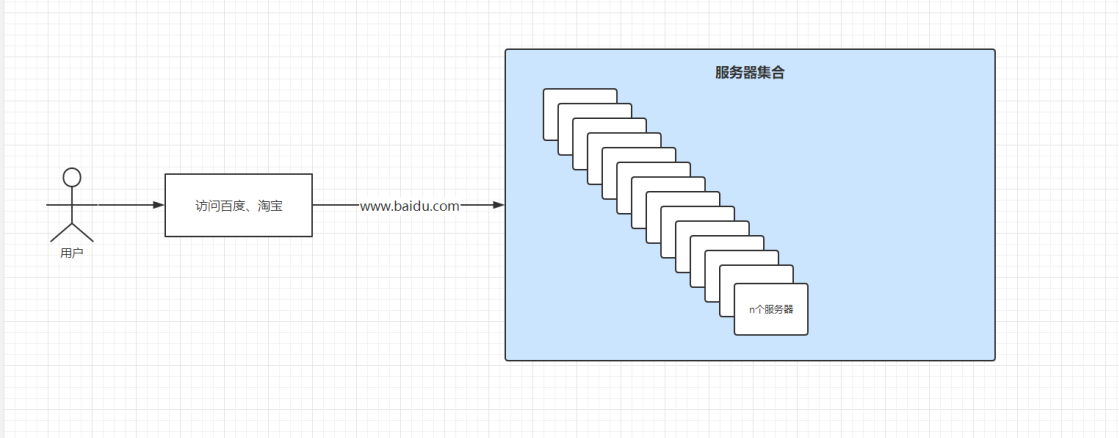
分布式 Dubbo+ZooKeeper

1 分布式理论

1.1 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”；

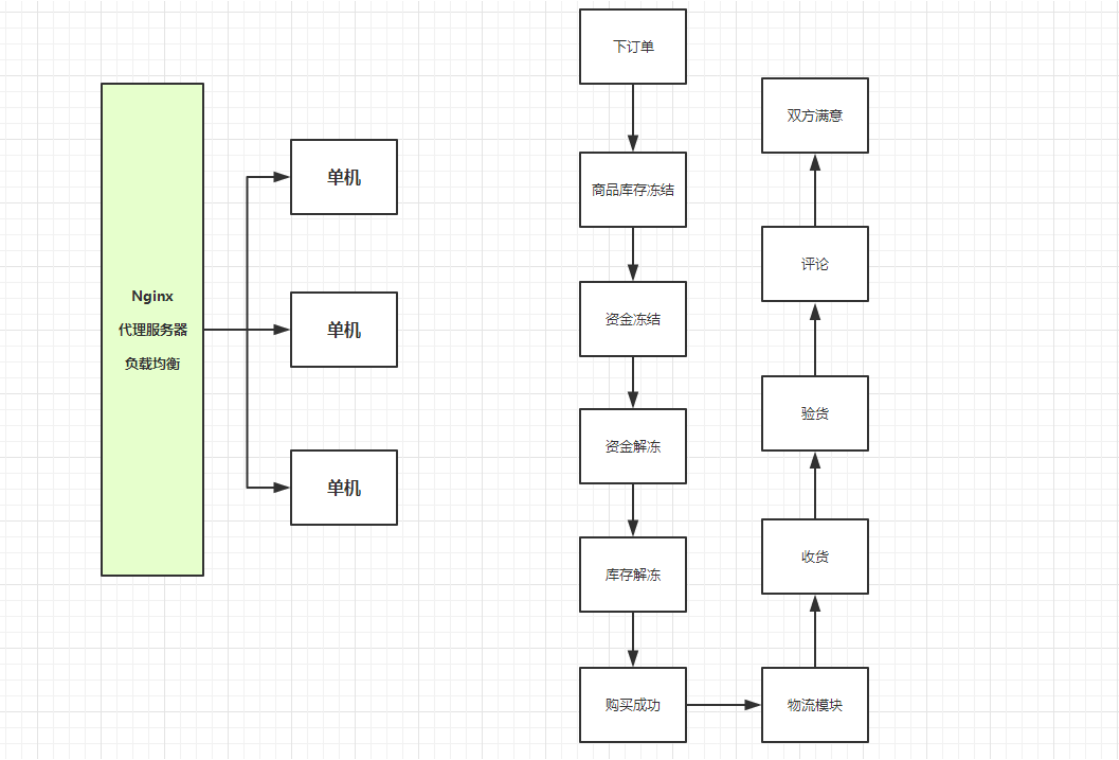
  画一张图来促进理解：



  - 对于用户来说就像单个系统：用户只访问www.baidu.com这个大主机
  - 若干独立计算机的集合：内部有多个服务器
- 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。

