11 | 服务发布和引用的实践

2018-09-15 胡忠想

11 | 服务发布和引用的实践
朗读人：胡忠想　08′40″ | 3.98M

在专栏第 4 期，我给你讲解了服务发布和引用常见的三种方式：Restful API、XML 配置以及 IDL 文件。今天我将以 XML 配置方式为例，给你讲解服务发布和引用的具体实践以及可能会遇到的问题。

首先我们一起来看下 XML 配置方式，服务发布和引用的具体流程是什么样的。

XML 配置方式的服务发布和引用流程

1. 服务提供者定义接口

服务提供者发布服务之前首先要定义接口，声明接口名、传递参数以及返回值类型，然后把接口打包成 JAR 包发布出去。

比如下面这段代码，声明了接口 UserLastStatusService，包含两个方法 getLastStatusId 和 getLastStatusIds，传递参数一个是 long 值、一个是 long 数组，返回值一个是 long 值、一个是 map。

```
package com.weibo.api.common.status.service;


public interface UserLastStatusService {
     * @param uids
     * @return
     */
    public long getLastStatusId(long uid);


    /**
     *
     * @param uids
     * @return
     */
    public Map<Long, Long> getLastStatusIds(long[] uids);
}
```

2. 服务提供者发布接口

服务提供者发布的接口是通过在服务发布配置文件中定义接口来实现的。

下面我以一个具体的服务发布配置文件 user-last-status.xml 来给你讲解，它定义了要发布的接口 userLastStatusLocalService，对外暴露的协议是 Motan 协议，端口是 8882。并且针对两个方法 getLastStatusId 和 getLastStatusIds，通过 requestTimeout="300" 单独定义了超时时间是 300ms，通过 retries="0" 单独定义了调用失败后重试次数为 0，也就是不重试。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:aop="http://www.springframework.org/schema/aop"
     xsi:schemaLocation="http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-2.5.xsd
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-be
http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.
">


   <motan:service ref="userLastStatusLocalService"
```

```
            requestTimeout="50" retries="2"    interface="com.weibo.api.common.status.service.Use
            basicService="serviceBasicConfig" export="motan:8882">
   <motan:method name="getLastStatusId" requestTimeout="300"
              retries="0" />
   <motan:method name="getLastStatusIds" requestTimeout="300"
              retries="0" />
</motan:service>


</beans>
```

然后服务发布者在进程启动的时候，会加载配置文件 user-last-status.xml，把接口对外暴露出去。

3. 服务消费者引用接口

服务消费者引用接口是通过在服务引用配置文件中定义要引用的接口，并把包含接口定义的 JAR 包引入到代码依赖中。

下面我再以一个具体的服务引用配置文件 user-last-status-client.xml 来给你讲解，它定义服务消费者引用了接口 commonUserLastStatusService，接口通信协议是 Motan。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:aop="http://www.springframework.org/schema/aop"
     xsi:schemaLocation="http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-2.5.xsd
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-be
http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.
">
   <motan:protocol name="motan" default="true" loadbalance="${service.loadbalance.name}" />
<motan:basicReferer id="userLastStatusServiceClientBasicConfig"
               protocol="motan"  />


<!-- 导出接口 -->
<motan:referer id="commonUserLastStatusService" interface="com.weibo.api.common.status.service.Us
            basicReferer="userLastStatusServiceClientBasicConfig" />
```

```
</beans>
```

然后服务消费者在进程启动时，会加载配置文件 user-last-status-client.xml 来完成服务引用。

上面所讲的服务发布和引用流程看似比较简单，但在实际使用过程中，还是有很多坑的，比如在实际项目中经常会遇到这个问题：一个服务包含了多个接口，可能有上行接口也可能有下行接口，每个接口都有超时控制以及是否重试等配置，如果有多个服务消费者引用这个服务，是不是每个服务消费者都必须在服务引用配置文件中定义？

