黑客派

springboot 整合 dubbox

作者：TheNow

原文链接：https://hacpai.com/article/1480817291800

来源网站：黑客派

简介

今天咱们来看看怎么利用Spring Boot整合Dubbox来开发去中心化的微服务。

系统环境

本文基于Jdk1.8/Maven 3.3.9/Spring Boot 1.4.2.RELEASE/Dubbo 2.8.5.SNAPSHOT(Dubbox后续源版本)/ZooKeeper3.4.8

Zookeeper环境搭建

下载并安装启动

下载

解压

修改配置文件

好吧 我们不改了,我们使用默认的配置.哈!

启动

Dubbox环境准备

zookeeper准备好了,先放着 一会再用.下面我们来准备下Dubbox.

dubbox是当当网基于dubbo开源的组件

为什么使用dubbox?

因为dubbox支持更新的spring版本...

Dubbox在maven中央仓库并没有对应的依赖,所以我们需要自己动手将其发布到我们的本地仓库来使.

下载

我们这次从码云下载

编译安装

等待 ... 等待...

之后我们在我们的maven本地仓库/com/alibaba/dubbo/2.8.5-SNAPSHOT中会发现这么一个东西:

dubbo-2.8.5-SNAPSHOT.jar

这个玩意就是我们需要的Dubbx的jar包...

Spring Boot Dubbo引导

dubbox的jar包准备好了,行,咱先放着.一会再用.

下面来介绍下spring-boot-starter-dubbo项目的准备.

我们可能以前在使用dubbo的时候都是用的xml配置.在整合Spring Boot的时候呢是用的@ImportResource注解来引入的dubbo的xml配置.

但是Spring Boot本身并不推荐xml配置.怎么解决这个矛盾,我们可以自己准备一个Spring Boot Starter dubbo的项目来引导Spring Boot对Dubbo的自动化配置.

下载

感谢这位悲伤的大神的开源贡献!. 顶一个,赞两个.

修改pom.xml

- 在环境准备的时候我们说过,我们的项目基于Spring Boot 1.4.2.RELEASE ,但是我们down下来sprin-boot-starter-dubbo的时候发现它用的是1.3.6的版本.我们动手自己改下parent的依赖吧.
- 我们发现在pom.xml中基本所有的依赖的 option都是true,我懒所以我想在其他项目依赖这个项的时候不要再写一遍,所以我把<option>true</option>都给干掉了..... 干不干掉这个倒是随意哈.
- 刚上一个环节我们打包并安装到本地库的Dubbox在这个地方需要用上了. 我们修改 dubbo的版为2.8.5-SNAPSHOT
- 修改java版本为1.8

完整pom.xml文件如下:

编译打包

上面我们对于spring-boot-starter-dubbo的准备工作完成,我们现在打包编译

然后我们去maven本地仓库中找到它:

xxx/org/springframework/boot/spring-boot-starter-dubbo/1.4.2.RELEASE/spring-boot-starter-dubbo-1.4.2.RELEASE.jar

dubbo系统监控工具

这里我们使用韩都衣舍马老师提供的dubbo-monitor

下载

运行

根据项目README.MD我们先创建一个叫monitor的数据库,然后maven打包运行,我们也可以导入到ID中直接运行,当然生产环境我们不能这么干.

我们的home页面:

一会儿我们需要在这里验证我们的provider和consumer是否已经成功.

基本工作已经准备妥当,我们来看下我们怎么使用它.

目录结构

首先我们来看一下整个maven项目的目录结构:

说明

- business作为父项目
- consumer是我们的服务消费者
- provider是我们的服务提供者
- service是提供domain和接口service的项目

为什么要单独把service建module呢?

因为我们写的service(java interface)和domain(java bean)是需要在consumer和provider端共享的.

单独打成jar包有利用我们的代码重用和序列化反序列化.

基本结构介绍完成,下面我们分每一个模块来详细探讨.

business 父项目

既然business作为maven父项目,就做点它应该干的事.

pom.xml

在其中我们引入spring-boot-starter-parent spring-boot-starter-web spring-boot-starter-test spring-boot-starter-dubbo,其中除了spring-boot-starter-parent 我们定义其他依赖都可选.

没了...

service子项目

service子项目提供domain和service接口.

pom.xml

定义一下parent和artifactId完事.

BusinessDomain.java

定义我们需要在provider和consumer中使用的domain,实现java.io.Serializable来进行序列化.

dubbo支持的序列化方式很多,这个可以参考dubbo.io里关于协议和序列化的介绍,我们使用默认的协议ubbo.

BusinessService.java

定义我们需要在provider和consumer中使用的接口方法.

provider子项目

我们的服务提供者.

pom.xml

定义parent和artifactId并引入spring-boot-starter-web spring-boot-starter-dubbo service依赖.

application.yml

定义我们的dubbo配置.