  内部的计算机之间肯定有联系，这里有两种联系的方法：

  - Http协议
  - RPC
- 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。**【最早是Google提出来的】
- 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。



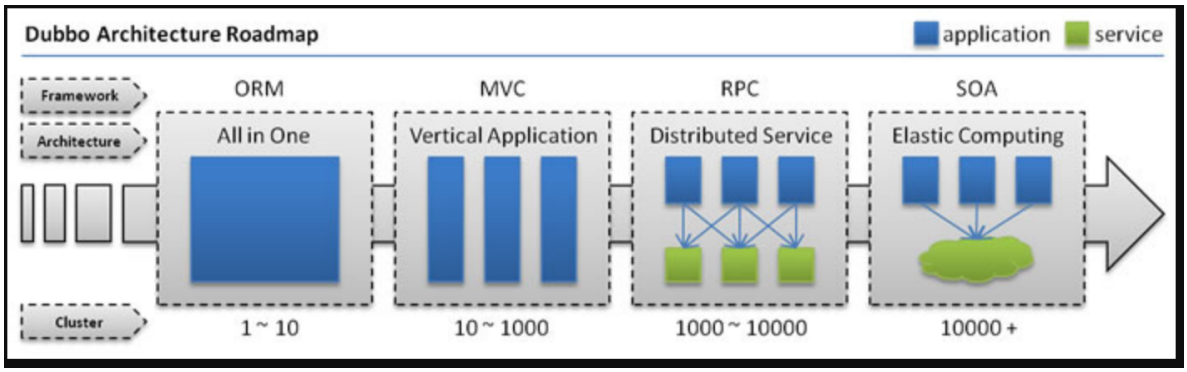
上面的流程中可以分为多个模块，每个模块都放在不同的服务器上，又可以根据模块的复杂度安排服务器（比如订单系统简单，那么可以把所有订单放在一台服务器上；物流和支付系统比较慢，就放在多台服务器上），这就将原来的单机系统分为多个服务器。

但是，我们这时候就得想办法让多个服务器之间进行通信，这就会变得十分复杂了。所以使用分布式系统，我们必须考虑到用户量或者一些业务量是否达到使用分布式的标准。

- 简而言之，分布式系统就是将一个整体拆分成多个部分，然后这些部分根据实际复杂程度复制成不同的份数，每部分之间进行通信。

1.2 Dubbo官网文档

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。



了解一下各种架构模式：

- 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：

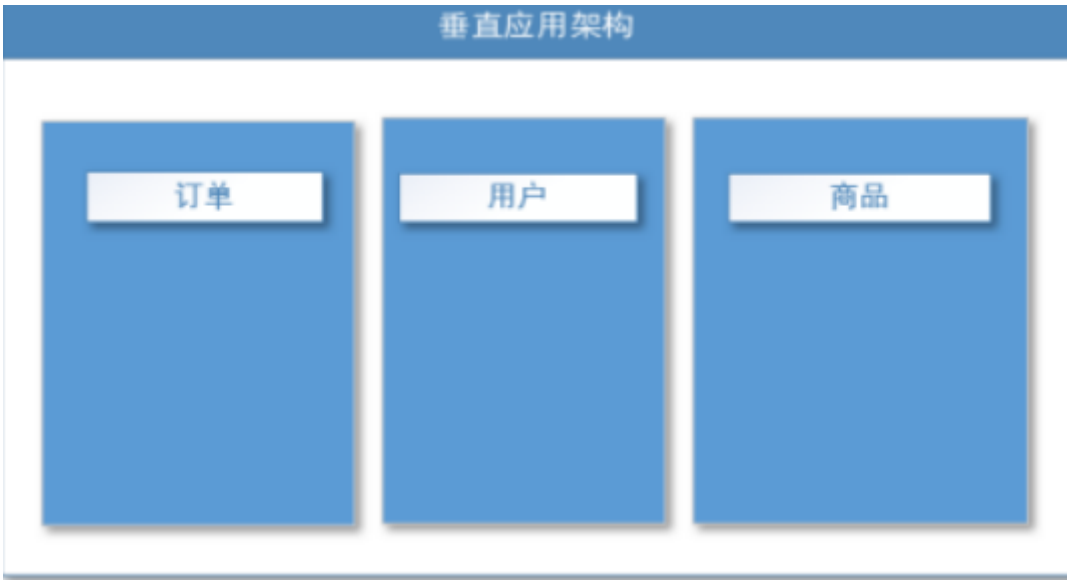
1、性能扩展比较难

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，提升效率的方法之一是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。

