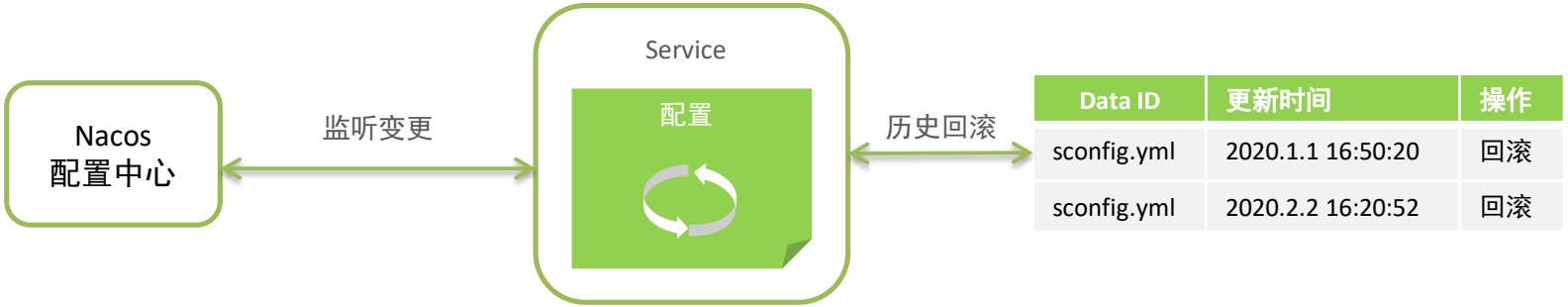
小节导学

应用对接 Nacos 配置中心后，Nacos 中变更后应用中的配置可以自动刷新。

对于配置的变更，Nacos 会维护一个历史版本的记录，可以随时做回滚操作。

本节我们的目标就是实现动态刷新，并实践历史版本的回滚。

- 配置动态刷新
- 历史版本回滚



Data ID	更新时间	操作
sconfig.yml	2020.1.1 16:50:20	回滚
sconfig.yml	2020.2.2 16:20:52	回滚

1. 配置动态刷新

目标

Nacos 中修改配置，应用中输出更新后的属性值。

步骤

1. 代码中添加刷新支持代码

```
@RefreshScope
@RestController
public class TestController {
    @Value("${test.name}")
    private String testName;
    @GetMapping("/testconfig")
    public String testConfig(){
        return testName;
    }
}
```