服务注册发现使用zookeeper.协议使用dubbo,包扫描路径写cn.veryjava.business.provider

ProviderApplication.java

这是我们服务提供者的引导类.重点是@EnableDubboAutoConfiguration 这个注解将引导我们自动配置dubbox

BusinessServiceImpl.java

这个是我们需要提供的服务,重点是@Service这个注解,需要注意的是此@Service非彼@Service.

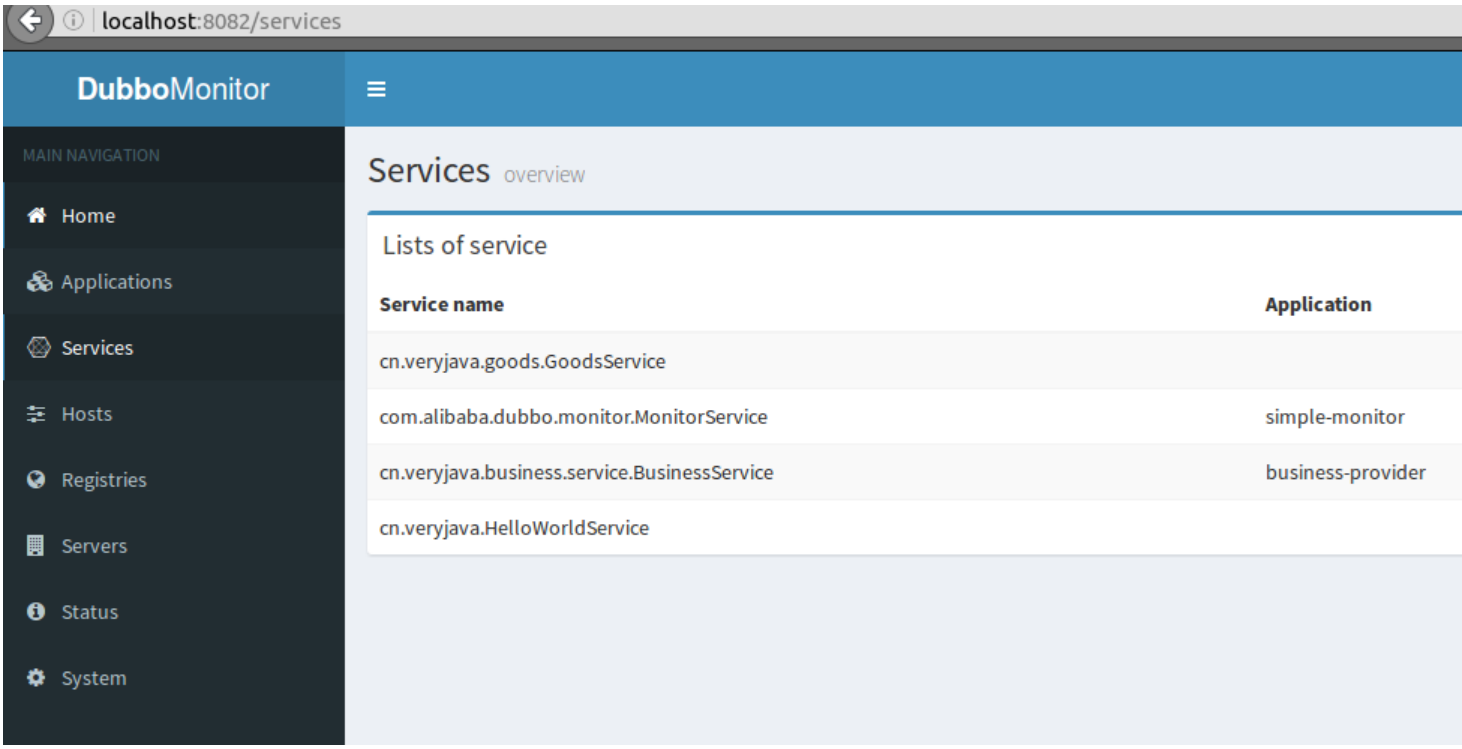
我们在这个地方使用的@Service是dubbo提供的,注意看import部分.然后,dubbo的springBoot自动配置会自动发现这个类并将其注册到zookeeper.

当然我们使用spring提供的@Service也是可以的,不过这种方式比较麻烦.这个地方我们就不介绍了,有了解的同学可以去dubbo.io去详细了解

编译运行

代码写好了,服务提供了,我们来验证下我们提供的服务是否能够成功注册并被发现.

启动后我们打开dubbo-monitor的Services页面,如果看到如下情况,则证明我们的服务已经注册成功:



注意观察其中cn.veryjava.business.service.BusinessService我们发现这个时候的BusinessService已被提供但是还没有相应的消费者来使用.那么我们接下来看一下消费者怎么去使用.

consumer子项目

这个是我们的服务消费者

pom.xml

定义parent artifactId并引入spring-boot-starter-web spring-boot-starter-dubbo service依赖.

application.yml

实测 不写scan不行,可能是我刚开始理解有问题....我刚开始以为scan只是用来进行服务发现的,结果跟费者进行消费也有关系...

