SpringBoot 整合 dubbo

> Springboot 整合 dubbo 和以前的流程是一致,不过是没有了 xml 配置文件,改为了 springboot 的配置文件
>
> 大约流程:
>
> 1. 导入依赖
> 2. 编写中间接口
> 3. 编写提供者
> 4. 编写消费者
> 5. 测试

> 本案例是简单的入门配置,使用的是 IDEA 软件,采用的是一个 project 内部多个 module 的方式,所有的依赖添加在了 project 的 pom 文件中,请注意

一 Project 的 POM 文件

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.qianfeng</groupId>
  <artifactId>springboot-dubbo</artifactId>
  <version>1.0-SNAPSHOT</version>
    <!--声明为 springboot 项目-->
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.9.RELEASE</version>
  </parent>
  <modules>
    <module>springdubboprovider</module>
    <module>springdubbointerface</module>
    <module>springdubboconsumer</module>
  </modules>
  <packaging>pom</packaging>

  <name>springboot-dubbo</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
      <!--web 依赖-->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
      <!--dubbo 需要的健康兼容依赖-->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
      <!--dubbo 需要的快速启动依赖,如果无法下载,请配置下面的仓库-->
    <dependency>
      <groupId>com.alibaba.boot</groupId>
      <artifactId>dubbo-spring-boot-starter</artifactId>
      <version>1.0.0-SNAPSHOT</version>
    </dependency>
      <!--springboot 需要的依赖-->
    <dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
    </dependency>
    <dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-core</artifactId>
    </dependency>
    <!--zookeeper 客户端 -->
    <dependency>
      <groupId>com.101tec</groupId>
      <artifactId>zkclient</artifactId>
      <version>0.10</version>
    </dependency>

  </dependencies>
  <!--如果上面的依赖包无法下载,添加一下仓库地址-->
  <repositories>
    <repository>
      <id>sonatype-nexus-snapshots</id>
      <url>https://oss.sonatype.org/content/repositories/snapshots</url>
      <releases>
        <enabled>false</enabled>
      </releases>
      <snapshots>
```

```
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
    </project>
```

二 中间接口 DemoService

> 主要是用于提供者和接收者依赖用,在一个独立的 module 中,此module属于 project 的子项目

```
/**
 * Created by jackiechan on 2018/2/8/下午7:16
 * 提供者的统一接口
 */
public interface DemoService {
    /**
     * 测试方法,没有实际意义
     * @param name
     * @return
     */
    String getData(String name);
}
```

三 安装 service 到仓库

> 需要首先安装 project 才可以,因为要安装此 module 需要先安装父,所以最好先安装父,在安装当前 module

四 Provider提供者

> 用于提供具体的服务,内部中的类实现了上面的接口

4.1 pom.xml

> 只需要添加 service 的依赖即可

```
 <!--注意此依赖是上面的 service 对应的内容,实际开发请填写自己的具体的依赖-->
        <dependency>
            <groupId>com.qianfeng</groupId>
            <artifactId>spring-dubbo-interface</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
```

4.2 application.properties

> 此文件用于配置 springboot, 在 resources 目录中,内部其实配置的就是原先 xml 相关内容

```
# Spring boot application
spring.application.name = dubbo-provider-demo
server.port = 9090
management.port = 9091

#扫描 dubbo 的 service 注解
# Base packages to scan Dubbo Components (e.g @Service , @Reference)
dubbo.scan.basePackages  = com.qianfeng.springboot.service.impl


# Dubbo Config properties
## ApplicationConfig Bean
#在监控平台显示的程序的名字
dubbo.application.id = dubbo-provider
dubbo.application.name = dubbo-provider
#dubbo.application.qos.port=22222
#dubbo.application.qos.enable=true
#spring.dubbo.application.name 应用名称
#spring.dubbo.protocol.name 协议名称
#spring.dubbo.protocol.port 协议端口
#spring.dubbo.scan dubbo 服务类包目录
## ProtocolConfig Bean
# 相当于<dubbo:protocol  name="dubbo" port="33335"></dubbo:protocol>
dubbo.protocol.id = dubbo
dubbo.protocol.name = dubbo
dubbo.protocol.port = 33335
dubbo.protocol.status = server

## RegistryConfig Bean
#spring.dubbo.registry.address 注册中心地址
#<dubbo:registry address="47.95.244.39" port="2181" protocol="zookeeper">
</dubbo:registry>
dubbo.registry.id = my-registry
dubbo.registry.address =zookeeper://192.168.3.212:2181

