链滴

微服务——Nacos 第二篇

作者：strangebob

原文链接：https://ld246.com/article/1681885402803

来源网站：链滴

上次学的Nacos注册中心还未完成，现在来继续学习

Nacos注册中心□第二篇

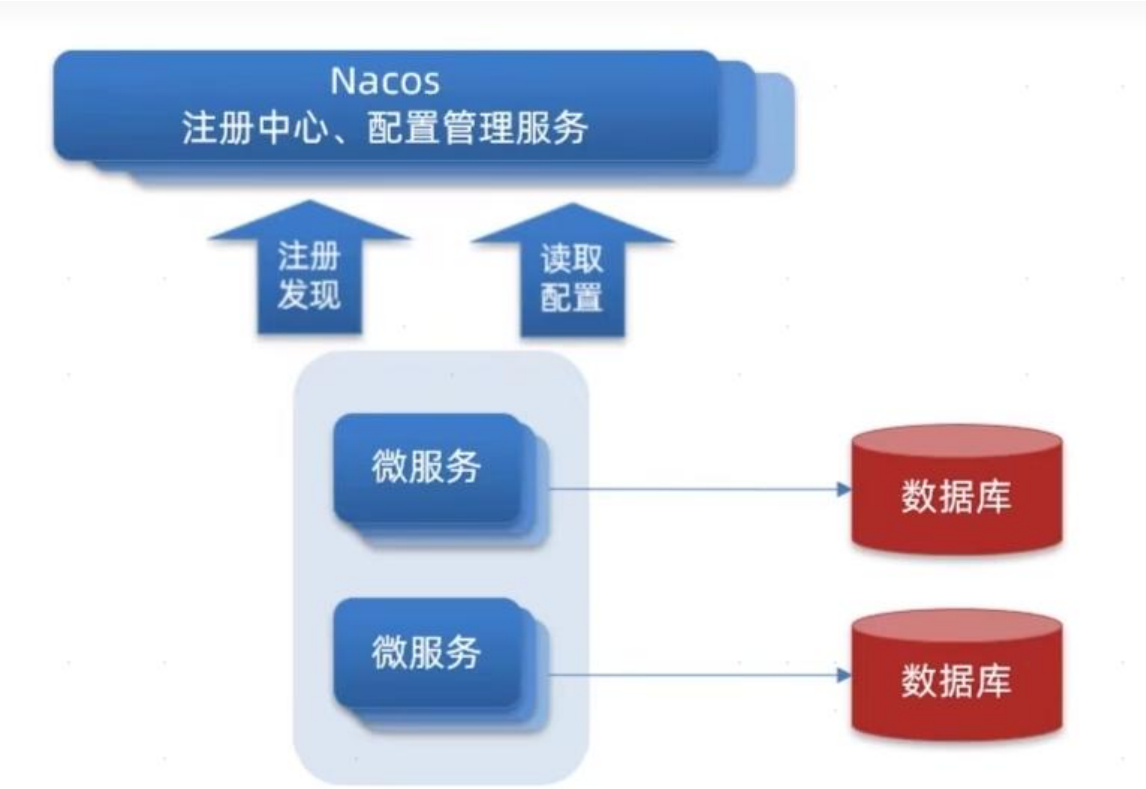
- Nacos配置管理
- Feign远程调用
- Gateway服务网关

1.Nacos配置管理

