# 狂神说SpringBoot17：Dubbo和Zookeeper集成

秦疆 狂神说 2020-03-28



## 分布式理论

### 什么是分布式系统？

在《分布式系统原理与范型》一书中有如下定义：“分布式系统是若干独立计算机的集合，这些计算机对于用户来说就像单个相关系统”；

分布式系统是由一组通过网络进行通信、为了完成共同的任务而协调工作的计算机节点组成的系统。分布式系统的出现是为了用廉价的、普通的机器完成单个计算机无法完成的计算、存储任务。其目的是**利用更多的机器，处理更多的数据**。

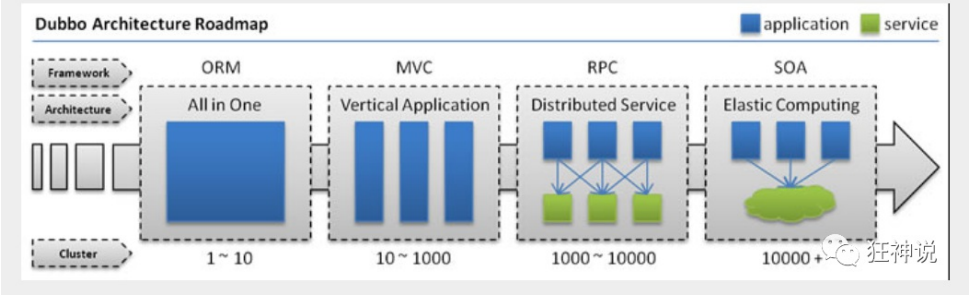
分布式系统（distributed system）是建立在网络之上的软件系统。

首先需要明确的是，只有当单个节点的处理能力无法满足日益增长的计算、存储任务的时候，且硬件的提升（加内存、加磁盘、使用更好的CPU）高昂到得不偿失的时候，应用程序也不能进一步优化的时候，我们才需要考虑分布式系统。因为，分布式系统要解决的问题本身就是和单机系统一样的，而由于分布式系统多节点、通过网络通信的拓扑结构，会引入很多单机系统没有的问题，为了解决这些问题又会引入更多的机制、协议，带来更多的问题。。。