# Dubbo Endpoint (default status is disable)
endpoints.dubbo.enabled = true

# Dubbo Health
## StatusChecker Name defaults (default : "memory", "load" )
management.health.dubbo.status.defaults = memory
## StatusChecker Name extras (default : empty )
management.health.dubbo.status.extras = load,threadpool
```

4.3 service 实现类 DemoServiceImpl

> 内部对业务做了具体实现,用于被调用

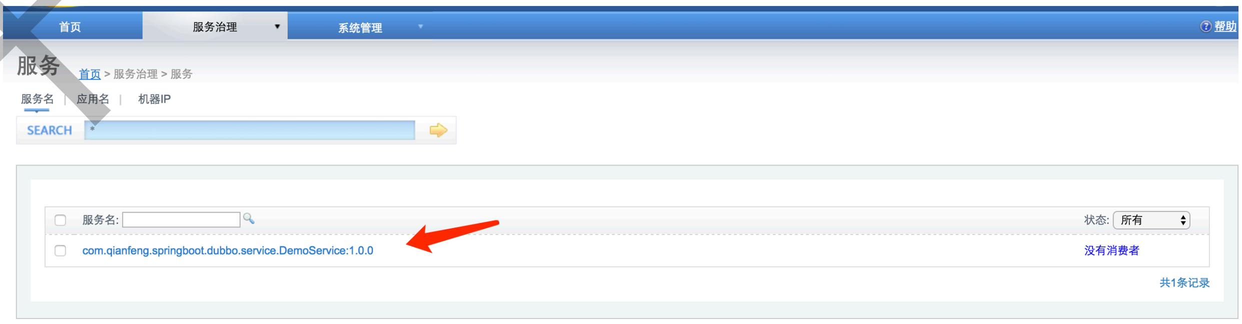
```
/**
 * Created by jackiechan on 2018/2/8/下午7:23
 */
@Service(//注意注解是 dubbo 的注解,不是 spring 的
        version = "1.0.0",
        application = "${dubbo.application.id}",//程序的 id
        protocol = "${dubbo.protocol.id}",// 协议的 id
        registry = "${dubbo.registry.id}"// 注册中心的 id
)
public class DemoServiceImpl implements DemoService {
    @Override
    public String getData(String name) {
        return "输入的内容是:" + name;
    }
}
```

4.3 SpringBoot 启动类StarProvider

```
/**
 * Created by jackiechan on 2018/2/8/下午7:24
 */
@SpringBootApplication
public class StarProvider {
    public static void main(String[] args) {
        SpringApplication.run(StarProvider.class, args);
    }
}
```

4.4 启动提供者,测试

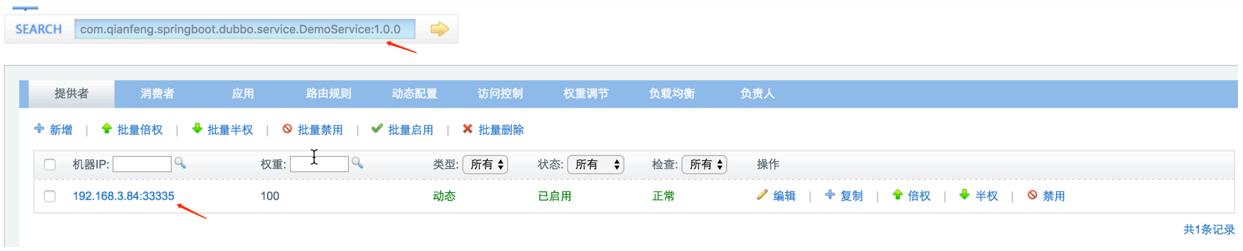
> 可以在 dubbo 的监控平台查看信息



服务名: 状态: 所有

com.qianfeng.springboot.dubbo.service.DemoService:1.0.0 没有消费者

共1条记录


SEARCH com.qianfeng.springboot.dubbo.service.DemoService:1.0.0

机器IP:	权重:	类型: 所有	状态: 所有	检查: 所有	操作
192.168.3.84:33335	100	动态	已启用	正常	编辑 | 复制 | 倍权 | 半权 | 禁用

共1条记录

五 Consumer消费者

5.1 pom文件

```
<!--注意此依赖是上面的 service 对应的内容,实际开发请填写自己的具体的依赖-->
       <dependency>
           <groupId>com.qianfeng</groupId>
           <artifactId>spring-dubbo-interface</artifactId>
           <version>1.0-SNAPSHOT</version>
       </dependency>
```

5.2 application.properties

> springboot 的配置文件,用于配置消费者相关信息,内部其实配置的就是原先 xml 相关内容

```
spring.application.name = dubbo-consumer-demo
server.port = 8080
management.port = 8081


# Dubbo Config properties
## ApplicationConfig Bean
dubbo.application.id = dubbo-consumer-demo
dubbo.application.name = dubbo-consumer-demo