ConsumerApplication.java

服务消费者引导类,@EnableDubboAutoConfiguration注解也得写,原因同上.

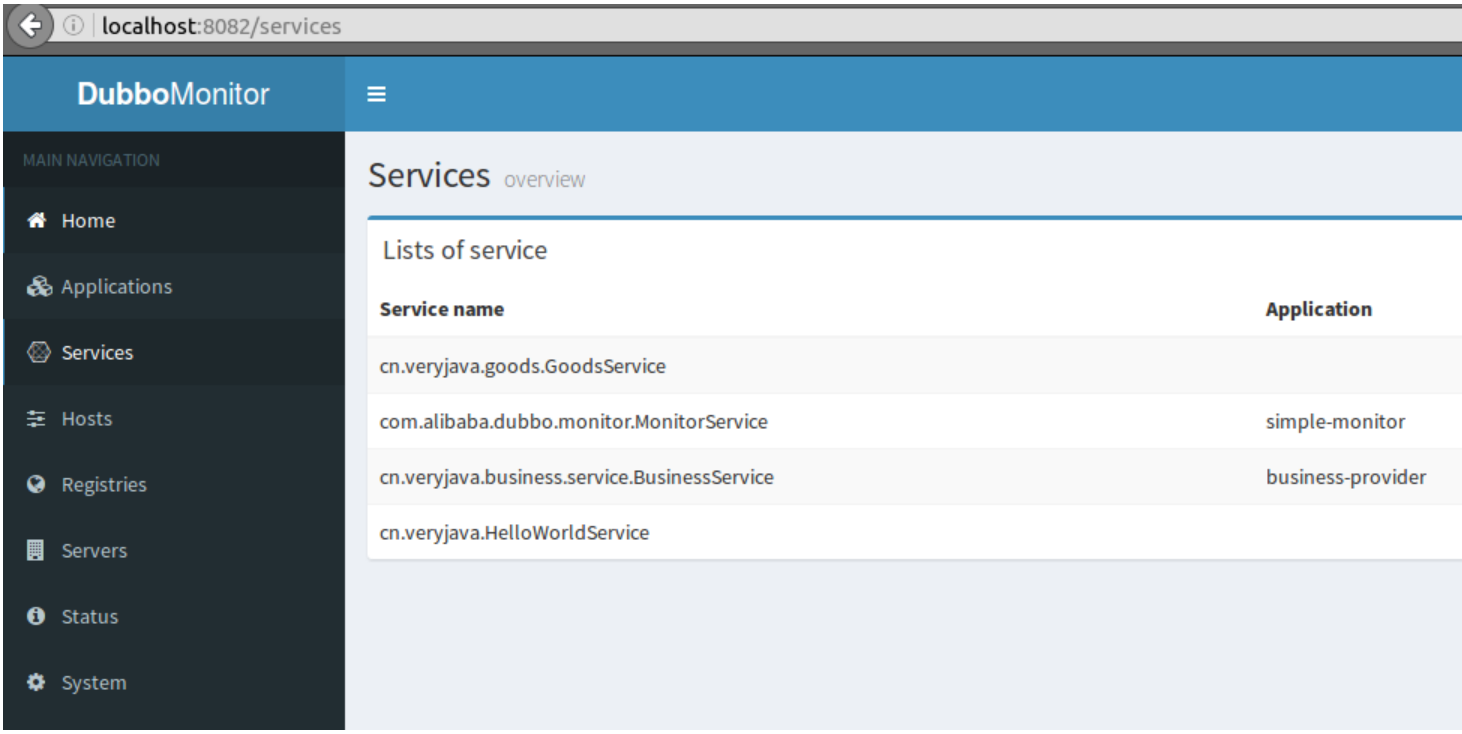
BusinessConsumerController.java

我们的BusinessService服务是怎么被消费的,怎么被依赖的.其实就是使用了dubbo提供的@Reference解... 告诉dubbo我要使用哪个版本服务,就是这么简单....

编译运行

我们来测一下dubbo-monitor能否监控到服务的消费者吧.

启动后我们打开dubbo-monitor的Services页面,如果看到如下情况,则证明我们的服务已经注册成功且消费者已经能够发现:



然后我们调用一下这个接口,看看到底是不是我们想要的数据.

输出如下:

好吧,到这一步,我们的服务发现和服务消费都可以成功了.

总结

我们发现,dubbo的使用还是很简单的,几乎没有任何的侵入性,也非常符合Spring的IOC/DI的理论概念可以说跟spring的结合非常完美!

我们的这个小项目,仅仅只是用来学习的小项目,不过我们可以在此基础上对zookeeper,对各个provider consumer进行集群配置.这样我们就可以慢慢实现后台服务的去中心化,很大程度上提高了我们架构的用性.

希望各位在Java的路上越走越好.!

代码

springboot整合dubbox的实例一枚

原文地址

springboot整合dubbox的实例一枚