

# 1、基础知识

## 1.1、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**。

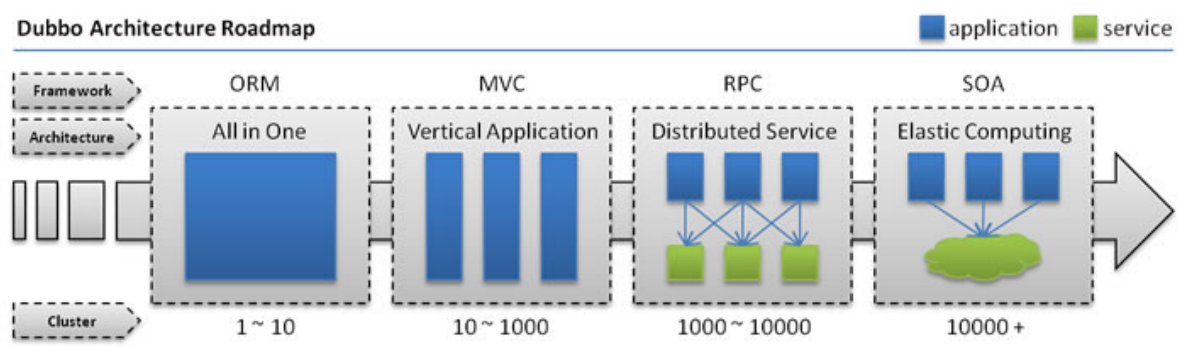
分布式系统 (distributed system) 是建立在网络之上的软件系统。