1. 配置动态刷新

步骤

2. Nacos 中修改属性配置

```
test:
  name: nacos-config-new
```

3. 验证

访问：`http://localhost:8001/testconfig`

输出：`nacos-config-new`

2. 历史版本回滚

目标

Nacos 中选择某一历史版本回滚，应用中可以输出此版本属性值。

步骤

1. Nacos 中历史版本回滚
2. 应用中输出历史属性值

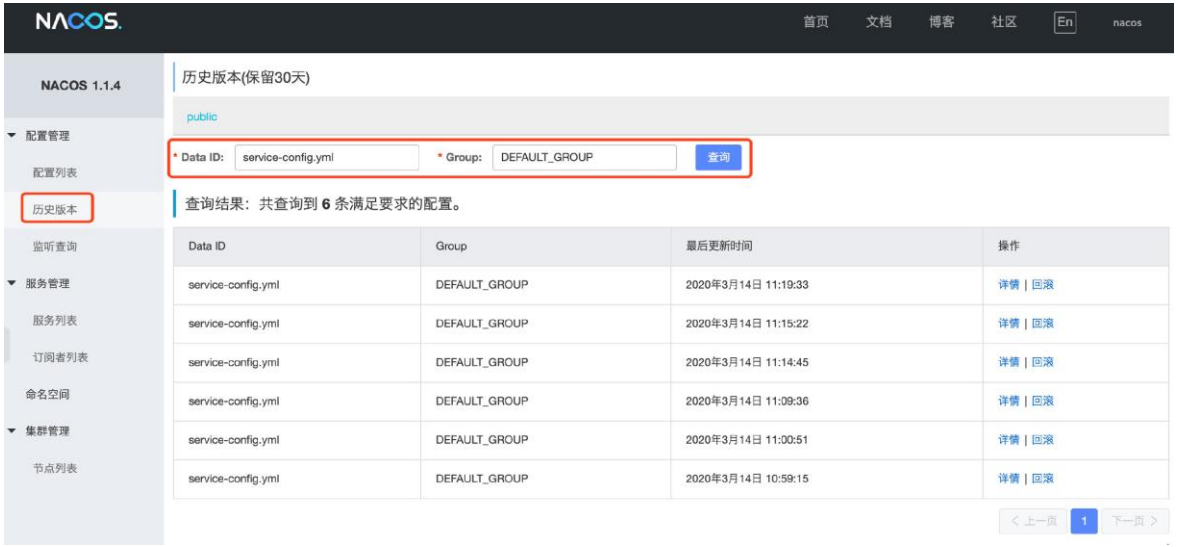
2. 历史版本回滚

问题

不能选择最初的历史版本 会造成此配置丢失

这是 Nacos 的一个问题，需要特别留意一下，官方说 1.2 版本会解决

如果不小心遇到了这个问题，可以在历史版本中找回



配置动态刷新与回滚-总结

重难点

1. Nacos 动态刷新的开发方法
2. Nacos 历史回滚的操作方法
3. Nacos 目前 1.X 版本中回滚注意事项