### Dubbo文档

随着互联网的发展，网站应用的规模不断扩大，常规的垂直应用架构已无法应对，分布式服务架构以及流动计算架构势在必行，急需**一个治理系统**确保架构有条不紊的演进。

在Dubbo的官网文档有这样一张图




### 单一应用架构

当网站流量很小时，只需一个应用，将所有功能都部署在一起，以减少部署节点和成本。此时，用于简化增删改查工作量的数据访问框架(ORM)是关键。


适用于小型网站，小型管理系统，将所有功能都部署到一个功能里，简单易用。

**缺点：**

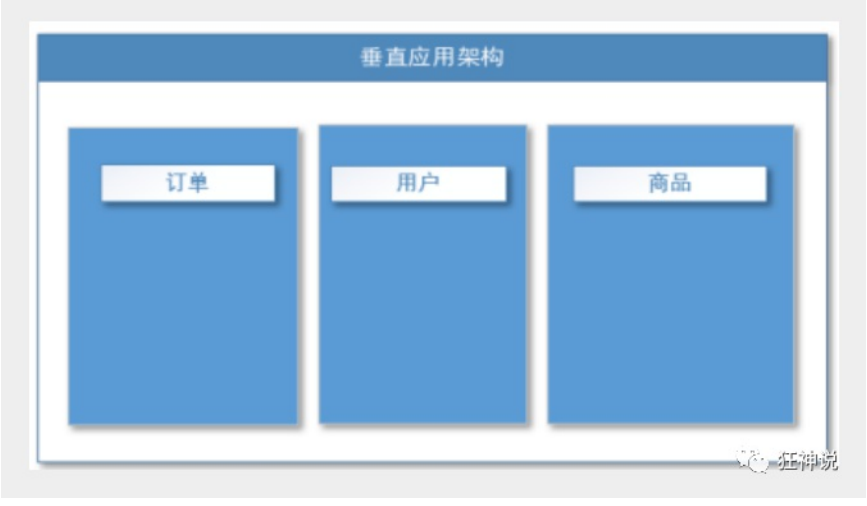
1、性能扩展比较难

2、协同开发问题

3、不利于升级维护

## 垂直应用架构

当访问量逐渐增大，单一应用增加机器带来的加速度越来越小，将应用拆成互不相干的几个应用，以提升效率。此时，用于加速前端页面开发的Web框架(MVC)是关键。

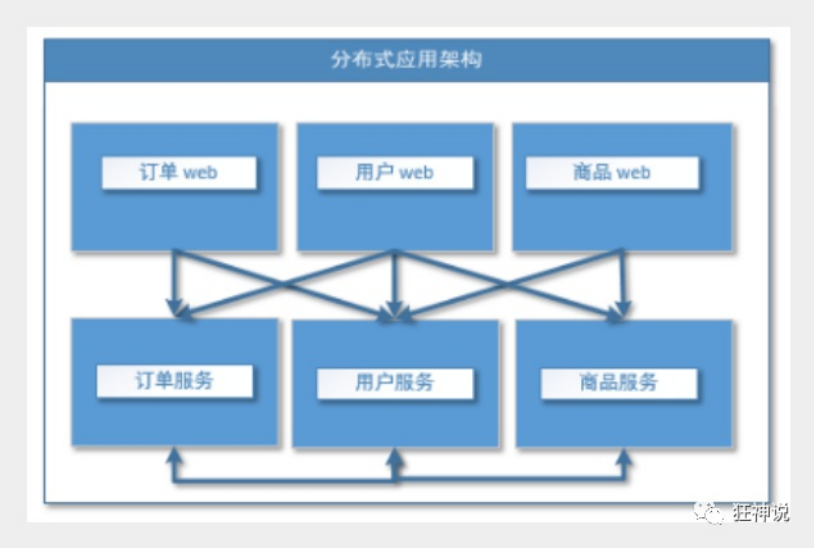


通过切分业务来实现各个模块独立部署，降低了维护和部署的难度，团队各司其职更易管理，性能扩展也更方便，更有针对性。

缺点：公用模块无法重复利用，开发性的浪费

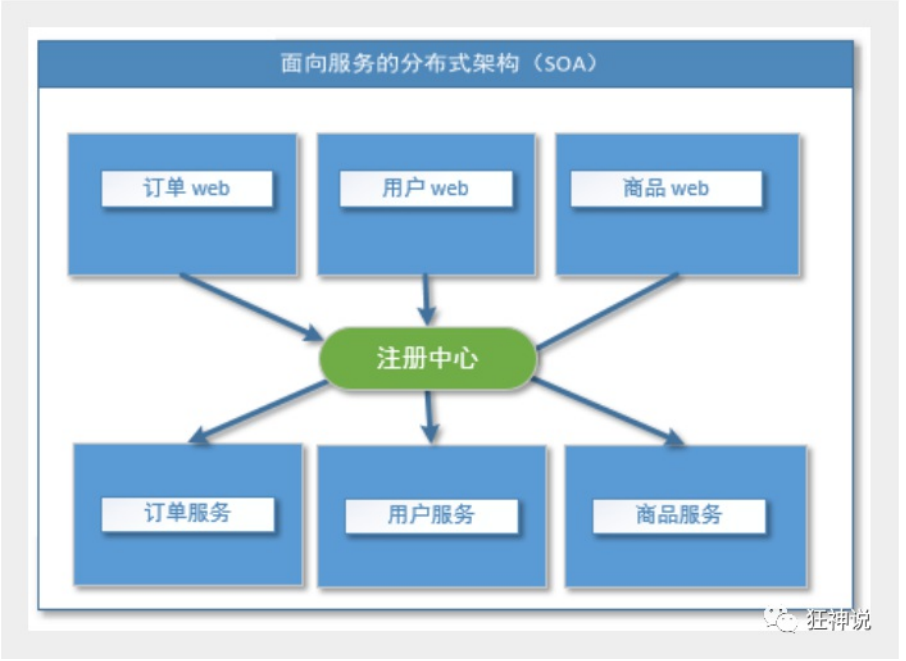
## 分布式服务架构

当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，使前端应用能更快速的响应多变的市场需求。此时，用于提高业务复用及整合的**分布式服务框架(RPC)**是关键。