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。



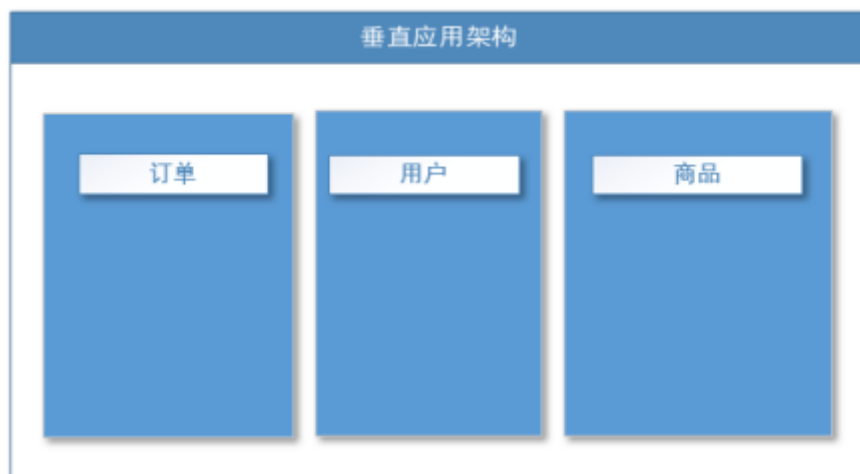
适用于小型网站，小型管理系统，将所有功能都部署到一个功能里，简单易用。

#### 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。

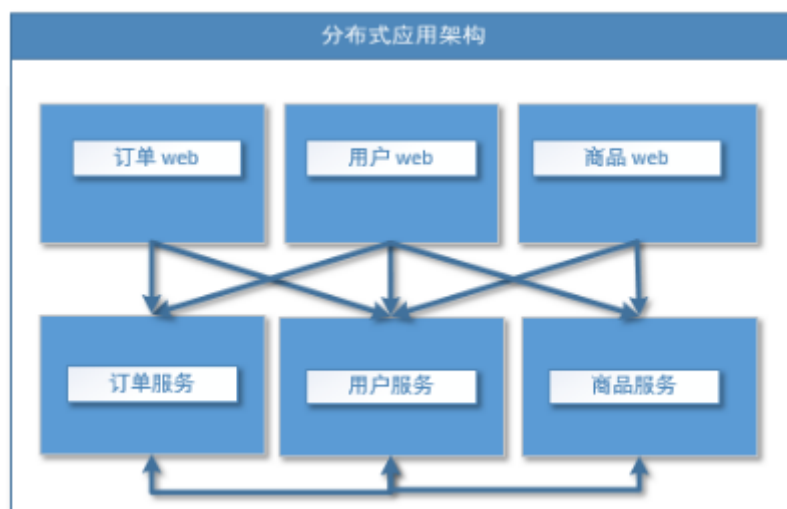


通过切分业务来实现各个模块独立部署，降低了维护和部署的难度，团队各司其职更易管理，性能扩展也更方便，更有针对性。

缺点： 公用模块无法重复利用，开发性的浪费

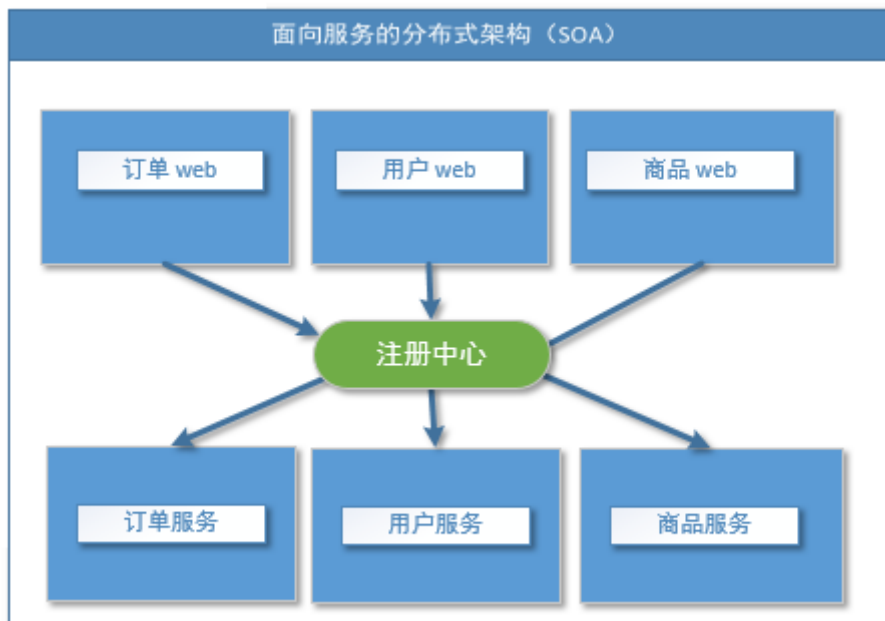
#### 分布式服务架构

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\*\*。



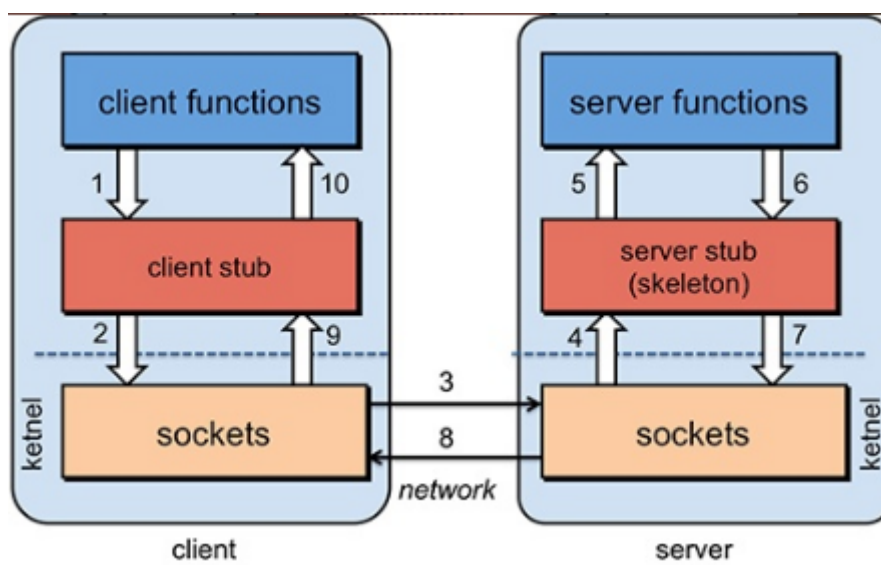
## 1.2、什么是RPC?