你可以先思考一下这个问题，联系自己的实践经验，是否有理想的解决方案呢？

服务发布和引用的那些坑

根据我的项目经验，在一个服务被多个服务消费者引用的情况下，由于业务经验的参差不齐，可能不同的服务消费者对服务的认知水平不一，比如某个服务可能调用超时了，最好可以重试来提供调用成功率。但可能有的服务消费者会忽视这一点，并没有在服务引用配置文件中配置接口调用超时重试的次数，**因此最好是可以在服务发布的配置文件中预定义好类似超时重试次数**，即使服务消费者没有在服务引用配置文件中定义，也能继承服务提供者的定义。这就是下面要讲的服务发布预定义配置。

1. 服务发布预定义配置

以下面的服务发布配置文件 server.xml 为例，它提供了一个服务 contentSliceRPCService，并且明确了其中三个方法的调用超时时间为 500ms 以及超时重试次数为 3。

```
<motan:service ref="contentSliceRPCService"       interface="cn.sina.api.data.service.ContentSlic
            basicService="serviceBasicConfig" export="motan:8882" >
   <motan:method name="saveContent" requestTimeout="500"
               retries="3" />
   <motan:method name="deleteContent" requestTimeout="500"
               retries="3" />
   <motan:method name="updateContent" requestTimeout="500"
               retries="3" />
</motan:service>
```

假设服务引用的配置文件 client.xml 的内容如下，那么服务消费者就会默认继承服务发布配置文件中设置的方法调用的超时时间以及超时重试次数。

```
<motan:referer id="contentSliceRPCService" interface="cn.sina.api.data.service.ContentSliceRPCSer
</motan:referer>
```

通过服务发布预定义配置可以解决多个服务消费者引用服务可能带来的配置复杂的问题，这样是不是最优的解决方案呢？

实际上我还遇到过另外一种极端情况，一个服务提供者发布的服务有上百个方法，并且每个方法都有各自的超时时间、重试次数等信息。服务消费者引用服务时，完全继承了服务发布预定义的各项配置。这种情况下，服务提供者所发布服务的详细配置信息都需要存储在注册中心中，这样服务消费者才能在实际引用时从服务发布预定义配置中继承各种配置。

这里就存在一种风险，当服务提供者发生节点变更，尤其是在网络频繁抖动的情况下，所有的服务消费者都会从注册中心拉取最新的服务节点信息，就包括了服务发布配置中预定的各项接口信息，这个信息不加限制的话可能达到 1M 以上，如果同时有上百个服务消费者从注册中心拉取服务节点信息，在注册中心机器部署为百兆带宽的情况下，很有可能会导致网络带宽打满的情况发生。

面对这种情况，最好的办法是把服务发布端的详细服务配置信息转移到服务引用端，这样的话注册中心中就不需要存储服务提供者发布的详细服务配置信息了。这就是下面要讲的服务引用定义配置。

2. 服务引用定义配置

以下面的服务发布配置文件为例，它详细定义了服务 userInfoService 的各个方法的配置信息，比如超时时间和重试次数等。

```
<motan:service ref="userInfoService" requestTimeout="50" retries="2"                    interface=
<motan:method name="addUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="updateUserPortrait" requestTimeout="300" retries="0"/>
    <motan:method name="modifyUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="addUserTags" requestTimeout="300" retries="0"/>
    <motan:method name="delUserTags" requestTimeout="300" retries="0"/>
    <motan:method name="processUserCacheByNewMyTriggerQ" requestTimeout="300" retries="0"/>
    <motan:method name="modifyObjectUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="addObjectUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="updateObjectUserPortrait" requestTimeout="300" retries="0"/>
    <motan:method name="updateObjectManager" requestTimeout="300" retries="0"/>
    <motan:method name="add" requestTimeout="300" retries="0"/>
```

```
    <motan:method name="deleteObjectManager" requestTimeout="300" retries="0"/>
    <motan:method name="getUserAttr" requestTimeout="300" retries="1" />
    <motan:method name="getUserAttrList" requestTimeout="300" retries="1" />
    <motan:method name="getAllUserAttr" requestTimeout="300" retries="1" />
    <motan:method name="getUserAttr2" requestTimeout="300" retries="1" />

    </motan:service>
```