## 流动计算架构

当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率。此时，用于**提高机器利用率的资源调度和治理中心**(SOA)[Service Oriented Architecture]是关键。



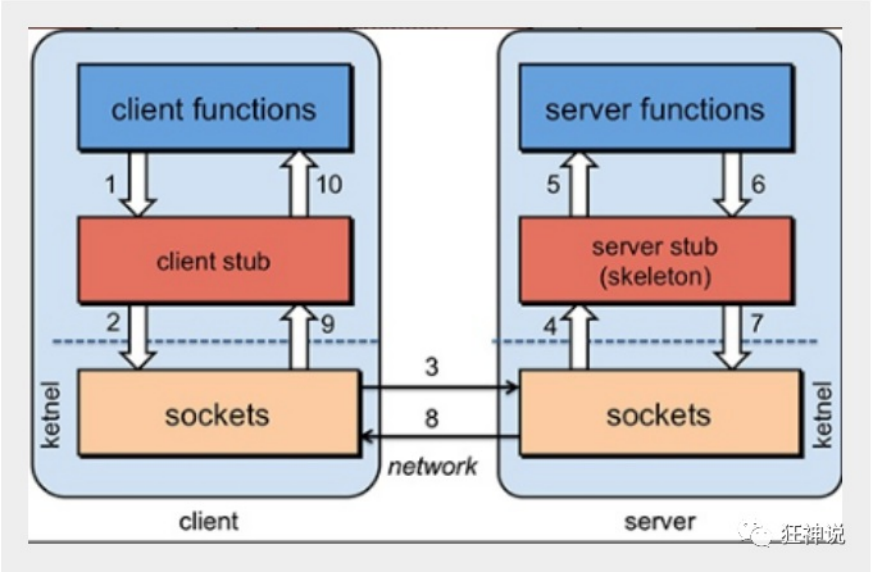

## 什么是RPC

RPC【Remote Procedure Call】是指远程过程调用，是一种进程间通信方式，他是一种技术的思想，而不是规范。它允许程序调用另一个地址空间（通常是共享网络的另一台机器上）的过程或函数，而不用程序员显式编码这个远程调用的细节。即程序员无论是调用本地的还是远程的函数，本质上编写的调用代码基本相同。

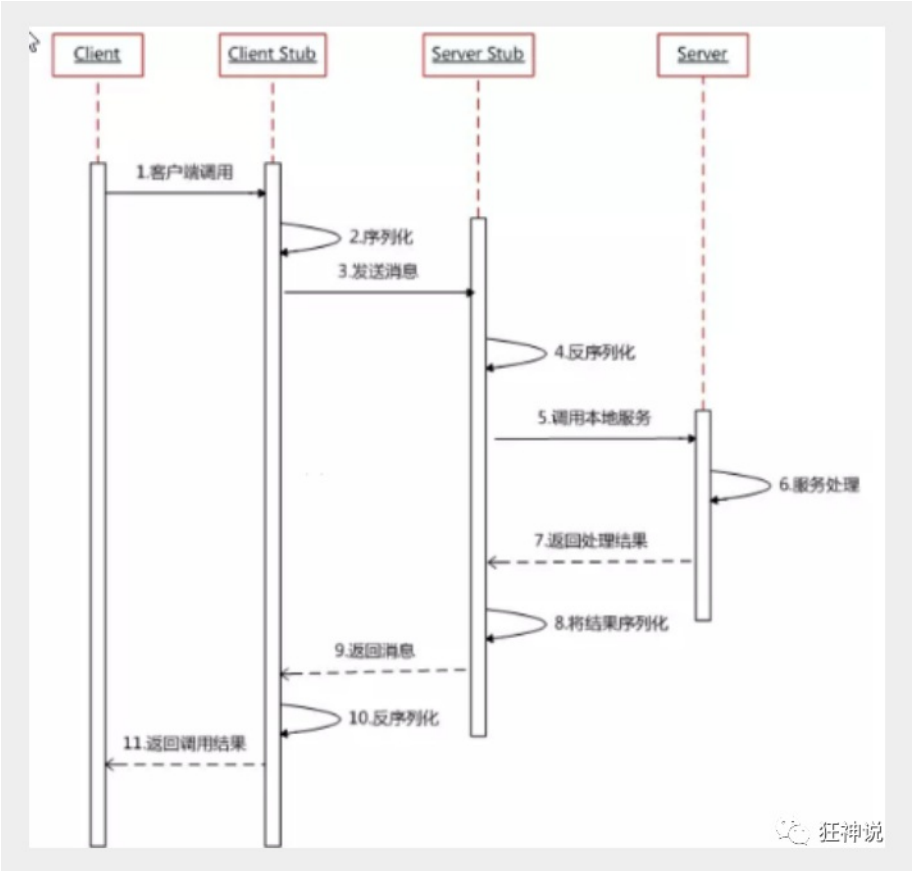
也就是说两台服务器A，B，一个应用部署在A服务器上，想要调用B服务器上应用提供的函数/方法，由于不在一个内存空间，不能直接调用，需要通过网络来表达调用的语义和传达调用的数据。为什么要用RPC呢？就是无法在一个进程内，甚至一个计算机内通过本地调用的方式完成的需求，比如不同的系统间的通讯，甚至不同的组织间的通讯，由于计算能力需要横向扩展，需要在多台机器组成的集群上部署应用。RPC就是要像调用本地的函数一样去调远程函数；