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。

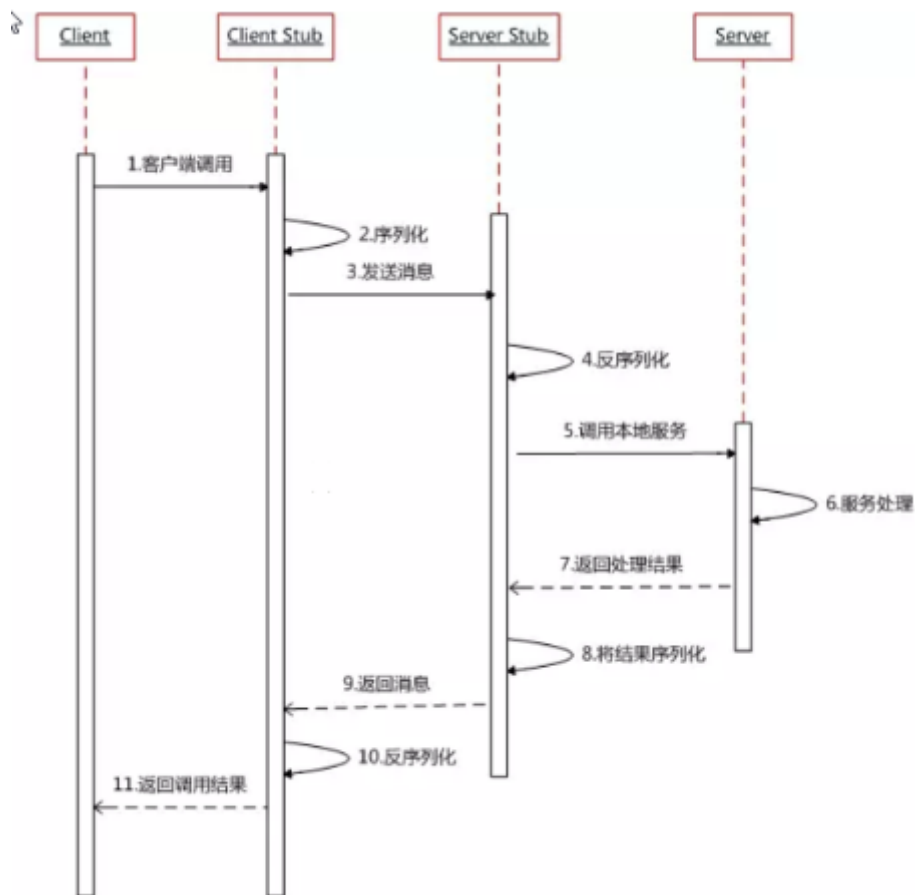
也就是说两台服务器A，B，一个应用部署在A服务器上，想要调用B服务器上应用提供的函数/方法，由于不在一个内存空间，不能直接调用，需要通过网络来表达调用的语义和传达调用的数据。为什么要用RPC呢？就是无法在一个进程内，甚至一个计算机内通过本地调用的方式完成的需求，比如不同的系统间的通讯，甚至不同的组织间的通讯，由于计算能力需要横向扩展，需要在多台机器组成的集群上部署应用。RPC就是要像调用本地的函数一样去调远程函数；

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

### RPC基本原理



步骤解析:



RPC两个核心模块：通讯，序列化。

## 2、Dubbo

### 2.1、什么是dubbo?

Apache Dubbo | 'dʌbəʊ| 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](#)