Nacos除了可以做注册中心，同样可以做配置管理来使用。

1.1.统一配置管理

当微服务部署的实例越来越多，达到数十、数百时，逐个修改微服务配置就会让人抓狂，而且很容易错。我们需要一种统一配置管理方案，可以集中管理所有实例的配置。

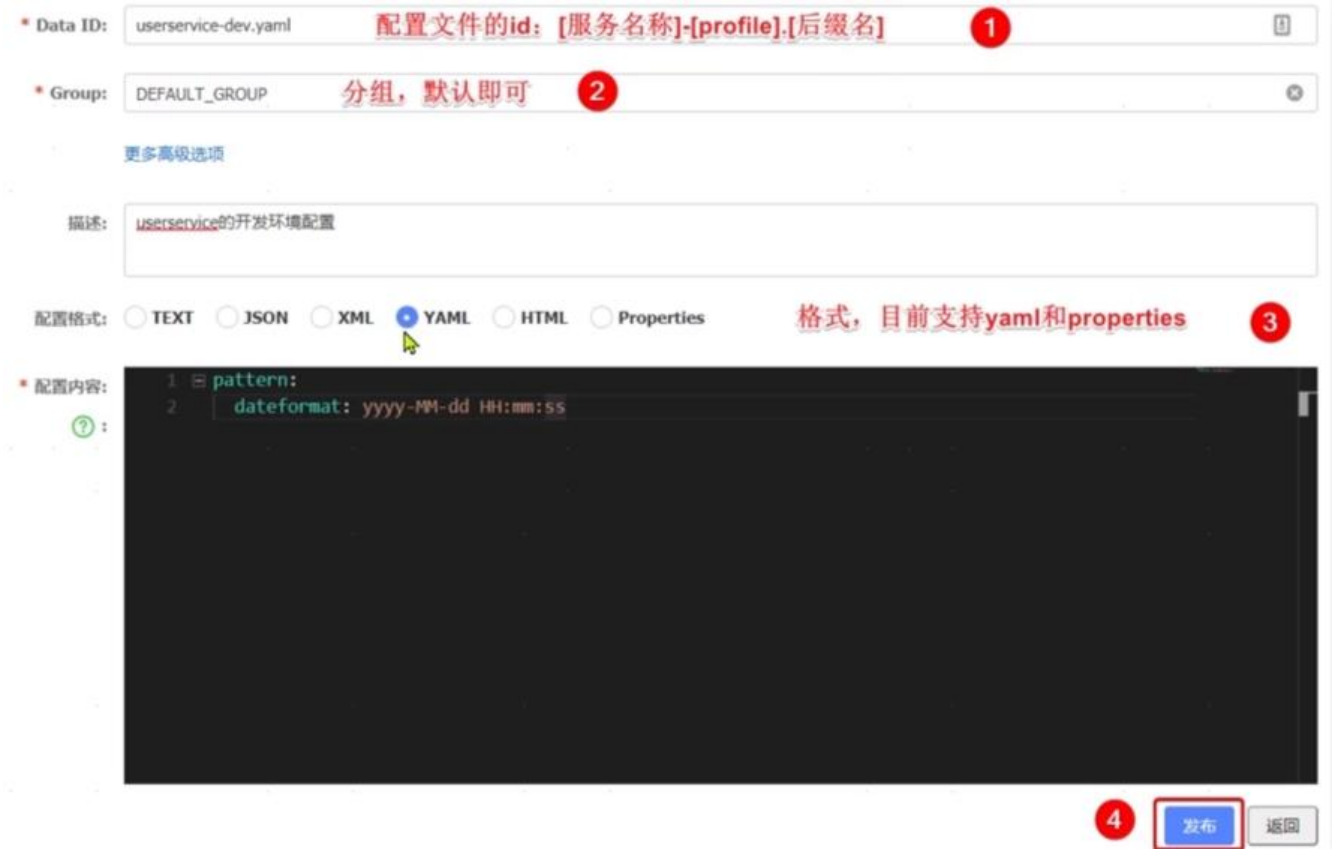


Nacos一方面可以将配置集中管理，另一方可以在配置变更时，及时通知微服务，实现配置的热更新。

1.1.1在nacos中添加配置文件

如何在nacos中管理配置呢?

