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

**发展演变**

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。**

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。

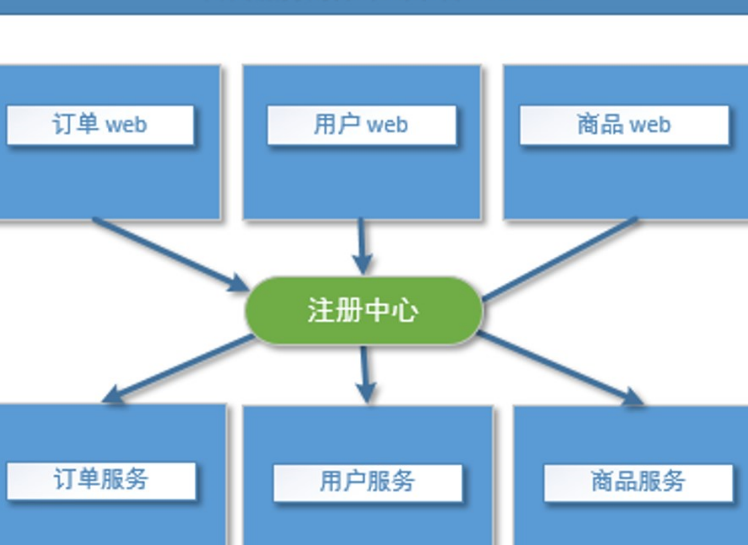
缺点： 公用模块无法重复利用，开发性的浪费

**2.3，分布式服务架构**

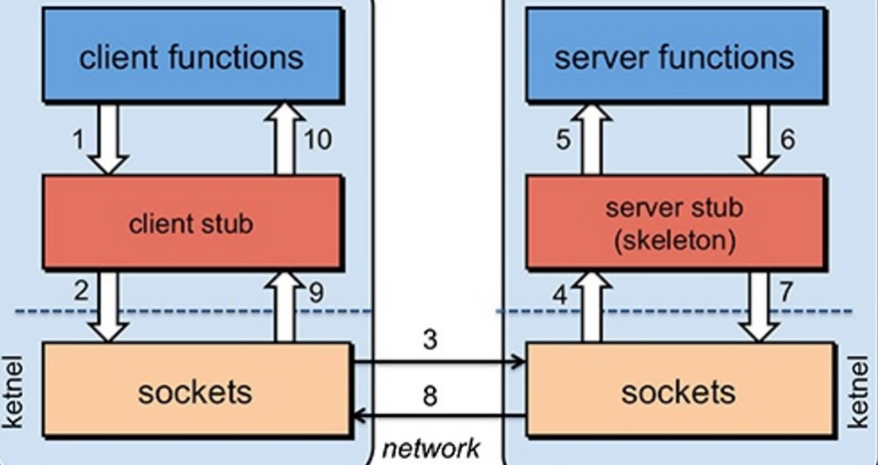
当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，使前端应用能更快速的响应多变的市场需求。此时，用于提高业务复用及整合的分布式服务框架(**RPC)是关键。**

**2.4，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。**

**RPC Remote Procedure Call**

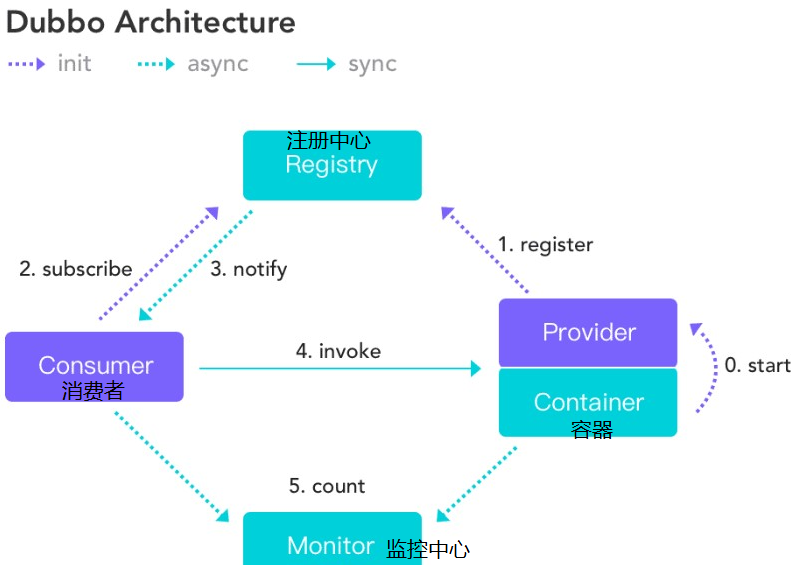
RPC是指远程调用或进程间通信的方式，他是一种技术的思想而不是规范。它允许程序调用另一个地址空间（通常是共享网络的另一台机器上）的过程或函数，而不用程序员显式编码这个远程调用的细节(之前要调用另一个进程需要手写套接字)。即程序员无论是调用本地的还是远程的函数，本质上编写的调用代码基本相同。