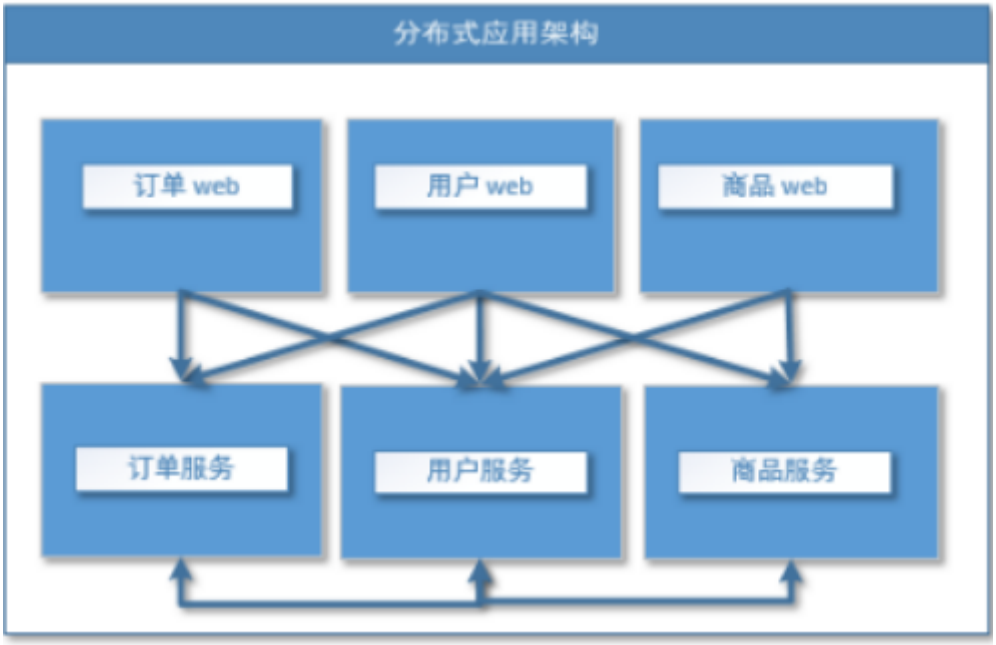


通过切分业务来实现各个模块独立部署，降低了维护和部署的难度，团队各司其职更易管理，性能扩展也更方便，更有针对性。

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)**是关键。

1.3 RPC

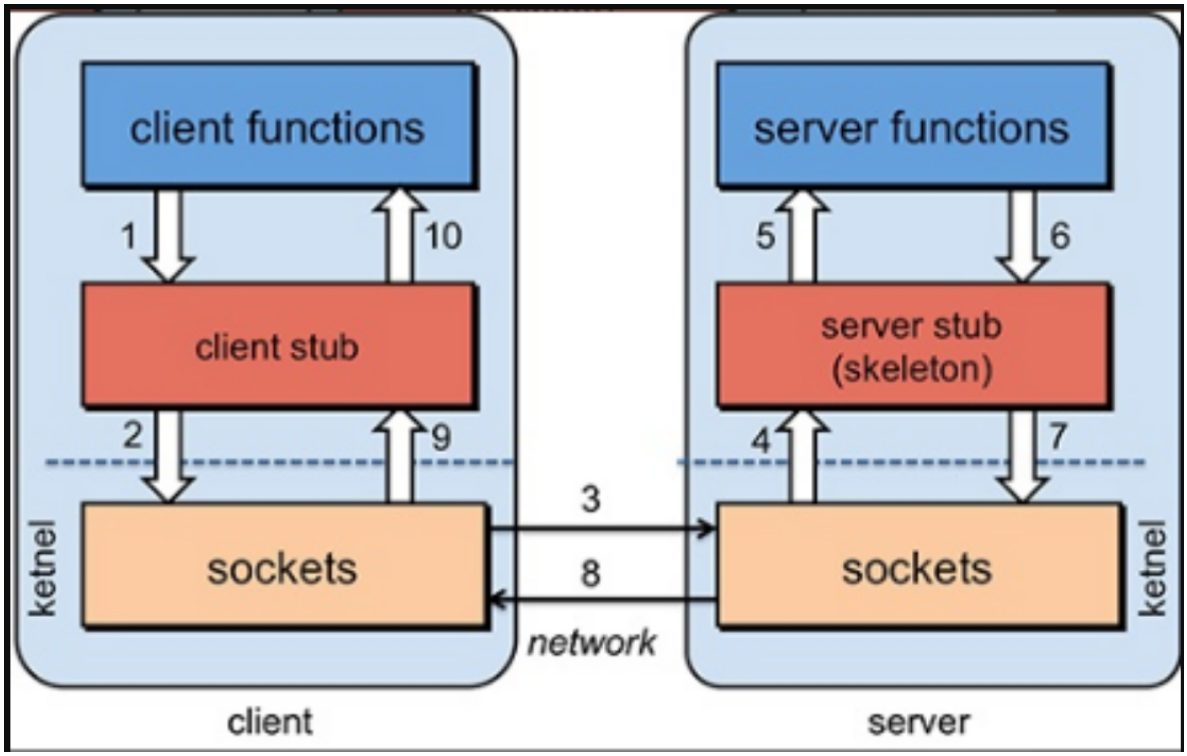
