crazyhzm 265

Dubbo源码解析（三十六）集群——configurator

dubbo java 阅读约 13 分钟

集群——configurator

> 目标：介绍dubbo中集群的配置规则，介绍dubbo-cluster下configurator包的源码。

前言

向注册中心写入动态配置覆盖规则 。该功能通常由监控中心或治理中心的页面完成。在最新的2.7.0版本中有新的配置规则，我会在后续讲解2.7.0新特性的时候提到。这里还是根据旧版本中配置规则来讲解，可以参考官方文档：

> http://dubbo.apache.org/zh-cn...

源码分析

（一）AbstractConfigurator

该类实现了Configurator接口，是配置规则 抽象类，配置有两种方式，一种是没有时添加配置，这种暂时没有用到，另一种是覆盖配置。

1.configure

```
@Override
public URL configure(URL url) {
    if (configuratorUrl == null || configuratorUrl.getHost() == null
            || url == null || url.getHost() == null) {
        return url;
    }
    // If override url has port, means it is a provider address. We want to control a specific provider with this override url, it may take effect on the specific provider instance or on consumers holding this provider instance.
    // 如果覆盖url具有端口，则表示它是提供者地址。我们希望使用此覆盖URL控制特定提供程序，它可以在提供端生效 也可以在消费端生效。
    if (configuratorUrl.getPort() != 0) {
        if (url.getPort() == configuratorUrl.getPort()) {
            return configureIfMatch(url.getHost(), url);
        }
    } else {// override url don't have a port, means the ip override url specify is a consumer address or 0.0.0.0
        // 1.If it is a consumer ip address, the intention is to control a specific consumer instance, it must takes effect at the consumer side, any provider received this override url should ignore;
        // 2.If the ip is 0.0.0.0, this override url can be used on consumer, and also can be used on provider
        // 配置规则，URL 没有端口，意味着override 输入消费端地址 或者 0.0.0.0
        if (url.getParameter(Constants.SIDE_KEY, Constants.PROVIDER).equals(Constants.CONSUMER)) {
            // 如果它是一个消费者ip地址，目的是控制一个特定的消费者实例，它必须在消费者一方生效，任何提供者收到这个覆盖url应该忽略;
            return configureIfMatch(NetUtils.getLocalHost(), url);// NetUtils.getLocalHost is the ip address consumer registered to registry.
        } else if (url.getParameter(Constants.SIDE_KEY, Constants.CONSUMER).equals(Constants.PROVIDER)) {
```

该方法是规则配置到URL中，但是关键逻辑在configureIfMatch方法中。

2.configureIfMatch

```
private URL configureIfMatch(String host, URL url) {
    // 匹配 Host
    if (Constants.ANYHOST_VALUE.equals(configuratorUrl.getHost()) || host.equals(configuratorUrl.getHost())) {
        String configApplication = configuratorUrl.getParameter(Constants.APPLICATION_KEY,
                configuratorUrl.getUsername());
        String currentApplication = url.getParameter(Constants.APPLICATION_KEY, url.getUsername());
        // 匹配 "application"
        if (configApplication == null || Constants.ANY_VALUE.equals(configApplication)
                || configApplication.equals(currentApplication)) {
            Set<String> conditionKeys = new HashSet<String>();
            // 配置 URL 中的条件 KEYS 集合。其中下面四个 KEY ，不算是条件，而是内置属性。考虑到下面要移除，所以添加到该集
合中。
            conditionKeys.add(Constants.CATEGORY_KEY);
            conditionKeys.add(Constants.CHECK_KEY);
            conditionKeys.add(Constants.DYNAMIC_KEY);
            conditionKeys.add(Constants.ENABLED_KEY);
            // 判断传入的 url 是否匹配配置规则 URL 的条件。
            for (Map.Entry<String, String> entry : configuratorUrl.getParameters().entrySet()) {
                String key = entry.getKey();
                String value = entry.getValue();
                // 除了 "application" 和 "side" 之外，带有 `"~"` 开头的 KEY ，也是条件。
                if (key.startsWith("~") || Constants.APPLICATION_KEY.equals(key) ||
Constants.SIDE_KEY.equals(key)) {
                    // 添加搭配条件集合
                    conditionKeys.add(key);
                    if (value != null && !Constants.ANY_VALUE.equals(value)
```