推荐阅读文章：https://www.jianshu.com/p/2accc2840a1b

**RPC基本原理**


client functions
server functions
1
10
5
6
client stub
server stub
(skeleton)
2
9
4
7
3
8
ketnel
sockets
sockets
ketnel
network
client
server
狂神说

**步骤解析：**




RPC两个核心模块：通讯，序列化。

## 测试环境搭建

### Dubbo

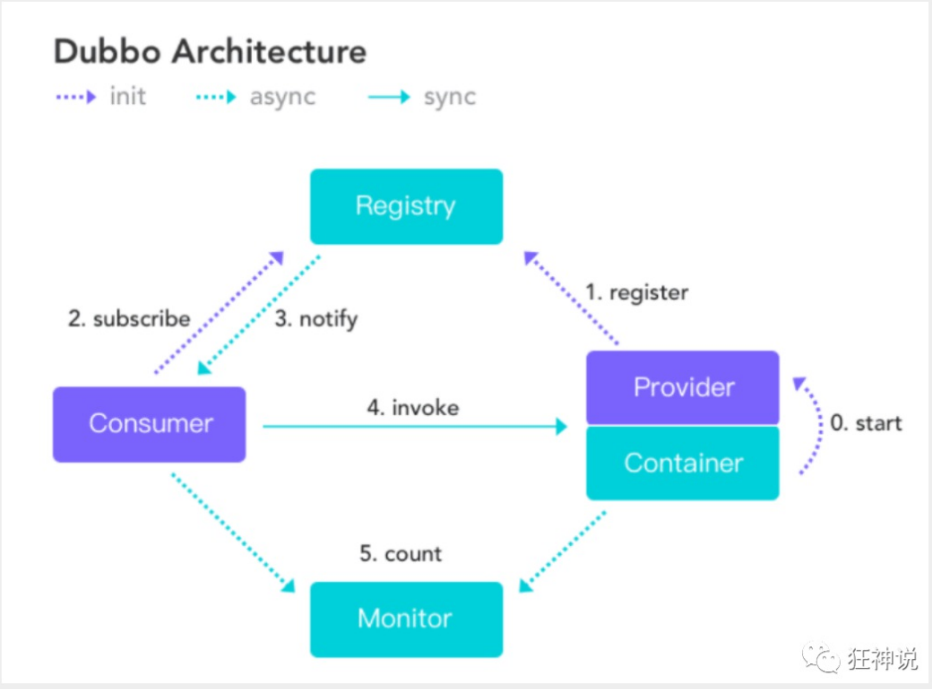
Apache Dubbo |ˈdʌbəʊ| 是一款高性能、轻量级的开源Java RPC框架，它提供了三大核心能力：面向接口的远程方法调用，智能容错和负载均衡，以及服务自动注册和发现。