## Legacy QOS Config
#dubbo.qos.port = 22223

## ProtocolConfig Bean
dubbo.protocol.id = dubbo
dubbo.protocol.name = dubbo
dubbo.protocol.port = 12345
# Dubbo Endpoint (default status is disable)
#endpoints.dubbo.enabled = true
#相当于<dubbo:registry address="47.95.244.39" port="2181" protocol="zookeeper">
</dubbo:registry>
dubbo.registry.id = my-registry
dubbo.registry.address =zookeeper://192.168.3.212:2181
# Dubbo Health
```

```
## StatusChecker Name defaults (default : "memory", "load" )
#management.health.dubbo.status.defaults = memory
```

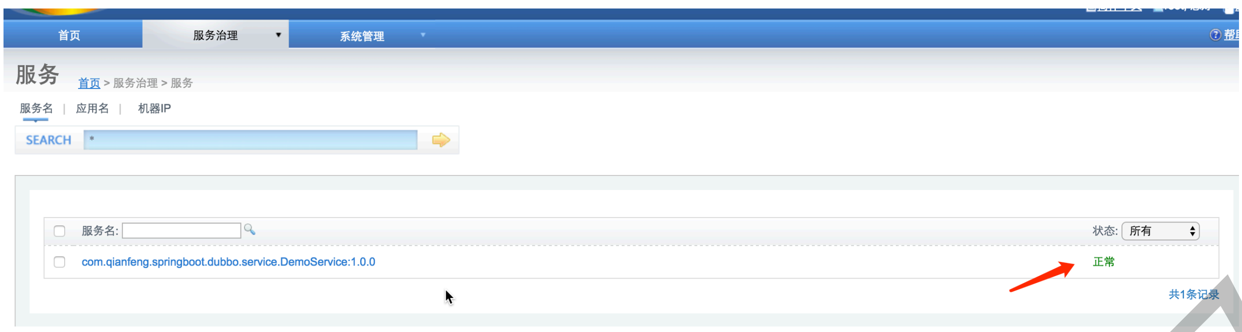
5.3 controller

```java

/**
 * Created by jackiechan on 2018/2/8/下午7:53
 */
@RestController
@RequestMapping("/test")
public class DemoController {
    @Reference(
            version = "1.0.0",
            application = "${dubbo.application.id}"


    )
    private DemoService demoService;
    @RequestMapping("/getdata/{name}")
    public String testData(@PathVariable("name") String name) {
        return demoService.getData(name);
    }
}
```

5.4 Springboot 启动类

```java
/**
 * Created by jackiechan on 2018/2/8/下午8:53
 */
@SpringBootApplication(scanBasePackages = {"com.qianfeng.controller"})
public class App
{
    public static void main( String[] args )
    {
        SpringApplication.run(App.class, args);
    }
}
```

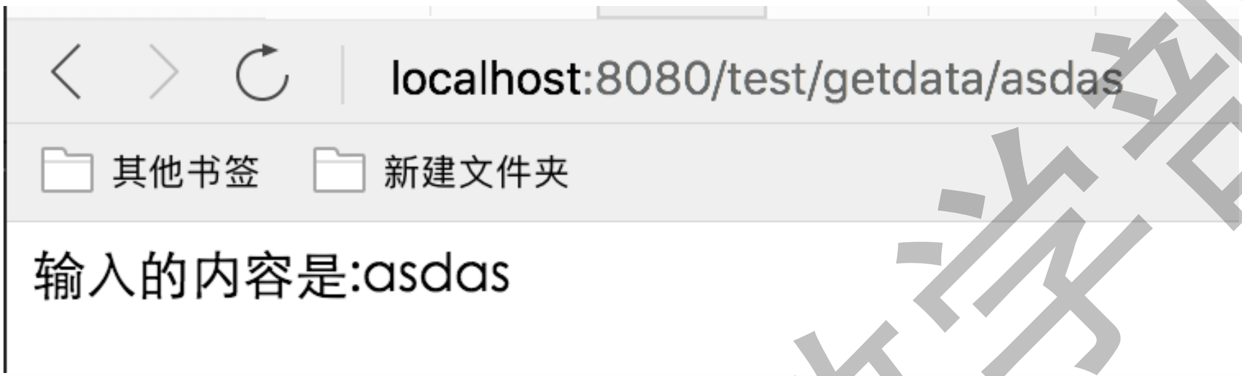
5.5 启动消费者测试



服务名: 状态: 所有

com.qianfeng.springboot.dubbo.service.DemoService:1.0.0 正常

共1条记录



localhost:8080/test/getdata/asdas

输入的内容是:asdas