1.3.1 什么是RPC

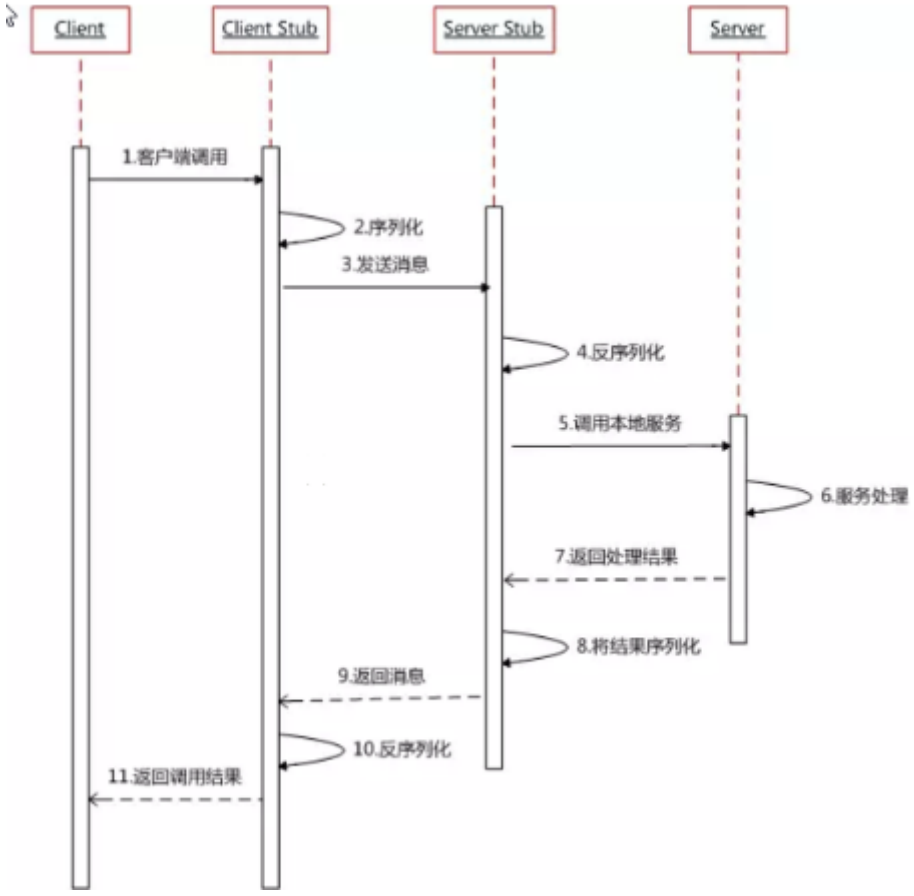
RPC【Remote Procedure Call】是指远程过程调用，是一种进程间通信方式，**是一种技术的思想**，而不是规范。它**允许程序调用另一个地址空间（通常是共享网络的另一台机器上）的过程或函数**，而不用程序员显式编码这个远程调用的细节。即程序员无论是调用本地的还是远程的函数，本质上编写的调用代码基本相同。