dubbo官网 http://dubbo.apache.org/zh-cn/index.html

1.了解Dubbo的特性

2.查看官方文档

**dubbo基本概念**




**服务提供者**（Provider）：暴露服务的服务提供方，服务提供者在启动时，向注册中心注册自己提供的服务。

**服务消费者**（Consumer）：调用远程服务的服务消费方，服务消费者在启动时，向注册中心订阅自己所需的服务，服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。

**注册中心**（Registry）：注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者

**监控中心**（Monitor）：服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心

**调用关系说明**

l 服务容器负责启动，加载，运行服务提供者。

l 服务提供者在启动时，向注册中心注册自己提供的服务。

l 服务消费者在启动时，向注册中心订阅自己所需的服务。

l 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。

l 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。

l 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

## Dubbo环境搭建

点进dubbo官方文档，推荐我们使用Zookeeper 注册中心

什么是zookeeper呢？可以查看官方文档

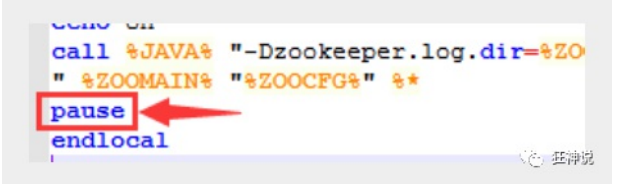
## Window下安装zookeeper

1、下载zookeeper：地址， 我们下载3.4.14， 最新版！解压zookeeper

2、运行/bin/zkServer.cmd，初次运行会报错，没有zoo.cfg配置文件；

可能遇到问题：闪退！

解决方案：编辑zkServer.cmd文件末尾添加pause 。这样运行出错就不会退出，会提示错误信息，方便找到原因。





3、修改zoo.cfg配置文件

将conf文件夹下面的zoo_sample.cfg复制一份改名为zoo.cfg即可。

注意几个重要位置：

dataDir=./ 临时数据存储的目录（可写相对路径）

clientPort=2181 zookeeper的端口号

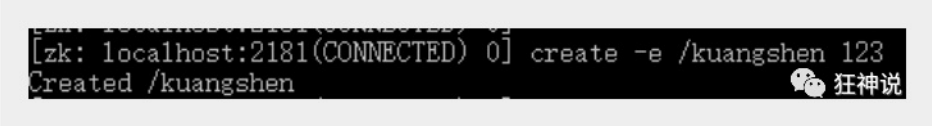
修改完成后再次启动zookeeper


4、使用zkCli.cmd测试

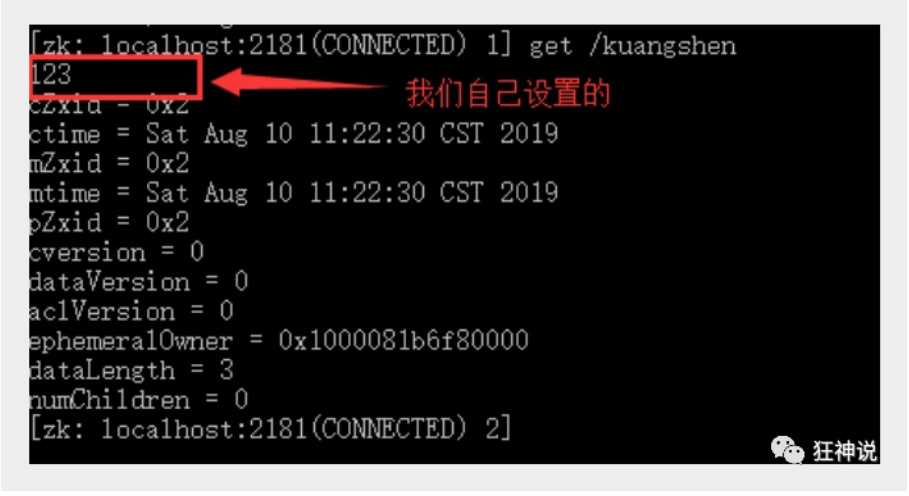
ls /：列出zookeeper根下保存的所有节点

```
[zk: 127.0.0.1:2181(CONNECTED) 4] ls /
[zookeeper]
```