目录 Contents

配置共享

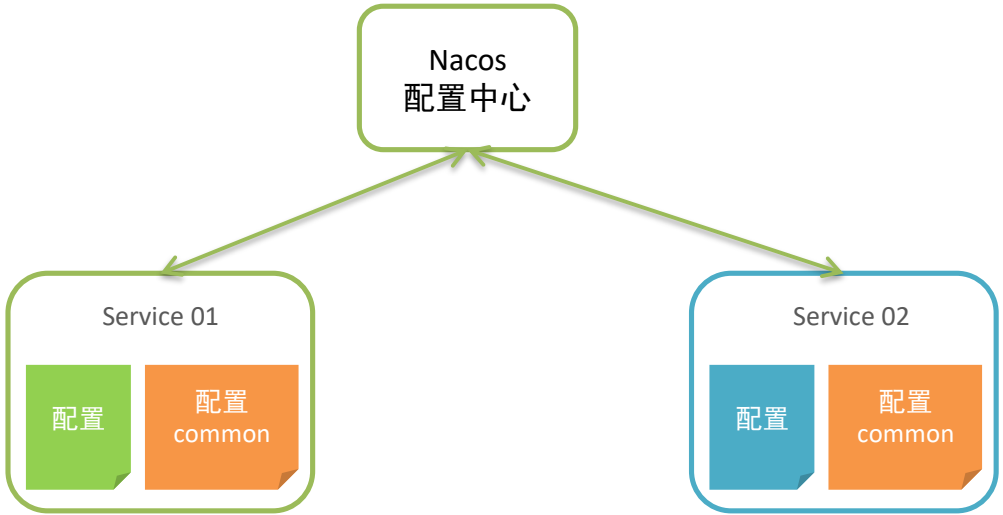
小节导学

有时我们需要共享一些配置信息，例如：

- 每个服务都需要配置连接同一个服务注册中心
- 同一个服务的 dev 环境和 test 环境配置连接同一个数据库

本节我们学习如何实现配置的共享。

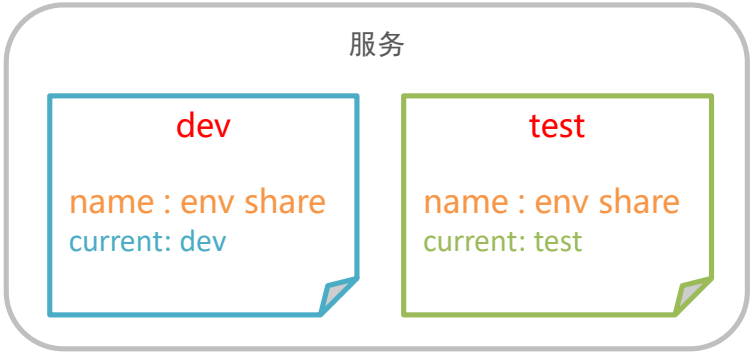
- 不同环境间的配置共享
- 服务间的配置共享

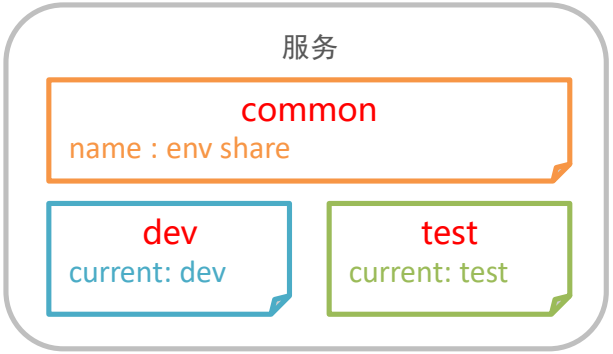


1. 不同环境间的配置共享

dev 与 test 环境有一个相同的属性“name : env share”，维护2份比较复杂。

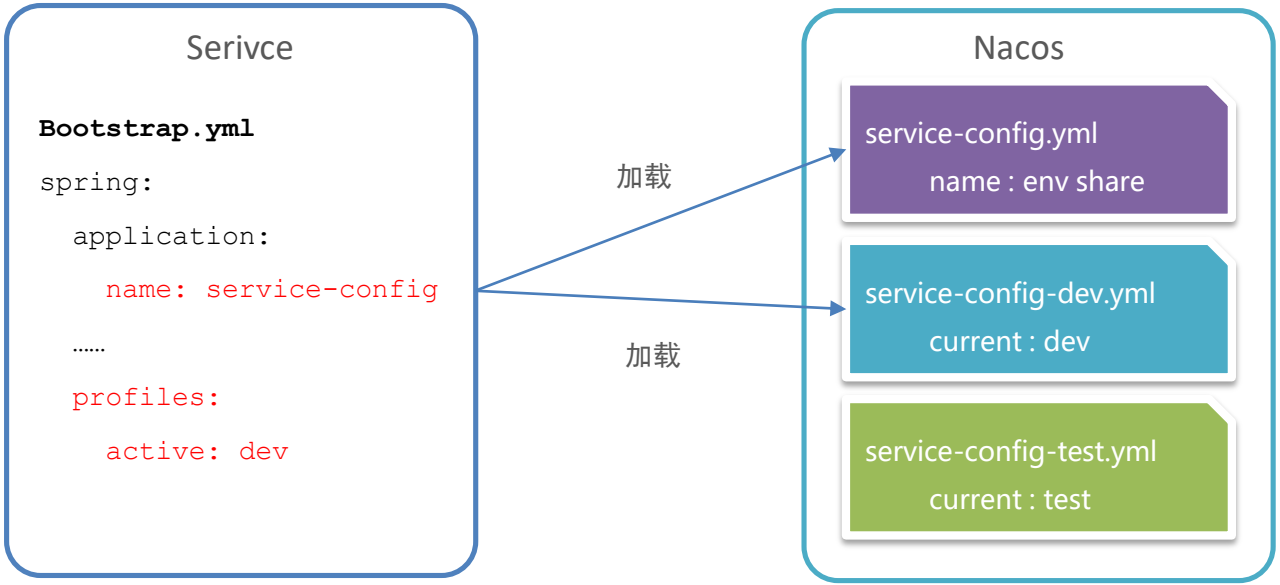
希望有一个共享机制，把相同的属性放入放入公共的配置文件中，不论什么环境都可以引用此配置文件。