然后在弹出的表单中，填写配置信息：



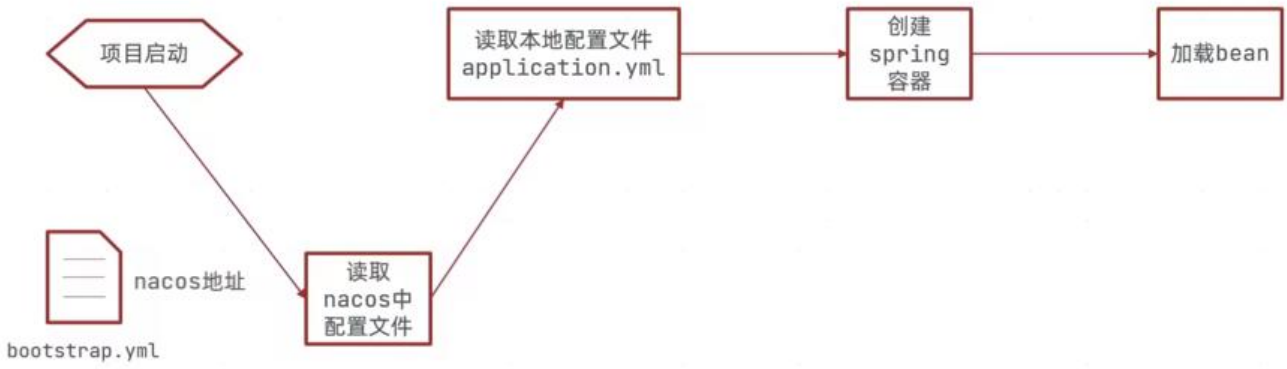
注意：项目的核心配置，需要热更新的配置才有放到nacos管理的必要。基本不会变更的一些配置还保存在微服务本地比较好。

1.1.2从微服务拉取配置

微服务要拉取nacos中管理的配置，并且与本地的application.yml配置合并，才能完成项目启动。

但如果尚未读取application.yml，又如何得知nacos地址呢?

因此spring引入了一种新的配置文件：bootstrap.yaml文件，会在application.yml之前被读取，流如下：



1）引入nacos-config依赖

首先，在user-service服务中，引入nacos-config的客户端依赖：

```
<!--nacos配置管理依赖-->
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
</dependency>
```

2）添加bootstrap.yaml

然后，在user-service中添加一个bootstrap.yaml文件，内容如下：

```
spring:
  application:
    name: userservice # 服务名称
  profiles:
    active: dev #开发环境，这里是dev
  cloud:
    nacos:
      server-addr: localhost:8848 # Nacos地址
      config:
        file-extension: yaml # 文件后缀名
```

这里会根据spring.cloud.nacos.server-addr获取nacos地址，再根据

${spring.application.name}-${spring.profiles.active}.${spring.cloud.nacos.config.file-extension}
为文件id，来读取配置。

本例中，就是去读取userservice-dev.yaml

3）读取nacos配置

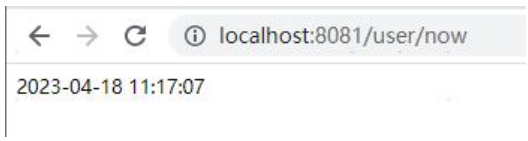
在user-service中的UserController中添加业务逻辑，读取pattern.dateformat配置：

在controller中写入以下代码

```
@Value("${pattern.dateformat}")
    private String dateformat;

    @GetMapping("now")
    public  String now(){
        return LocalDateTime.now().format(DateTimeFormatter.ofPattern(dateformat));
    }
```

在页面访问，可以看到效果：



1.2.配置热更新

我们最终的目的，是修改nacos中的配置后，微服务中无需重启即可让配置生效，也就是**配置热更新。**

要实现配置热更新，可以使用两种方式：

1.2.1.方式一

在@Value注入的变量所在类上添加注解@RefreshScope：

```
@Slf4j
@RestController
@RequestMapping("/user")
@RefreshScope
public class UserController {

    @Value("${pattern.dateformat}")
    private String dateformat;
```

1.2.2.方式二

使用@ConfigurationProperties注解代替@Value注解。

在user-service服务中，添加一个类，读取patterrn.dateformat属性：

```
package cn.itcast.user.config;

import lombok.Data;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@Component
```

```java
@Data
@ConfigurationProperties(prefix = "pattern")
public class PatternProperties {
    private String dateformat;
}
```