create –e /kuangshen 123：创建一个kuangshen节点，值为123




get /kuangshen：获取/kuangshen节点的值




我们再来查看一下节点


## window下安装dubbo-admin

dubbo本身并不是一个服务软件。它其实就是一个jar包，能够帮你的java程序连接到zookeeper，并利用zookeeper消费、提供服务。

但是为了让用户更好的管理监控众多的dubbo服务，官方提供了一个可视化的监控程序dubbo-admin，不过这个监控即使不装也不影响使用。

我们这里来安装一下：

### 1、下载dubbo-admin

地址 ：https://github.com/apache/dubbo-admin/tree/master

**2、解压进入目录**

修改 dubbo-admin\src\main\resources \application.properties 指定zookeeper地址

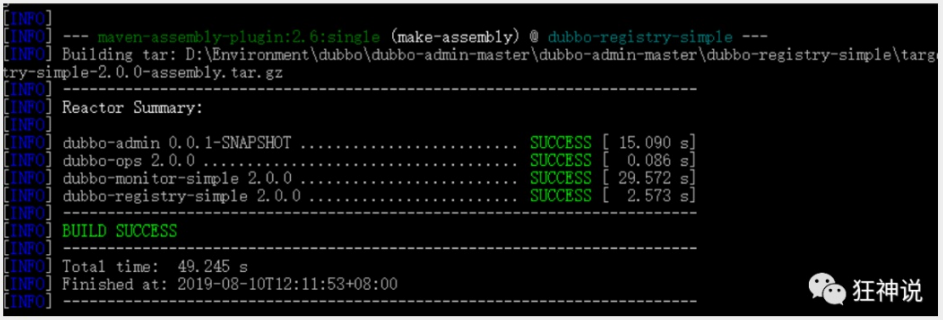
```
server.port=7001
spring.velocity.cache=false
spring.velocity.charset=UTF-8
spring.velocity.layout-url=/templates/default.vm
spring.messages.fallback-to-system-locale=false
spring.messages.basename=i18n/message
spring.root.password=root
spring.guest.password=guest

dubbo.registry.address=zookeeper://127.0.0.1:2181
```

**3、在项目目录下**打包dubbo-admin

```
mvn clean package -Dmaven.test.skip=true
```

**第一次打包的过程有点慢，需要耐心等待！直到成功！**



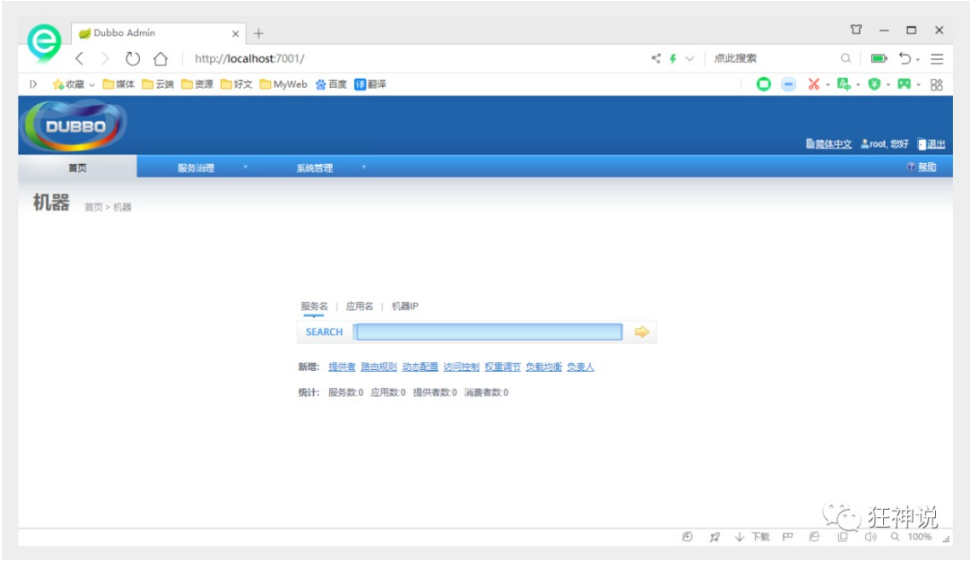

4、执行 dubbo-admin\target 下的dubbo-admin-0.0.1-SNAPSHOT.jar

```
java -jar dubbo-admin-0.0.1-SNAPSHOT.jar
```