RPC基本原理:进程之间通过socket套接字实现通信

RPC两个核心模块：通讯，序列化。

# 简介

Apache Dubbo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。



**服务提供者（Provider）：**暴露服务的服务提供方，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，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，定时每分钟发送统计数据到监控中心,记录在内存中累计调用次数和调用时间，

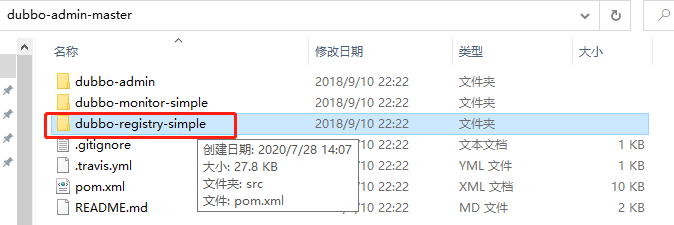
# 安装(windows)

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

dubbo并不是一个服务软件,而是一个jar包,能够帮java程序连接zookeeper，并利用zookeeper消费、提供服务。所以你不用在Linux上启动dubbo服务

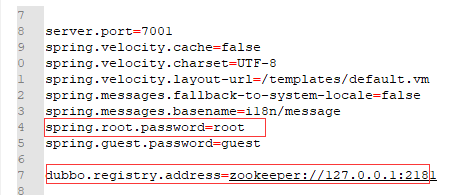
但是为了让用户更好的管理监控众多的dubbo服务，dubbo提供了一个可视化的监控程序，不过这个监控即使不装也不影响使用。

dubbo自带一个注册器,被zookeeper取代了注册的功能



进入dubbo-admin-master

修改 src\main\resources\application.properties 指定zookeeper地址



打包mvn package

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

# 整合springboot

提供者是一个java项目,消费者是web项目

dubbo整合springboot和三种方式

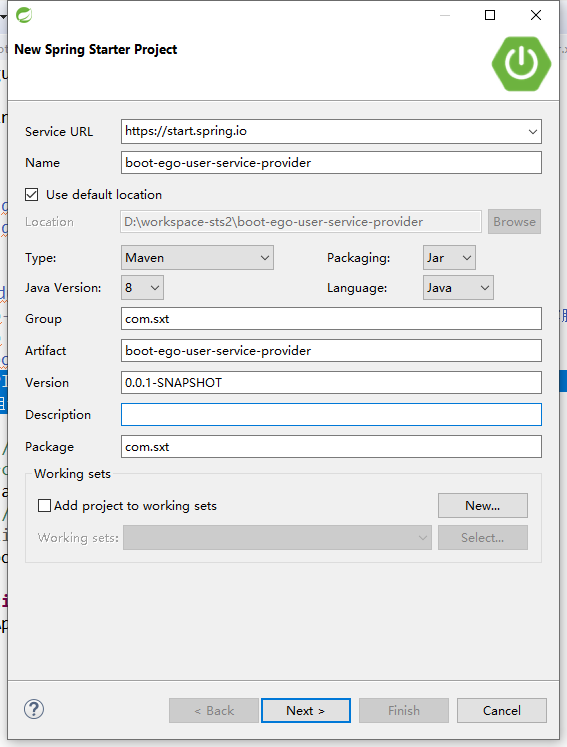
方式1：引入dubbo-starter，在application.properties配置属性，使用@Service【暴露服务】使用@Reference【引用服务】

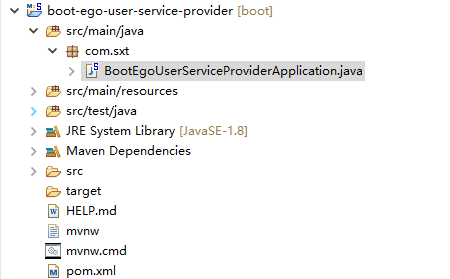
方式2：保留xml配置文件; 导入dubbo-starter，使用@ImportResource导入dubbo的配置文件即可