在UserController中使用这个类代替@Value：

```java
@Autowired
private PatternProperties properties;


@GetMapping("now")
public  String now(){
    return LocalDateTime.now().format(DateTimeFormatter.ofPatte
}

/**
```

1.2.3总结

Nacos配置更改后，微服务可以实现热更新，方式：

① 通过@Value注解注入，结合@RefreshScope来刷新

② 通过@ConfigurationProperties注入，自动刷新

注意事项：

- 不是所有的配置都适合放到配置中心，维护起来比较麻烦
- 建议将一些关键参数，需要运行时调整的参数放到nacos配置中心，一般都是自定义配置

1.3.配置共享

其实微服务启动时，会去nacos读取多个配置文件，例如：

- [spring.application.name]-[spring.profiles.active].yaml，例如：userservice-dev.yaml
- [spring.application.name].yaml，例如：userservice.yaml

而[spring.application.name].yaml不包含环境，因此可以被多个环境共享。

下面我们通过案例来测试配置共享

1.3.1添加一个环境共享配置

我们在nacos中添加一个userservice.yaml文件：

1.3.2在user-service中读取共享配置

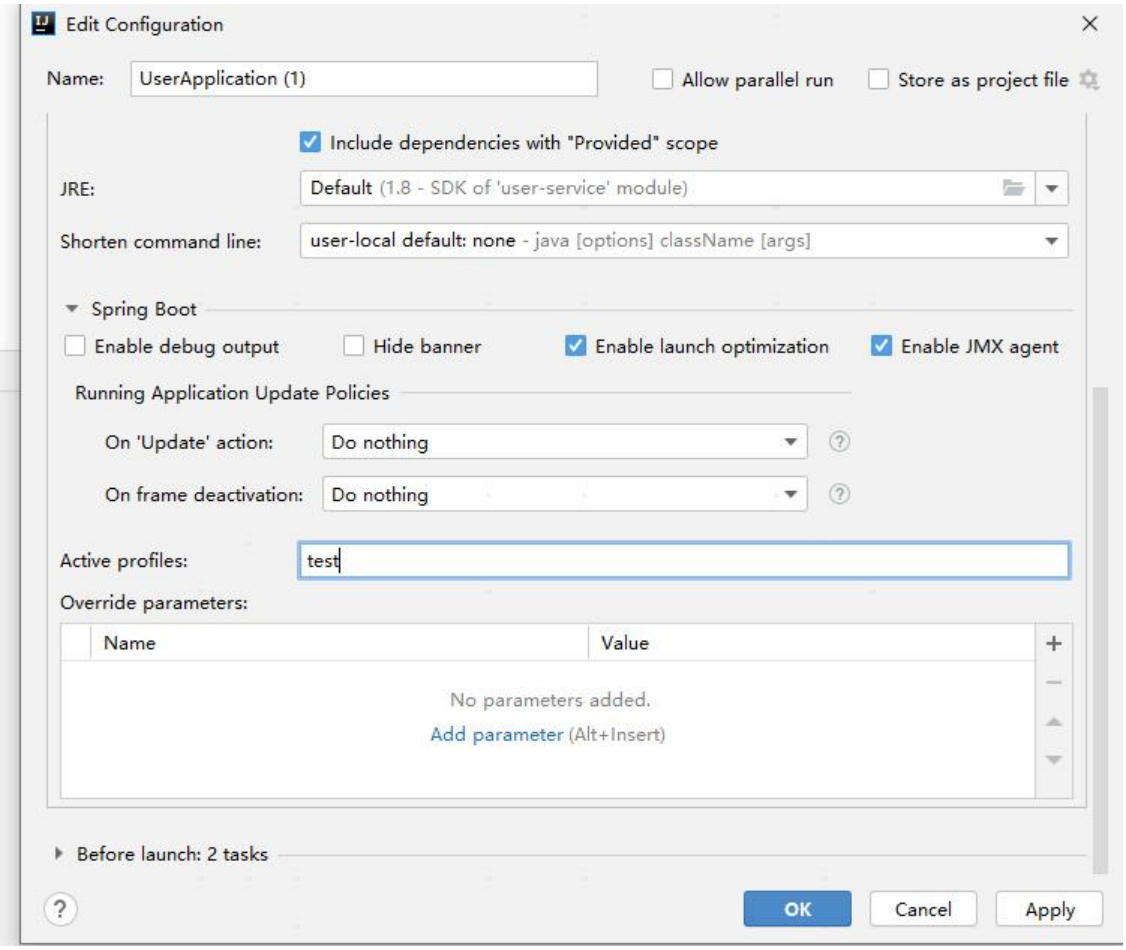
在user-service服务中，修改PatternProperties类，读取新添加的属性，然后修改UserController，加一个方法

```
public class PatternProperties {
    private String dateformat;
    private String envSharedValue;
}
```

```
@GetMapping("prop")
public PatternProperties properties()
{
    return properties;
}
```