1. 不同环境间的配置共享

实现



1. 不同环境间的配置共享

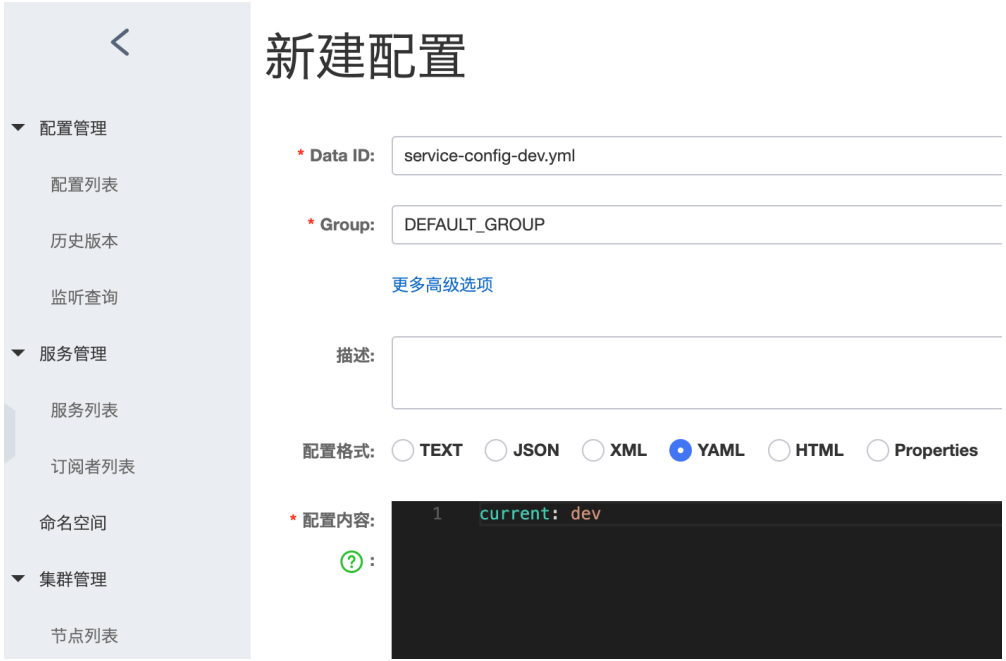
步骤

1. Nacos 中创建配置“service-config.yml”，添加配置 name: env share

1. 不同环境间的配置共享

步骤

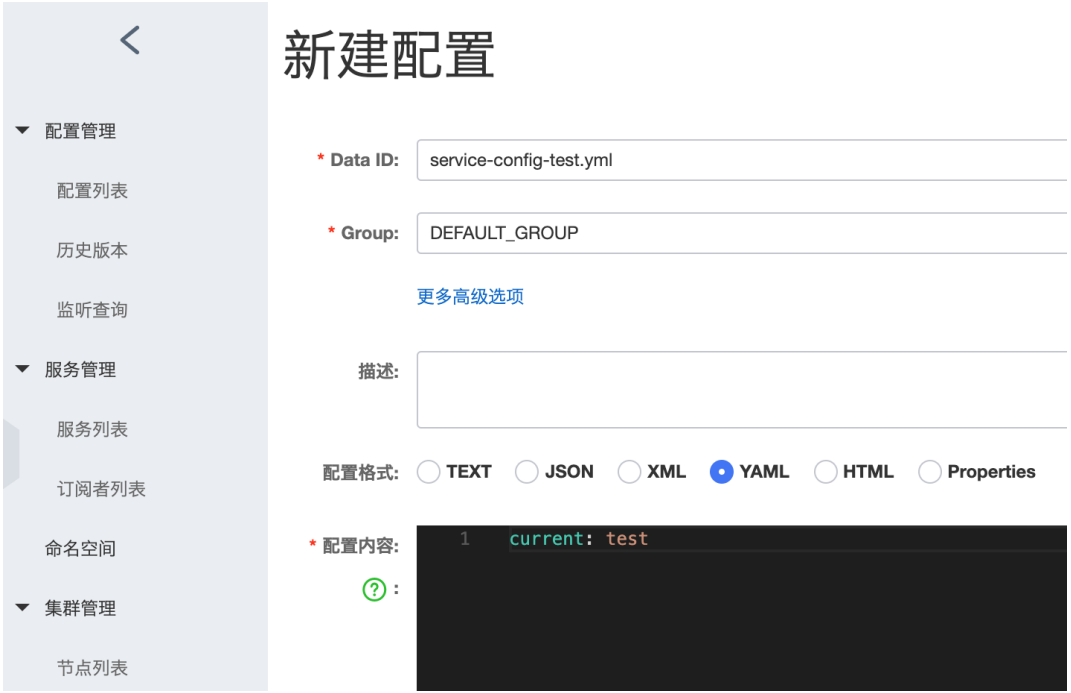
2. Nacos 中创建配置“service-config-dev.yml”，添加配置 current: dev