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；

概念进一步理解：https://www.jianshu.com/p/2accc2840a1b

1.3.2 RPC的基本原理





RPC两个核心模块：通讯，序列化。

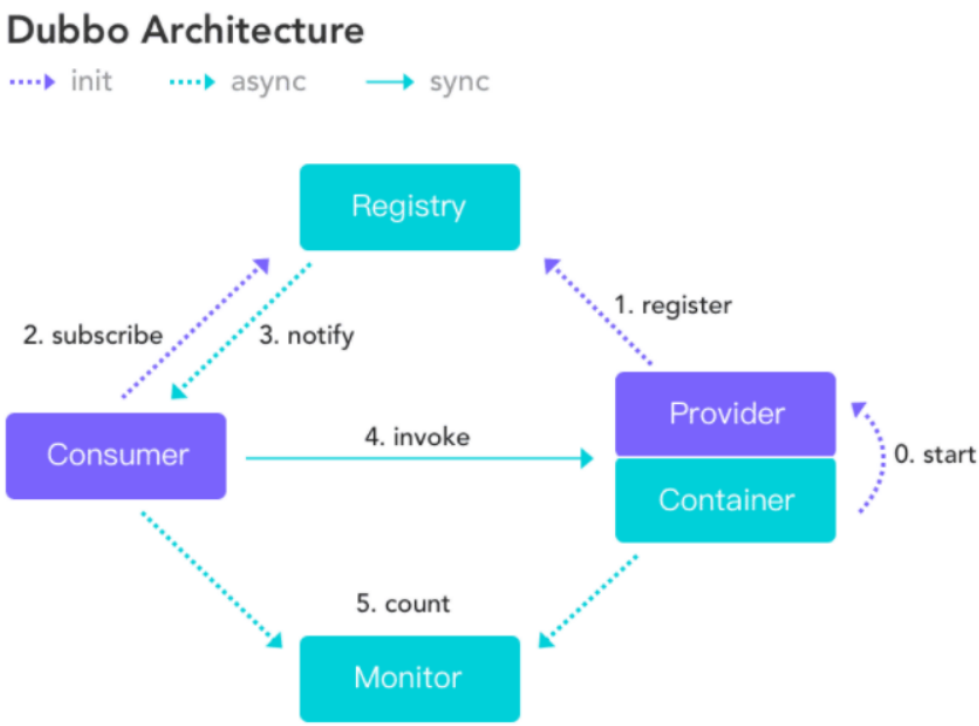
2 Dubbo和ZooKeeper安装

2.1 什么是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。

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

2.2 Dubbo的基本概念



- **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。

2.3 快速入门

参照官方文档：http://dubbo.apache.org/zh-cn/docs/user/quick-start.html

2.4 环境搭建

2.4.1 下载安装ZooKeeper

地址：http://mirror.bit.edu.cn/apache/zookeeper/zookeeper-3.4.14/

下载zookeeper-3.4.14.tar.gz ===》解压

进入到bin目录下：

编辑zkServer.cmd：

在endlocal上一行加上：pause，否则直接运行zkServer就会闪退。

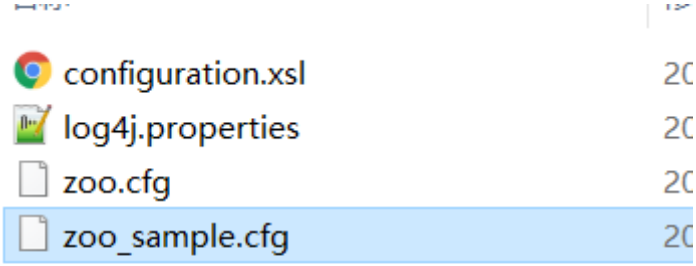
```
setlocal
call "%~dp0zkEnv.cmd"

set ZOOMAIN=org.apache.zookeeper.server.quorum.QuorumPee
echo on
call %JAVA% "-Dzookeeper.log.dir=%ZOO_LOG_DIR%" "-Dzooke
pause
endlocal
```

这时候我们再运行zkServer，会发现一个错误：

```
F:\zookeeper-3.4.14\bin>call "C:\Program Files\Java\jdk1.8.0_181"\bin\java "-Dzookeeper.log.dir=F:\zookeeper-3.4.14\b
.." "-Dzookeeper.root.logger=INFO,CONSOLE" -cp "F:\zookeeper-3.4.14\bin\..\build\classes;F:\zookeeper-3.4.14\bin\..\b
d\lib\*;F:\zookeeper-3.4.14\bin\..\*;F:\zookeeper-3.4.14\bin\..\lib\*;F:\zookeeper-3.4.14\bin\..\conf" org.apache.zoo
per.server.quorum.QuorumPeerMain "F:\zookeeper-3.4.14\bin\..\conf\zoo.cfg"
2020-02-27 16:17:11,121 [myid:] - INFO  [main:QuorumPeerConfig@136] - Reading configuration from: F:\zookeeper-3.4.14
\..\conf\zoo.cfg
2020-02-27 16:17:11,124 [myid:] - ERROR [main:QuorumPeerMain@88] - Invalid config, exiting abnormally
org.apache.zookeeper.server.quorum.QuorumPeerConfig$ConfigException: Error processing F:\zookeeper-3.4.14\bin\..\conf
oo.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: F:\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

F:\zookeeper-3.4.14\bin>pause
请按任意键继续. . .
```