1.3.3运行两个UserApplication，使用不同的profile

修改UserApplication2这个启动项，改变其profile值：



这样，UserApplication(8081)使用的profile是dev，UserApplication(1)(8082)使用的profile是test。

启动UserApplication和UserApplication2

访问http://localhost:8081/user/prop

和

http://localhost:8082/user/prop

结果可以看出来，不管是dev，还是test环境，都读取到了envSharedValue这个属性的值。

1.3.4配置共享的优先级

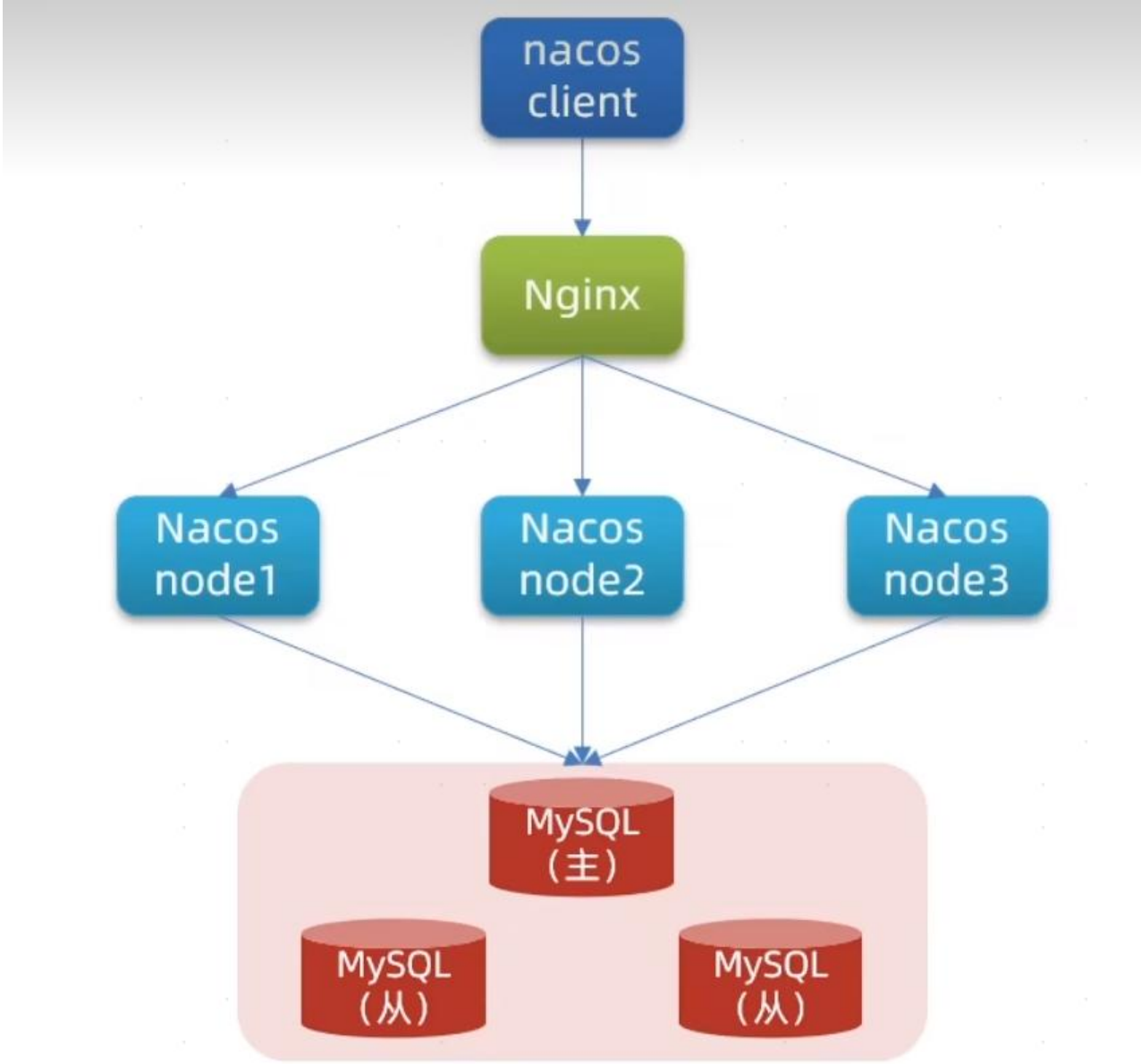
当nacos、服务本地同时出现相同属性时，优先级有高低之分：

- red_circle服务名-profile.yaml>服务名称.yaml>本地配置

1.4.搭建Nacos集群

1.4.1结构图

Nacos生产环境下一定要部署为集群状态，部署方式为：



1.4.2搭建集群

搭建集群的基本步骤：

- 搭建数据库，初始化数据库表结构
- 下载nacos安装包
- 配置nacos
- 启动nacos集群
- nginx反向代理

配置Nacos

进入nacos的conf目录，修改配置文件cluster.conf.example，重命名为cluster.conf：

然后添加内容（主机地址+端口）：

127.0.0.1:8845

```
127.0.0.1.8846
127.0.0.1.8847
```

然后修改application.properties文件，添加数据库配置

```
spring.datasource.platform=mysql

db.num=1

db.url.0=jdbc:mysql://127.0.0.1:3306/nacos?characterEncoding=utf8&connectTimeout=1000