方式3：使用注解API的方式， 将每一个组件手动创建到容器中,让dubbo来扫描组件

## **2，服务提供者**

### **2.1，创建boot-ego-user-service-provider**





### **2.2，加入依赖**

1

<?xml version="1.0" encoding="UTF-8"?>

2

<project xmlns="http://maven.apache.org/POM/4.0.0"

3

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

4

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

5

<modelVersion>4.0.0</modelVersion>

6

<parent>

7

<groupId>org.springframework.boot</groupId>

8

<artifactId>spring-boot-starter-parent</artifactId>

9

<version>2.0.4.RELEASE</version>

10

<relativePath /> <!-- lookup parent from repository -->

11

</parent>

12

<groupId>com.sxt</groupId>

13

<artifactId>boot-ego-user-service-provider</artifactId>

14

<version>0.0.1-SNAPSHOT</version>

15

<name>boot-ego-user-service-provider</name>

16

<description>Demo project for Spring Boot</description>

17

18

<properties>

19

<java.version>1.8</java.version>

20

<maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>

21

<dubbo.version>2.6.5</dubbo.version>

22

23

</properties>

24

25

<dependencies>

26

27

<dependency>

28

<groupId>com.sxt</groupId>

29

<artifactId>ego-interface</artifactId>

30

<version>0.0.1-SNAPSHOT</version>

31

</dependency>

32

<!-- Dubbo Spring Boot Starter -->

33

<dependency>

34

<groupId>com.alibaba.boot</groupId>

35

<artifactId>dubbo-spring-boot-starter</artifactId>

36

<version>0.2.1.RELEASE</version>

37

</dependency>

38

<dependency>

39

<groupId>com.alibaba</groupId>

40

<artifactId>dubbo</artifactId>

41

<version>${dubbo.version}</version>

42

</dependency>

43

<dependency>

44

<groupId>io.netty</groupId>

45

<artifactId>netty-all</artifactId>

46

</dependency>

47

<!-- curator-framework -->

48

<dependency>

49

<groupId>org.apache.curator</groupId>

50

<artifactId>curator-framework</artifactId>

51

<version>2.12.0</version>

52

</dependency>

53

<dependency>

54

<groupId>org.springframework.boot</groupId>

55

<artifactId>spring-boot-starter</artifactId>

56

</dependency>

57

58

<dependency>

59

<groupId>org.springframework.boot</groupId>

60

<artifactId>spring-boot-starter-test</artifactId>

61

<scope>test</scope>

62

</dependency>

63

</dependencies>

64

65

<build>

66

<plugins>

67

<plugin>

68

<groupId>org.springframework.boot</groupId>

69

<artifactId>spring-boot-maven-plugin</artifactId>

70

</plugin>

71

</plugins>

72

</build>

73

74

</project>

75

### **2.3，创建UserServiceImpl**

1

import com.alibaba.dubbo.config.annotation.Service;

2

import com.sxt.domain.UserAddress;

3

import com.sxt.service.UserService;

4

5

@Service//暴露服务

6

public class UserServiceImpl implements UserService{

7

public static List<UserAddress> address=new ArrayList<>();

8

static {

9

address.add(new UserAddress(1, "湖北省武汉市东湖高新区金融港B22栋11楼", "whsxt"));

10

address.add(new UserAddress(2, "北京市海淀区西三旗街道建材城西路中腾建华商务大厦东侧二层尚学堂", "bjsxt"));

11

}

12

@Override

13

public List<UserAddress> getUserAddressList(String userId) {

14

//讲道理要是去数据库里面去查询的

15

return address;

16

}

17

}

18

### **2.4，配置properties文件**

1

dubbo.application.name=boot-ego-user-service-provider

2

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

3

4

dubbo.protocol.name=dubbo

5

dubbo.protocol.port=20880

6

7

#dubbo.monitor.protocol=registry

### **2.5，修改启动类**

10

9

1

@EnableDubbo//开户注解的dubbo功能

2

@SpringBootApplication

3

public class BootEgoUserServiceProviderApplication {

4

5

public static void main(String[] args) {

6

SpringApplication.run(BootEgoUserServiceProviderApplication.class, args);