1.了解Dubbo的特性

2.查看官方文档

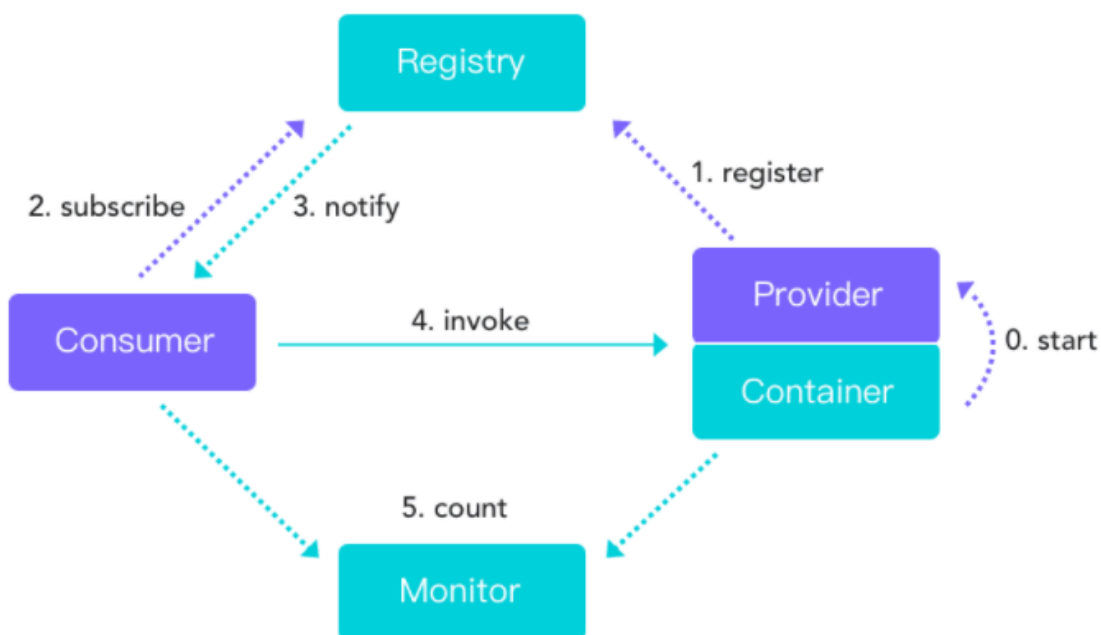
**dubbo基本概念**

# Dubbo Architecture

.....▶ init

.....▶ async

——▶ sync



**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

┆ 服务容器负责启动，加载，运行服务提供者。

┆ 服务提供者在启动时，向注册中心注册自己提供的服务。

┆ 服务消费者在启动时，向注册中心订阅自己所需的服务。

┆ 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。

┆ 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。

┆ 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。

## 2.2、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](#)

## 2.3、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。

```
call %JAVA% "-Dzookeeper.log.dir=%ZO  
"% %ZOOMAIN% "%ZOOCFG%" %*  
pause  
endlocal
```

```
D:\Environment\zookeeper-3.4.14\bin>call "D:\Environment\java\jdk1.8.0.201\bin\java" -Dzookeeper.log.dir=D:\Environment  
zookeeper-3.4.14\bin\.. "-Dzookeeper.root.logger=INFO,CONSOLE" -cp "D:\Environment\zookeeper-3.4.14\bin\..\build\class  
es;D:\Environment\zookeeper-3.4.14\bin\..\build\lib\*;D:\Environment\zookeeper-3.4.14\bin\..\*;D:\Environment\zookeeper-  
3.4.14\bin\..\lib\*;D:\Environment\zookeeper-3.4.14\bin\..\conf" org.apache.zookeeper.server.quorum.QuorumPeerMain "D:\E  
nvironment\zookeeper-3.4.14\bin\..\conf\zoo.cfg"  
2019-08-10 11:05:13,020 [myid:] - INFO [main:QuorumPeerConfig@136] - Reading configuration from: D:\Environment\zookeep  
er-3.4.14\bin\..\conf\zoo.cfg  
2019-08-10 11:05:13,023 [myid:] - ERROR [main:QuorumPeerMain@88] - Invalid config, exiting abnormally  
org.apache.zookeeper.server.quorum.QuorumPeerConfig$ConfigException: Error processing D:\Environment\zookeeper-3.4.14\bi  
n\..\conf\zoo.cfg  
    at org.apache.zookeeper.server.quorum.QuorumPeerConfig.parse(QuorumPeerConfig.java:156)  
    at org.apache.zookeeper.server.quorum.QuorumPeerMain.initializeAndRun(QuorumPeerMain.java:104)  
    at org.apache.zookeeper.server.quorum.QuorumPeerMain.main(QuorumPeerMain.java:81)  
Caused by: java.lang.IllegalArgumentException: D:\Environment\zookeeper-3.4.14\bin\..\conf\zoo.cfg file is missing  
    at org.apache.zookeeper.server.quorum.QuorumPeerConfig.parse(QuorumPeerConfig.java:140)  
    ... 2 more  
Invalid config, exiting abnormally  
D:\Environment\zookeeper-3.4.14\bin>pause  
请按任意键继续. . .
```

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

```
2019-08-10 11:08:27,535 [myid:] - INFO [main:NIOServerCnxnFactory@89] - binding to port 0.0.0.0/0.0.0.0:2181
```