可以像下面一样，把服务 userInfoService 的详细配置信息转移到服务引用配置文件中。

```
<motan:referer id="userInfoService" interface="cn.sina.api.user.service.UserInfoService" basicRef
    <motan:method name="addUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="updateUserPortrait" requestTimeout="300" retries="0"/>
    <motan:method name="modifyUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="addUserTags" requestTimeout="300" retries="0"/>
    <motan:method name="delUserTags" requestTimeout="300" retries="0"/>
    <motan:method name="processUserCacheByNewMyTriggerQ" requestTimeout="300" retries="0"/>
    <motan:method name="modifyObjectUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="addObjectUserInfo" requestTimeout="300" retries="0"/>
    <motan:method name="updateObjectUserPortrait" requestTimeout="300" retries="0"/>
    <motan:method name="updateObjectManager" requestTimeout="300" retries="0"/>
    <motan:method name="add" requestTimeout="300" retries="0"/>
    <motan:method name="deleteObjectManager" requestTimeout="300" retries="0"/>
    <motan:method name="getUserAttr" requestTimeout="300" retries="1" />
    <motan:method name="getUserAttrList" requestTimeout="300" retries="1" />
    <motan:method name="getAllUserAttr" requestTimeout="300" retries="1" />
    <motan:method name="getUserAttr2" requestTimeout="300" retries="1" />
</motan:referer>
```

这样的话，服务发布配置文件可以简化为下面这段代码，是不是信息精简了许多。

```
<motan:service ref="userInfoService" requestTimeout="50" retries="2"                    interface=
    </motan:service>
```

在进行类似的服务详细信息配置，由服务发布配置文件迁移到服务引用配置文件的过程时，尤其要注意迁移步骤问题，这就是接下来我要给你讲的服务配置升级问题。

3. 服务配置升级

实际项目中，我就经历过一次服务配置升级的过程。由于引用服务的服务消费者众多，并且涉及多个部门，升级步骤就显得异常重要，通常可以按照下面步骤操作。

- 各个服务消费者在服务引用配置文件中添加服务详细信息。
- 服务提供者升级两台服务器，在服务发布配置文件中删除服务详细信息，并观察是否所有的服务消费者引用时都包含服务详细信息。
- 如果都包含，说明所有服务消费者均完成升级，那么服务提供者就可以删除服务发布配置中的服务详细信息。
- 如果有不包含服务详细信息的服务消费者，排查出相应的业务方进行升级，直至所有业务方完成升级。

总结

今天我给你介绍了 XML 配置方式的服务发布和引用的具体流程，简单来说就是服务提供者定义好接口，并且在服务发布配置文件中配置要发布的接口名，在进程启动时加载服务发布配置文件就可以对外提供服务了。而服务消费者通过在服务引用配置文件中定义相同的接口名，并且在服务引用配置文件中配置要引用的接口名，在进程启动时加载服务引用配置文件就可以引用服务了。