7

}

8

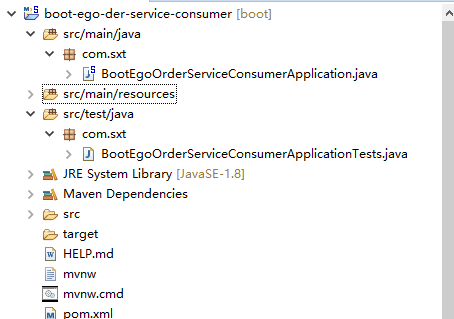
}

### **2.6，启动测试**



**3，服务消费者**

### **3.1，创建boot-ego-order-service-consumer**



### **3.2，加入依赖**

1

<?xml version="1.0" encoding="UTF-8"?>

2

<project xmlns="http://maven.apache.org/POM/4.0.0"

3

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

4

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

5

<modelVersion>4.0.0</modelVersion>

6

<parent>

7

<groupId>org.springframework.boot</groupId>

8

<artifactId>spring-boot-starter-parent</artifactId>

9

<version>2.0.4.RELEASE</version>

10

<relativePath /> <!-- lookup parent from repository -->

11

</parent>

12

<groupId>com.sxt</groupId>

13

<artifactId>boot-ego-user-service-provider</artifactId>

14

<version>0.0.1-SNAPSHOT</version>

15

<name>boot-ego-user-service-provider</name>

16

<description>Demo project for Spring Boot</description>

17

18

<properties>

19

<java.version>1.8</java.version>

20

<dubbo.version>2.6.5</dubbo.version>

21

</properties>

22

23

<dependencies>

24

25

<dependency>

26

<groupId>com.sxt</groupId>

27

<artifactId>ego-interface</artifactId>

28

<version>0.0.1-SNAPSHOT</version>

29

</dependency>

30

<dependency>

31

<groupId>org.springframework.boot</groupId>

32

<artifactId>spring-boot-starter</artifactId>

33

</dependency>

34

<dependency>

35

<groupId>org.springframework.boot</groupId>

36

<artifactId>spring-boot-starter-web</artifactId>

37

</dependency>

38

<dependency>

39

<groupId>org.springframework.boot</groupId>

40

<artifactId>spring-boot-starter-test</artifactId>

41

<scope>test</scope>

42

</dependency>

43

<!-- Dubbo Spring Boot Starter -->

44

<dependency>

45

<groupId>com.alibaba.boot</groupId>

46

<artifactId>dubbo-spring-boot-starter</artifactId>

47

<version>0.2.1.RELEASE</version>

48

</dependency>

49

<dependency>

50

<groupId>com.alibaba</groupId>

51

<artifactId>dubbo</artifactId>

52

<version>${dubbo.version}</version>

53

</dependency>

54

<dependency>

55

<groupId>io.netty</groupId>

56

<artifactId>netty-all</artifactId>

57

</dependency>

58

<!-- curator-framework -->

59

<dependency>

60

<groupId>org.apache.curator</groupId>

61

<artifactId>curator-framework</artifactId>

62

<version>2.12.0</version>

63

</dependency>

64

</dependencies>

65

66

<build>

67

<plugins>

68

<plugin>

69

<groupId>org.springframework.boot</groupId>

70

<artifactId>spring-boot-maven-plugin</artifactId>

71

</plugin>

72

</plugins>

73

</build>

74

75

</project>

76

### **3.3，创建OrderServiceImpl**

1

@Service

2

public class OrderServiceImpl implements OrderService {

3

4

@Reference

5

UserService userService;

6

7

@Override

8

public List<UserAddress> initOrder(String userId) {

9

System.out.println("用户id："+userId);

10

//1、查询用户的收货地址

11

List<UserAddress> addressList = userService.getUserAddressList(userId);

12

return addressList;

13

}

14

}

### **3.4，创建OrderController**

1

@Controller

2

public class OrderController {

3

4

@Autowired

5

OrderService orderService;

6

7

@ResponseBody

8

@RequestMapping("/initOrder")

9

public List<UserAddress> initOrder(@RequestParam("uid")String userId) {

10

return orderService.initOrder(userId);

11

}

12

13

}

14

### **3.5，修改properties**

1

dubbo.application.name=boot-ego-order-service-consumer

2

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

3

4

dubbo.monitor.protocol=registry

5

6

server.port=8888

### **3.5，启动类配置**

10

9

1

@SpringBootApplication

2

@EnableDubbo //启动dubbo

3

public class BootEgoOrderServiceConsumerApplication {

4

public static void main(String[] args) {

5

SpringApplication.run(BootEgoOrderServiceConsumerApplication.class, args);

6

}

7

}

**3.6，启动测试**