socketTimeout=3000&autoReconnect=true&useUnicode=true&useSSL=false&serverTimezo
e=UTC
db.user.0=root
db.password.0=123
```

将nacos文件夹复制三份，分别命名为：nacos1、nacos2、nacos3，然后分别修改三个文件夹中的a
plication.properties，

nacos1:

```
server.port=8845
```

nacos2:

```
server.port=8846
```

nacos3:

```
server.port=8847
```

然后分别启动三个nacos节点：

集群启动

```
startup.cmd
```

nginx反向代理

修改conf/nginx.conf文件，配置如下：

```
upstream nacos-cluster {
    server 127.0.0.1:8845;
    server 127.0.0.1:8846;
    server 127.0.0.1:8847;
}

server {
    listen       80;
    server_name  localhost;

    location /nacos {
        proxy_pass http://nacos-cluster;
    }
}
```

而后在浏览器访问：http://localhost/nacos即可。

代码中application.yml文件配置如下：

```
spring:
  cloud:
    nacos:
      server-addr: localhost:80 # Nacos地址
```

- 实际部署时，需要给做反向代理的nginx服务器设置一个域名，这样后续如果有服务器迁移nacos的户端也无需更改配置.
- Nacos的各个节点应该部署到多个不同服务器，做好容灾和隔离

Nacos总结

到此为止Nacos就全部结束了，下面要开始组件Feign的学习了