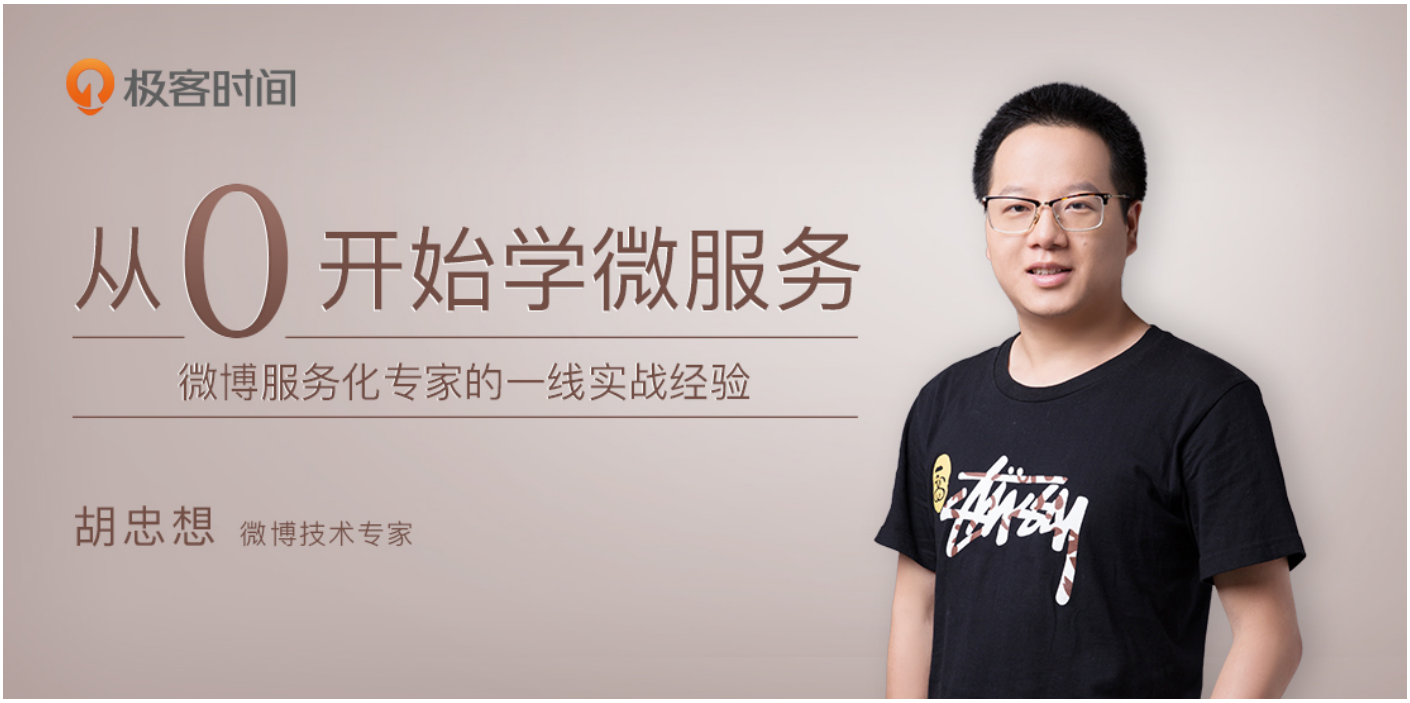
在业务具体实践过程中可能会遇到引用服务的服务消费者众多，对业务的敏感度参差不齐的问题，所以在服务发布的时候，最好预定义好接口的各种配置。在服务规模不大，业务比较简单的时候，这样做比较合适。但是对于复杂业务，虽然服务发布时预定义好接口的各种配置，但在引用的服务消费者众多且同时访问的时候，可能会引起网络风暴。这种情况下，比较保险的方式是，把接口的各种配置放在服务引用配置文件里。

在进行服务配置升级过程时，要考虑好步骤，在所有服务消费者完成升级之前，服务提供者还不能把服务的详细信息去掉，否则可能会导致没有升级的服务消费者引用异常。

思考题

如果你在实际项目中采用过 XML 配置的服务发布和应用方式，是否还遇到过其他问题？你是如何解决的呢？

欢迎你在留言区写下自己的思考，与我一起讨论。



精选留言

echo _ 陈 2

遇到过版本变更时序列化兼容问题

我们用的dubbo，经常会出现，某个dubbo接口的API升级：包含新增方法，或者某个方法的入参或者返回值新增字段。

我们的服务提供者更新消费者并不是一定要更新，如果我的api改动没有改动某个消费者调用的方法或者那个消费者可以兼容提供者的改动，那么消费者是可以不升级的。也就是允许系统中存在：服务提供者依赖的api是1.1版本，而服务消费者依赖的的api的jar包是1.0版本……这样的情况。

以前用hessian2做序列化方式，服务提供者单方面引用新版本api，老的消费者一样能正常调用。可是有同事听说FST序列化更快更强……于是某些接口改动了序列化方式为FST……发现这时依赖老版本api的服务都异常了……

经验：性能是一方面，但也要考虑业务兼容性

2018-09-15

作者回复

是的，兼容性很重要

2018-09-15

张小小的席大da 1

目前只会spring cloud ，dubbo没有细研究过 只是看dubbo官网会使用，觉得体系都差不多 没有太多机会去实战

2018-09-17

Mrsong 0

多个服务消费者调用了服务提供者A，如果服务提供者A的接口参数发生变化，那所有消费者都需要变更，是否有好的解决方案

呢？
2018-09-20

莲花 0
dubbo本身就会有provider和consumer的xml定义配置。这个和引用文件什么关系？没怎么明白
2018-09-19

张振华 0
Spring. cloud.呢
2018-09-19

少帅 0
注解貌似没有找到方法级别的配置
2018-09-15