1. 不同环境间的配置共享

步骤

3. Nacos 中创建配置“service-config-test.yml”，添加配置 current: test



1. 不同环境间的配置共享

步骤

4. 创建接口

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

@Value("${current}")
private String current;

@GetMapping("/envshare")
public String envshare(){
  return "name=" + name + ", current=" + current;
}
```

1. 不同环境间的配置共享

步骤

5. 测试

```
spring:
  application:
    name: service-config
  cloud:
    nacos:
      config:
        server-addr: nacos.me:8848
        file-extension: yml
  profiles:
    active: dev
```

访问 http://localhost:8001/envshare

输出 name=env share, current=dev

```
spring:
  application:
    name: service-config
  cloud:
    nacos:
      config:
        server-addr: nacos.me:8848
        file-extension: yml
  profiles:
    active: test
```

访问 http://localhost:8001/envshare

输出 name=env share, current=test

1. 不同环境间的配置共享

要点

- Nacos 配置 Data ID 命名约定

默认配置：[application name] . [file-extension]

特定环境配置：[application name] - [profile] . [file-extension]

- 特定环境配置优先级高于默认配置
- 默认配置相当于各个环境下的公共配置

2. 服务间的配置共享

目标

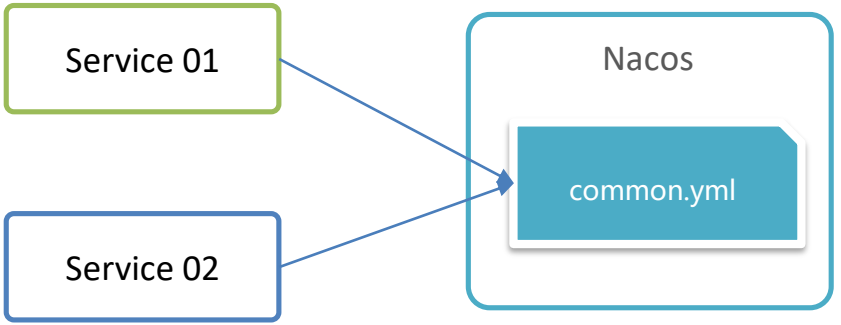
service01、service02 这2个应用都可以引用配置文件 `common.yml`

输出其中的配置“`test.common: common-config`”

实现方式

通过以下2中方式可以引用其他配置文件：

1. shared-dataids
2. ext-config



2. 服务间的配置共享

方式1：shared-dataids

```
spring:
  application:
    name: service01
  cloud:
    nacos:
      config:
        server-addr: nacos.me:8848
        file-extension: yml
        shared-dataids: common.yml
        refreshable-dataids: common.yml
```

2. 服务间的配置共享

方式2：ext-config