4、使用zkCli.cmd测试

ls /: 列出zookeeper根下保存的所有节点

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

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

```
[zk: localhost:2181(CONNECTED) 0] create -e /kuangshen 123  
Created /kuangshen
```

get /kuangshen: 获取/kuangshen节点的值

```
[zk: localhost:2181(CONNECTED) 1] get /kuangshen
123
cZxid = 0x2
ctime = Sat Aug 10 11:22:30 CST 2019
mZxid = 0x2
mtime = Sat Aug 10 11:22:30 CST 2019
pZxid = 0x2
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x1000081b6f80000
dataLength = 3
numChildren = 0
[zk: localhost:2181(CONNECTED) 2]
```

我们自己设置的

我们再来看一下节点

```
[zk: localhost:2181(CONNECTED) 3] ls /
[kuangshen, zookeeper]
```

## 2.4、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>  
[\]\(https://github.com/apache/incubator-dubbo-ops\)](https://github.com/apache/incubator-dubbo-ops)

### 2、解压进入目录

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

```
1 server.port=7001
2 spring.velocity.cache=false
3 spring.velocity.charset=UTF-8
4 spring.velocity.layout-url=/templates/default.vm
5 spring.messages.fallback-to-system-locale=false
6 spring.messages.basename=i18n/message
7 spring.root.password=root
8 spring.guest.password=guest
9
10 dubbo.registry.address=zookeeper://127.0.0.1:2181
```

### 3、在项目目录下打包dubbo-admin

```
1 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！

```

[INFO] --- maven-assembly-plugin:2.6:single (make-assembly) @ dubbo-registry-simple ---
[INFO] Building tar: D:\Environment\dubbo\dubbo-admin-master\dubbo-admin-master\dubbo-registry-simple\targ
try-simple-2.0.0-assembly.tar.gz
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] dubbo-admin 0.0.1-SNAPSHOT ..... SUCCESS [ 15.090 s]
[INFO] dubbo-ops 2.0.0 ..... SUCCESS [ 0.086 s]
[INFO] dubbo-monitor-simple 2.0.0 ..... SUCCESS [ 29.572 s]
[INFO] dubbo-registry-simple 2.0.0 ..... SUCCESS [ 2.573 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 49.245 s
[INFO] Finished at: 2019-08-10T12:11:53+08:00
[INFO] -----

```

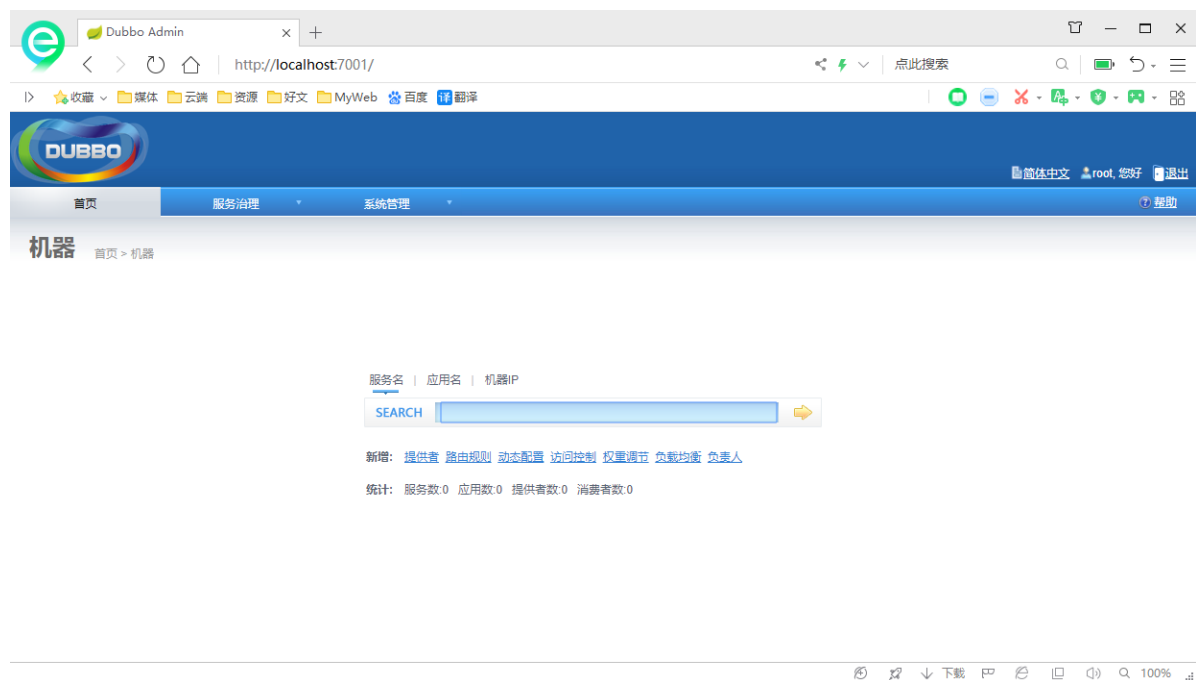
4、执行 dubbo-admin\target 下的dubbo-admin-0.0.1-SNAPSHOT.jar