【注意：zookeeper的服务一定要打开！】

执行完毕，我们去访问一下 http://localhost:7001/， 这时候我们需要输入登录账户和密码，我们都是默认的root-root；

登录成功后，查看界面




安装完成！

## SpringBoot + Dubbo + zookeeper

### 框架搭建

**1. 启动zookeeper ！**

**2. IDEA创建一个空项目；**

**3.创建一个模块，实现服务提供者：provider-server ， 选择web依赖即可**

**4.项目创建完毕，我们写一个服务，比如卖票的服务；**

编写接口

```
package com.kuang.provider.service;

public interface TicketService {
   public String getTicket();
}
```

编写实现类

```
package com.kuang.provider.service;

public class TicketServiceImpl implements TicketService {
   @Override
   public String getTicket() {
       return "《狂神说Java》";
  }
}
```

**5.创建一个模块，实现服务消费者：consumer-server ， 选择web依赖即可**

**6.项目创建完毕，我们写一个服务，比如用户的服务；**

编写service

```
package com.kuang.consumer.service;

public class UserService {
   //我们需要去拿去注册中心的服务
}
```

**需求：现在我们的用户想使用买票的服务，这要怎么弄呢 ？**

## 服务提供者

**1、将服务提供者注册到注册中心，我们需要整合Dubbo和zookeeper，所以需要导包**

**我们从dubbo官网进入github，看下方的帮助文档，找到dubbo-springboot，找到依赖包**

```
<!-- Dubbo Spring Boot Starter -->
<dependency>
   <groupId>org.apache.dubbo</groupId>
   <artifactId>dubbo-spring-boot-starter</artifactId>
   <version>2.7.3</version>
</dependency>
```

**zookeeper的包我们去maven仓库下载，zkclient；**

```
<!-- https://mvnrepository.com/artifact/com.github.sgroschupf/zkclient -->
<dependency>
   <groupId>com.github.sgroschupf</groupId>
   <artifactId>zkclient</artifactId>
   <version>0.1</version>
</dependency>
```

**【新版的坑】zookeeper及其依赖包，解决日志冲突，还需要剔除日志依赖；**

```
<!-- 引入zookeeper -->
<dependency>
   <groupId>org.apache.curator</groupId>
   <artifactId>curator-framework</artifactId>
   <version>2.12.0</version>
</dependency>
<dependency>
   <groupId>org.apache.curator</groupId>
   <artifactId>curator-recipes</artifactId>
   <version>2.12.0</version>
</dependency>
<dependency>
   <groupId>org.apache.zookeeper</groupId>
   <artifactId>zookeeper</artifactId>
```

```
    <version>3.4.14</version>
    <!--排除这个slf4j-log4j12-->
    <exclusions>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

**2、在springboot配置文件中配置dubbo相关属性！**

```
#当前应用名字
dubbo.application.name=provider-server
#注册中心地址
dubbo.registry.address=zookeeper://127.0.0.1:2181
#扫描指定包下服务
dubbo.scan.base-packages=com.kuang.provider.service
```

**3、在service的实现类中配置服务注解，发布服务！注意导包问题**

```
import org.apache.dubbo.config.annotation.Service;
import org.springframework.stereotype.Component;

@Service //将服务发布出去
@Component //放在容器中
public class TicketServiceImpl implements TicketService {
    @Override
    public String getTicket() {
        return "《狂神说Java》";
    }
}
```

**逻辑理解：应用启动起来，dubbo就会扫描指定的包下带有@component注解的服务，将它发布在指定的注册中心中！**

## 服务消费者

**1、导入依赖，和之前的依赖一样；**

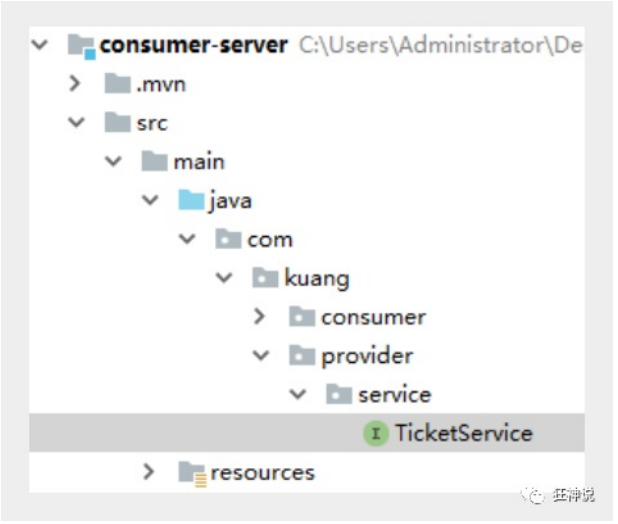
```
<!--dubbo-->
<!-- Dubbo Spring Boot Starter -->
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo-spring-boot-starter</artifactId>
    <version>2.7.3</version>
</dependency>
<!--zookeeper-->
<!-- https://mvnrepository.com/artifact/com.github.sgroschupf/zkclient -->
<dependency>
    <groupId>com.github.sgroschupf</groupId>
```