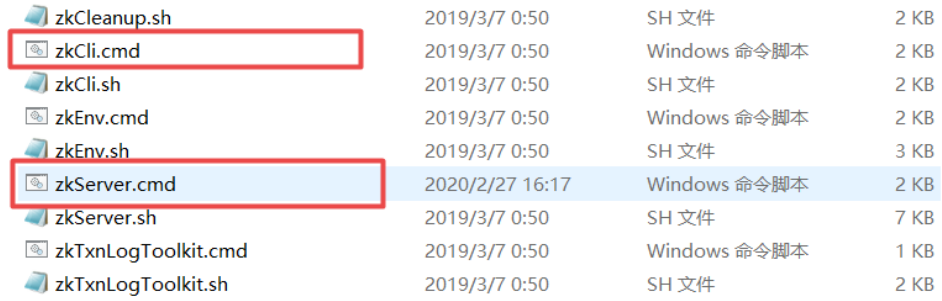
发现是conf下少了zoo.cfg；

我们进入conf，将已有的zoo_sample.cfg复制一份，改个名为zoo.cfg。



- 使用cmd进行测试

  先开启zkServer，再开启zkCli，记得用管理员身份打开。



- 在zkCli进行操作

```
ls  /
[zookeeper]
[zk: localhost:2181(CONNECTED) 1] ls /
[zookeeper]
[zk: localhost:2181(CONNECTED) 2] create -e /geek 123
Created /geek
[zk: localhost:2181(CONNECTED) 3] ls /
[zookeeper, geek]
[zk: localhost:2181(CONNECTED) 4] get /geek
123
cZxid = 0x6
ctime = Thu Feb 27 16:37:26 CST 2020
mZxid = 0x6
mtime = Thu Feb 27 16:37:26 CST 2020
pZxid = 0x6
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x10000ac47c30001
dataLength = 3
numChildren = 0
```