```
1 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！

## 3、SpringBoot + Dubbo + zookeeper

### 3.1、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



```

1 package com.kuang.provider.service;
2
3 public interface TicketService {
4     public String getTicket();
5 }

```

编写实现类

```

1 package com.kuang.provider.service;
2
3 public class TicketServiceImpl implements TicketService {
4     @Override
5     public String getTicket() {
6         return "《狂神说Java》";
7     }
8 }

```

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

```

1 package com.kuang.consumer.service;
2
3 public class UserService {
4     //我们需要去拿去注册中心的服务
5 }

```

需求：现在的用户想使用买票的服务，这要怎么弄呢？

## 3.2、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

```

1 <!-- Dubbo Spring Boot Starter -->
2 <dependency>
3     <groupId>org.apache.dubbo</groupId>
4     <artifactId>dubbo-spring-boot-starter</artifactId>
5     <version>2.7.3</version>
6 </dependency>

```

zookeeper的包我们去maven仓库下载，zkclient;

```

1 <!-- https://mvnrepository.com/artifact/com.github.sgroschupf/zkclient -->
2 <dependency>
3     <groupId>com.github.sgroschupf</groupId>
4     <artifactId>zkclient</artifactId>
5     <version>0.1</version>
6 </dependency>
7

```

【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；

```

1 <!-- 引入zookeeper -->
2 <dependency>

```

```

3     <groupId>org.apache.curator</groupId>
4     <artifactId>curator-framework</artifactId>
5     <version>2.12.0</version>
6 </dependency>
7 <dependency>
8     <groupId>org.apache.curator</groupId>
9     <artifactId>curator-recipes</artifactId>
10    <version>2.12.0</version>
11 </dependency>
12 <dependency>
13     <groupId>org.apache.zookeeper</groupId>
14     <artifactId>zookeeper</artifactId>
15     <version>3.4.14</version>
16     <!--排除这个slf4j-log4j12-->
17     <exclusions>
18         <exclusion>
19             <groupId>org.slf4j</groupId>
20             <artifactId>slf4j-log4j12</artifactId>
21         </exclusion>
22     </exclusions>
23 </dependency>
24

```

## 2、在springboot配置文件中配置dubbo相关属性!

```

1  #当前应用名字
2  dubbo.application.name=provider-server
3  #注册中心地址
4  dubbo.registry.address=zookeeper://127.0.0.1:2181
5  #扫描指定包下服务
6  dubbo.scan.base-packages=com.kuang.provider.service
7

```

## 3、在service的实现类中配置服务注解，发布服务！注意导包问题

```

1  import org.apache.dubbo.config.annotation.Service;
2  import org.springframework.stereotype.Component;
3
4  @Service //将服务发布出去
5  @Component //放在容器中
6  public class TicketServiceImpl implements TicketService {
7      @Override
8      public String getTicket() {
9          return "《狂神说Java》";
10     }
11 }
12

```

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！

## 3.3、消费者

### 1、导入依赖，和之前的依赖一样；

```

1  <!--dubbo-->
2  <!-- Dubbo Spring Boot Starter -->

```

```

3 <dependency>
4   <groupId>org.apache.dubbo</groupId>
5   <artifactId>dubbo-spring-boot-starter</artifactId>
6   <version>2.7.3</version>
7 </dependency>
8 <!--zookeeper-->
9 <!-- https://mvnrepository.com/artifact/com.github.sgroschupf/zkclient -->
10 <dependency>
11   <groupId>com.github.sgroschupf</groupId>
12   <artifactId>zkclient</artifactId>
13   <version>0.1</version>
14 </dependency>
15 <!-- 引入zookeeper -->
16 <dependency>
17   <groupId>org.apache.curator</groupId>
18   <artifactId>curator-framework</artifactId>
19   <version>2.12.0</version>
20 </dependency>
21 <dependency>
22   <groupId>org.apache.curator</groupId>
23   <artifactId>curator-recipes</artifactId>
24   <version>2.12.0</version>
25 </dependency>
26 <dependency>
27   <groupId>org.apache.zookeeper</groupId>
28   <artifactId>zookeeper</artifactId>
29   <version>3.4.14</version>
30 <!--排除这个slf4j-log4j12-->
31 <exclusions>
32   <exclusion>
33     <groupId>org.slf4j</groupId>
34     <artifactId>slf4j-log4j12</artifactId>
35   </exclusion>
36 </exclusions>
37 </dependency>
38