```
    <artifactId>zkclient</artifactId>
    <version>0.1</version>
</dependency>
<!-- 引入zookeeper -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
    <version>2.12.0</version>
</dependency>
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-recipes</artifactId>
    <version>2.12.0</version>
</dependency>
<dependency>
    <groupId>org.apache.zookeeper</groupId>
    <artifactId>zookeeper</artifactId>
    <version>3.4.14</version>
    <!--排除这个slf4j-log4j12-->
    <exclusions>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

2、**配置参数**

```
#当前应用名字
dubbo.application.name=consumer-server
#注册中心地址
dubbo.registry.address=zookeeper://127.0.0.1:2181
```

**3. 本来正常步骤是需要将服务提供者的接口打包，然后用pom文件导入，我们这里使用简单的方式，直接将服务的接口拿过来，路径必须保证正确，即和服务提供者相同；**




**4. 完善消费者的服务类**

```
package com.kuang.consumer.service;

import com.kuang.provider.service.TicketService;
import org.apache.dubbo.config.annotation.Reference;
import org.springframework.stereotype.Service;

@Service //注入到容器中
public class UserService {

   @Reference //远程引用指定的服务，他会按照全类名进行匹配，看谁给注册中心注册了这个全类名
   TicketService ticketService;

   public void bugTicket(){
       String ticket = ticketService.getTicket();
       System.out.println("在注册中心买到"+ticket);
   }

}
```

**5. 测试类编写；**

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class ConsumerServerApplicationTests {

   @Autowired
   UserService userService;

   @Test
   public void contextLoads() {
```

```
        userService.bugTicket();

    }

}
```

## 启动测试

1. 开启zookeeper

2. 打开dubbo-admin实现监控【可以不用做】

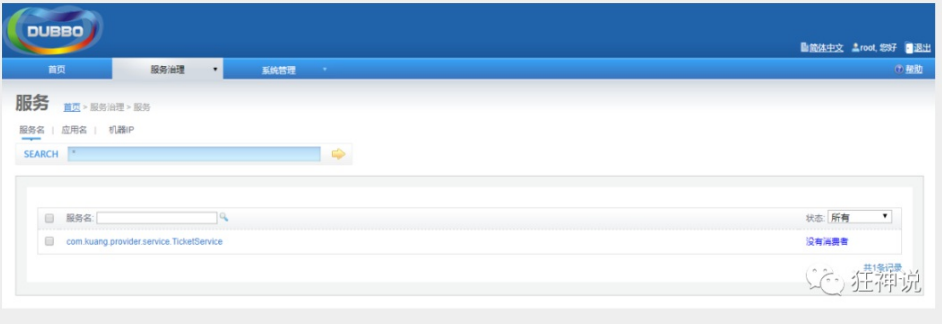
3. 开启服务者

4. 消费者消费测试，结果：

```
2019-08-11 12:45:47.951  INFO 1049
2019-08-11 12:45:48.639  INFO 1049
在注册中心买到《狂神说Java》
2019-08-11 12:45:49.186  INFO 1049
2019-08-11 12:45:49.200  INFO 1049
```



监控中心：




ok，这就是SpingBoot + dubbo + zookeeper实现分布式开发的应用，其实就是一个服务拆分的思想；

end

视频同步更新，这次一定！

“赠人玫瑰，手有余香”

狂神说 的赞赏码

狂神说