ls /：列出zookeeper根下保存的所有节点

create –e /geek123：创建一个geek节点，值为123

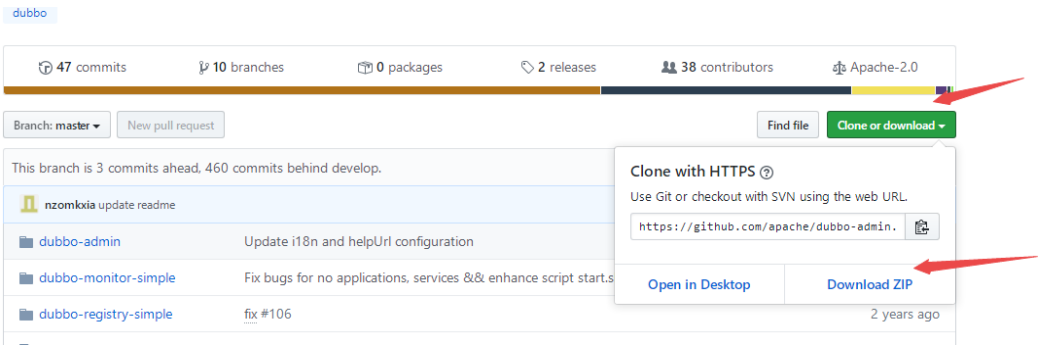
get /geek：获取/geek节点的值

2.4.2 下载安装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。

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。

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



解压，发现是一个Maven项目。

- 进入目录dubbo-admin-master\dubbo-admin\src\main\resources

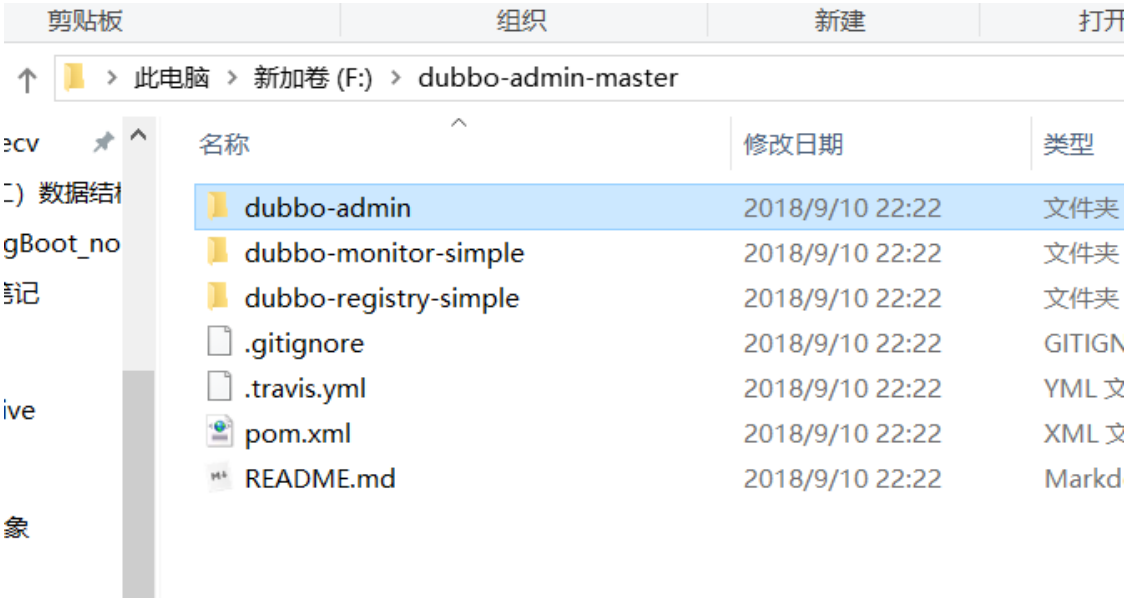
查看一些配置：

```
# limitations under the License.
#

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
# 注册中心的地址
dubbo.registry.address=zookeeper://127.0.0.1:2181
```