```

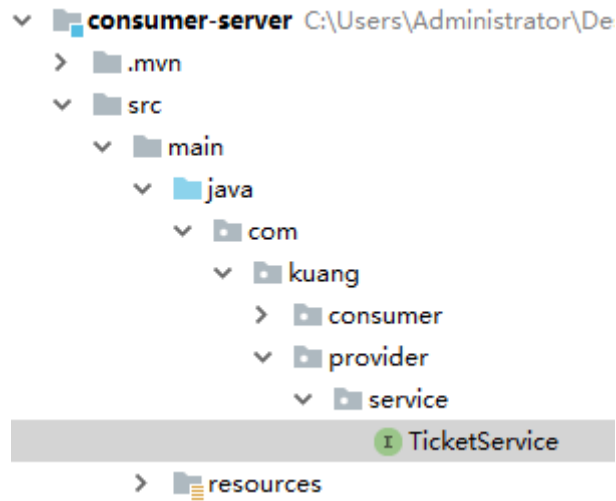
## 2、配置参数

```

1 #当前应用名字
2 dubbo.application.name=consumer-server
3 #注册中心地址
4 dubbo.registry.address=zookeeper://127.0.0.1:2181
5

```

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

```
1 package com.kuang.consumer.service;
2
3 import com.kuang.provider.service.TicketService;
4 import org.apache.dubbo.config.annotation.Reference;
5 import org.springframework.stereotype.Service;
6
7 @Service //注入到容器中
8 public class UserService {
9
10     @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
11     TicketService ticketService;
12
13     public void bugTicket(){
14         String ticket = ticketService.getTicket();
15         System.out.println("在注册中心买到"+ticket);
16     }
17
18 }
19
```

#### 5. 测试类编写;

```
1 @RunWith(SpringRunner.class)
2 @SpringBootTest
3 public class ConsumerServerApplicationTests {
4
5     @Autowired
6     UserService userService;
7
8     @Test
9     public void contextLoads() {
10
11         userService.bugTicket();
12
13     }
14
15 }
16
```

## 3.4、启动测试

1. 开启zookeeper

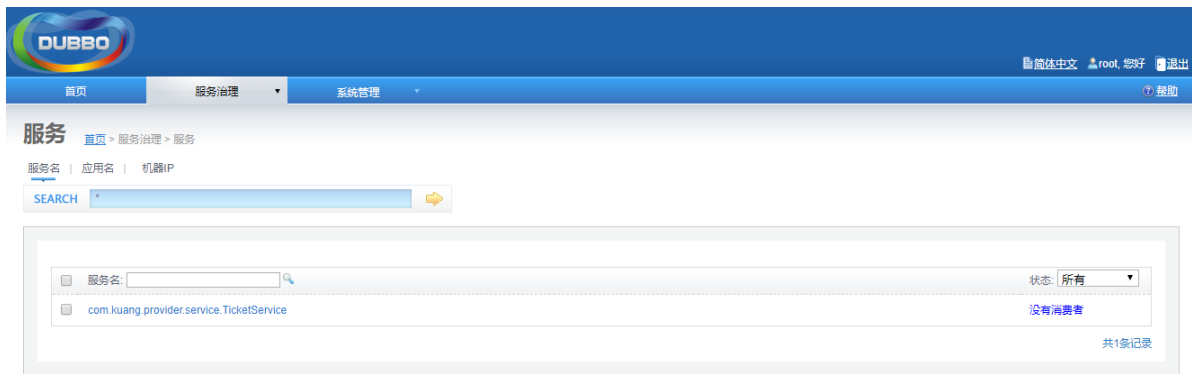
2. 打开dubbo-admin实现监控【可以不用做】

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SpringBoot + 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；