该方法是当条件匹配时，才对url进行配置。

3.compareTo

```
@Override
public int compareTo(Configurator o) {
    if (o == null) {
        return -1;
    }

    // // host 升序
    int ipCompare = getUrl().getHost().compareTo(o.getUrl().getHost());
    // 如果host相同，则根据priority降序来对比
    if (ipCompare == 0) {//host is the same, sort by priority
        int i = getUrl().getParameter(Constants.PRIORITY_KEY, 0),
                j = o.getUrl().getParameter(Constants.PRIORITY_KEY, 0);
        return i < j ? -1 : (i == j ? 0 : 1);
    } else {
        return ipCompare;
    }


}
```

这是配置的排序策略。先根据host升序，如果相同，再通过priority降序。

（二）AbsentConfigurator

```
public class AbsentConfigurator extends AbstractConfigurator {

    public AbsentConfigurator(URL url) {
        super(url);
    }

    @Override
    public URL doConfigure(URL currentUrl, URL configUrl) {
        // 当不存在时添加
        return currentUrl.addParametersIfAbsent(configUrl.getParameters());
    }

}
```

该配置方式就是当配置不存在的时候添加。

(三) AbsentConfiguratorFactory

```
public class AbsentConfiguratorFactory implements ConfiguratorFactory {

    @Override
    public Configurator getConfigurator(URL url) {
        // 创建一个AbsentConfigurator。
        return new AbsentConfigurator(url);
    }

}
```

该类是不存在时添加配置的工厂类，用来创建AbsentConfigurator。

(四) OverrideConfigurator

```
public class OverrideConfigurator extends AbstractConfigurator {

    public OverrideConfigurator(URL url) {
        super(url);
    }

    @Override
    public URL doConfigure(URL currentUrl, URL configUrl) {
        // 覆盖添加
        return currentUrl.addParameters(configUrl.getParameters());
    }

}
```

这种是覆盖添加。是目前在用的配置方式。

(五) OverrideConfiguratorFactory

```
public class OverrideConfiguratorFactory implements ConfiguratorFactory {

    @Override
    public Configurator getConfigurator(URL url) {
        // 创建OverrideConfigurator
        return new OverrideConfigurator(url);
    }

}
```

该类是OverrideConfigurator的工厂类，用来提供OverrideConfigurator实例。

后记

该部分相关的源码解析地址：https://github.com/CrazyHZM/i...

该文章讲解了集群中关于configurator实现的部分，讲了两种配置方式，分别是不存在再添加和覆盖添加。接下来我将开始对集群模块关于Directory部分进行讲解。

crazyhzm

265

关注作者

0 条评论

得票 • 时间

撰写评论 ...

提交评论

推荐阅读

Retrofit源码分析三 源码分析

我们先来看一下Retrofit的常见使用方法：上面是Retrofit的最基本使用方法，当然现在使用最多的还是RxJava2+Retrofit搭配使用，...

BlackFlagBin • 阅读 9

Dubbo 源码分析 - 自适应拓展原理

我在上一篇文章中分析了Dubbo的SPI机制，DubboSPI是Dubbo框架的核心。Dubbo中的很多拓展都是通过SPI机制进行加载的，比...

coolblog • 阅读 329

dubbo源码解析（二）Dubbo扩展机制SPI

前一篇文章《dubbo源码解析（一）Hello,Dubbo》是对dubbo整个项目大体的介绍，而从这篇文章开始，我将会从源码来解读dub...

CrazyHzm • 阅读 355

dubbo源码解析——概要篇

这次源码解析借鉴《肥朝》前辈的dubbo源码解析，进行源码学习。总结起来就是先总体,后局部.也就是先把需要注意的概念先抛...

• 阅读 631

结合Dubbo源码分析Spi

如前所说，DubboSPI的目的是获取一个指定实现类的对象。那么Dubbo是通过什么方式获取的呢？其实是调用ExtensionLoader.ge...

hnxydq • 阅读 15

Dubbo 源码分析 - SPI 机制

SPI全称为ServiceProviderInterface，是Java提供的一种服务发现机制。SPI的本质是将接口实现类的全限定名配置在文件中，并由服...

coolblog • 阅读 250

dubbo源码解析（十五）远程通信——Mina

ApacheMINA是一个网络应用程序框架，可帮助用户轻松开发高性能和高可扩展性的网络应用程序。它通过JavaNIO在各种传输（...

• 阅读 721