```
spring:
  application:
    name: service01
  cloud:
    nacos:
      config:
        server-addr: nacos.me:8848
        file-extension: yml
        ext-config:
          - data-id: common.yml
            group: DEFAULT_GROUP
            refresh: true
```

2. 服务间的配置共享

2个方式的差异

- shared-dataids 只加载 DEFAULT_GROUP 下的配置

```
...
    nacos:
      config:
        shared-dataids: common.yml # 即使指定了 group 此处仍加载 DEFAULT_GROUP 下的
        refreshable-dataids: common.yml
        group: group01
```

- ext-config 可以灵活指定 group

配置共享-总结

重难点

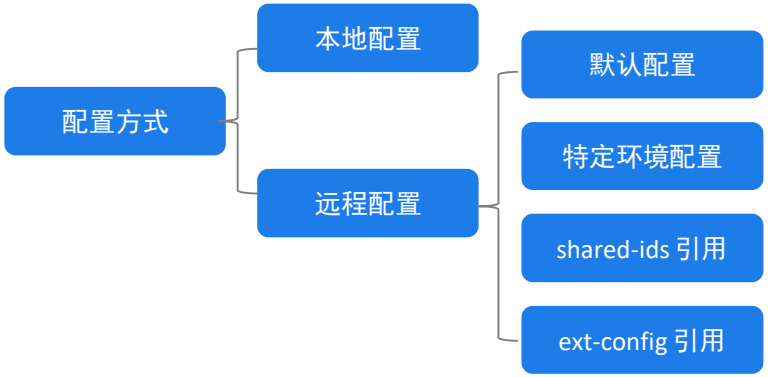
1. 不同环境间共享配置的方式
2. 不同服务间共享配置的2种方式
3. shared-ids 与 ext-config 对于 group 的差异性

目录 Contents

配置优先级

小节导学

现在我们已经学习了多种配置方式，例如：



那么就有了一个重要的问题：各种配置方式的优先级是怎样的？

本节我们就对比一下各种方式的优先级

- 本地与远程配置对比
- 远程配置方式对比

1. 本地与远程配置对比

目标

本地配置文件与 Nacos 中都配置同一个属性，通过接口输出此属性值，得出优先级。

步骤

1. Nacos 中 service-config.yml 与本地 bootstrap.yml 中都配置同一属性

```
test:
  name: nacos-config
```

```
test:
  name: local-config
```

2. 接口中输出“nacos-config”，得出远程配置优先级更高

1. 本地与远程配置对比

修改本地配置优先级

在 Nacos 中配置：

```
spring:
  cloud:
    config:
      # 外部源配置是否可被覆盖
      allowOverride: true
      # 外部源配置是否不覆盖任何源
      overrideNone: true
      # 外部源配置是否可覆盖本地属性
      overrideSystemProperties: false
```

2. 远程配置对比

目标

我们已经知道环境特定配置优先于默认配置，现在对比：

默认配置、shared-ids、ext-config

步骤

1. Nacos 中添加3个配置：

```
# service-config.yml
test:
  priority: application-name
# service-config-shared.yml
test:
  priority: shared
# service-config-ext.yml
test:
  priority: ext
```

2. 远程配置对比

步骤

2. bootstrap.yml 配置：

```
spring:
  application:
    name: service-config
  cloud:
    nacos:
      config:
        server-addr: nacos.me:8848
        file-extension: yml
        shared-dataids: service-config-shared.yml
        ext-config:
          - dataId: service-config-ext.yml
```

2. 远程配置对比

步骤

3. 测试接口

```
@Value("${test.priority}")
private String test_priority;

@GetMapping("/testpriority")
public String testpriority(){
    return test_priority;
}
```

4. 测试

访问 `http://localhost:8001/testpriority`，输出 `application-name`

得出 默认配置优先级最高

配置优先级

结论

配置优先级-总结

重难点

1. 配置方式列表
2. 各种配置方式的优先级
3. 本地配置优先级调整方式

一样的在线教育，不一样的教学品质