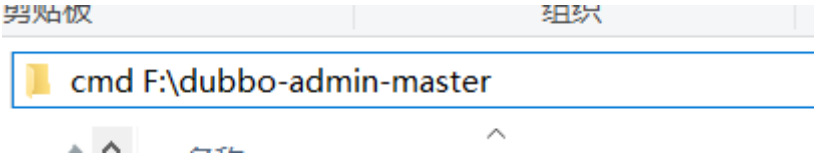
最后的地址和Dubbo的一样。

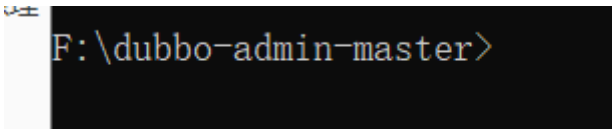
- **在项目目录下**打包dubbo-admin

  项目目录：

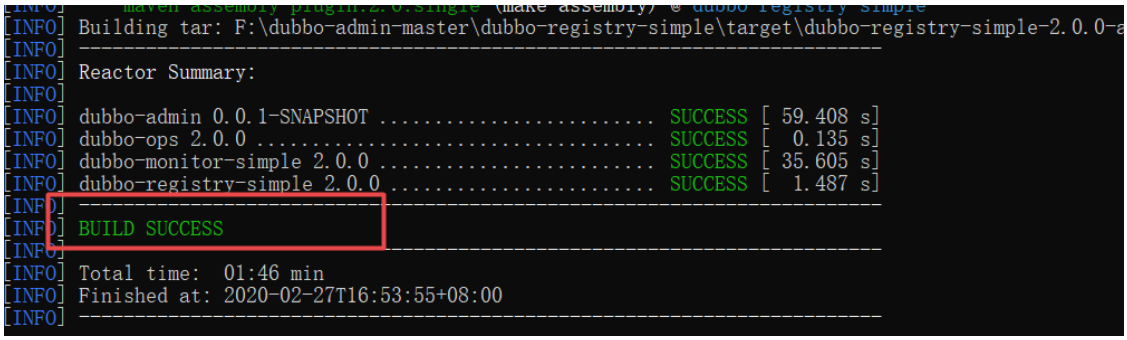


  进入命令行：

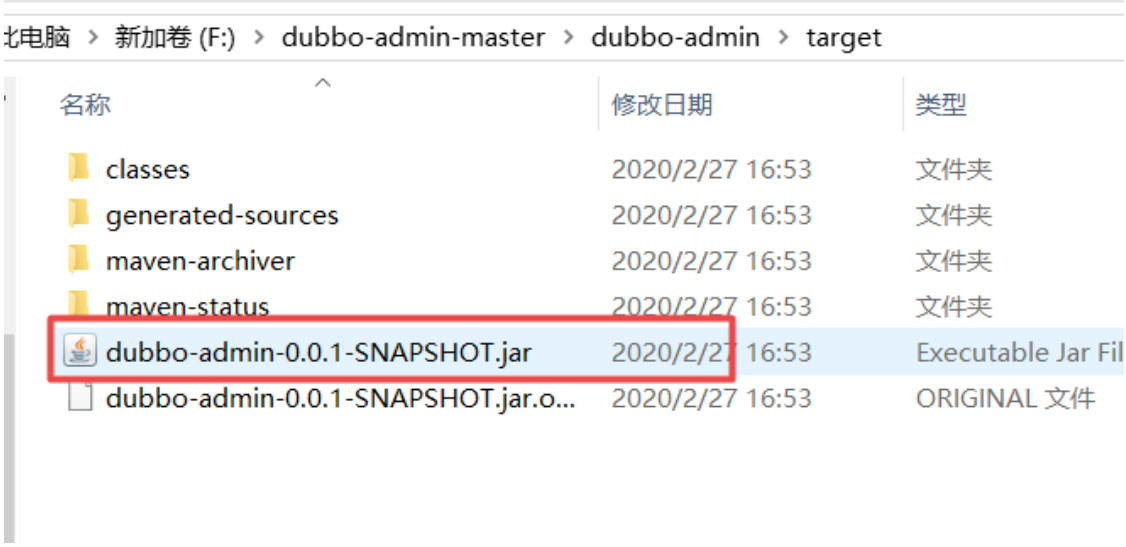




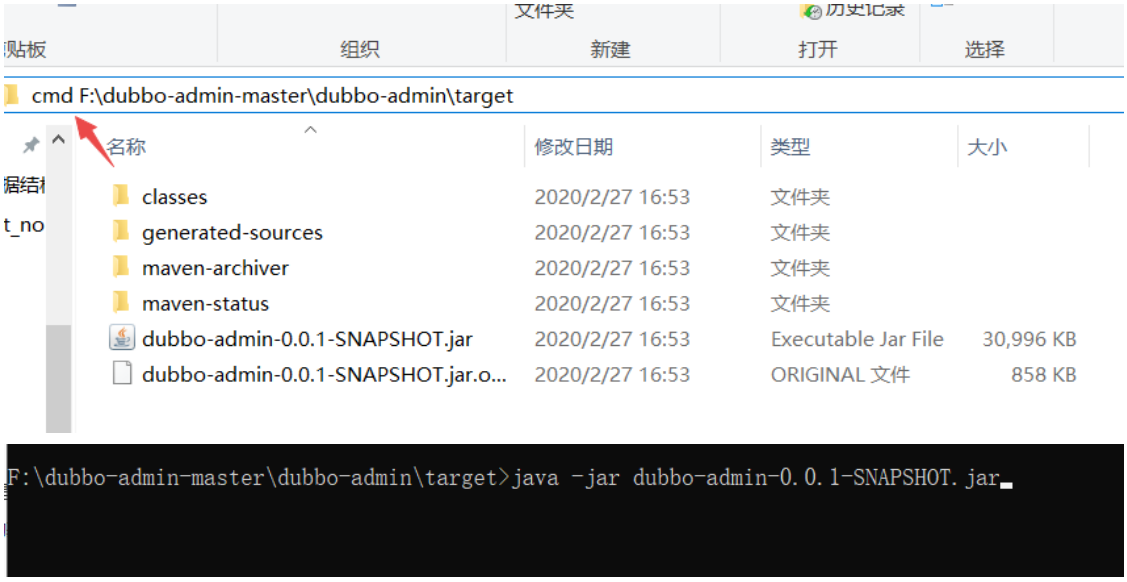
打包命令：mvn clean package -Dmaven.test.skip=true



打包成功！

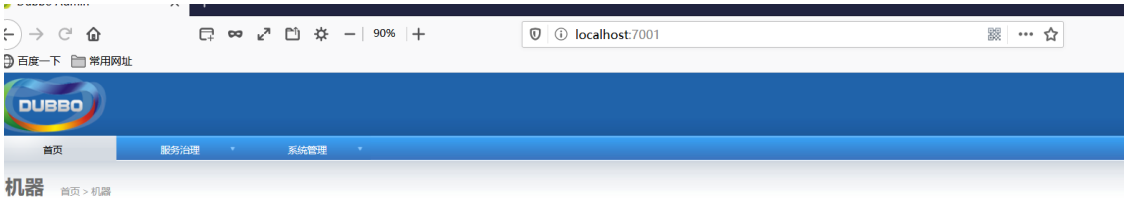


- 执行打包完的jar包
- 打开zkServer.cmd。【一定要打开ZooKeeper服务！】



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

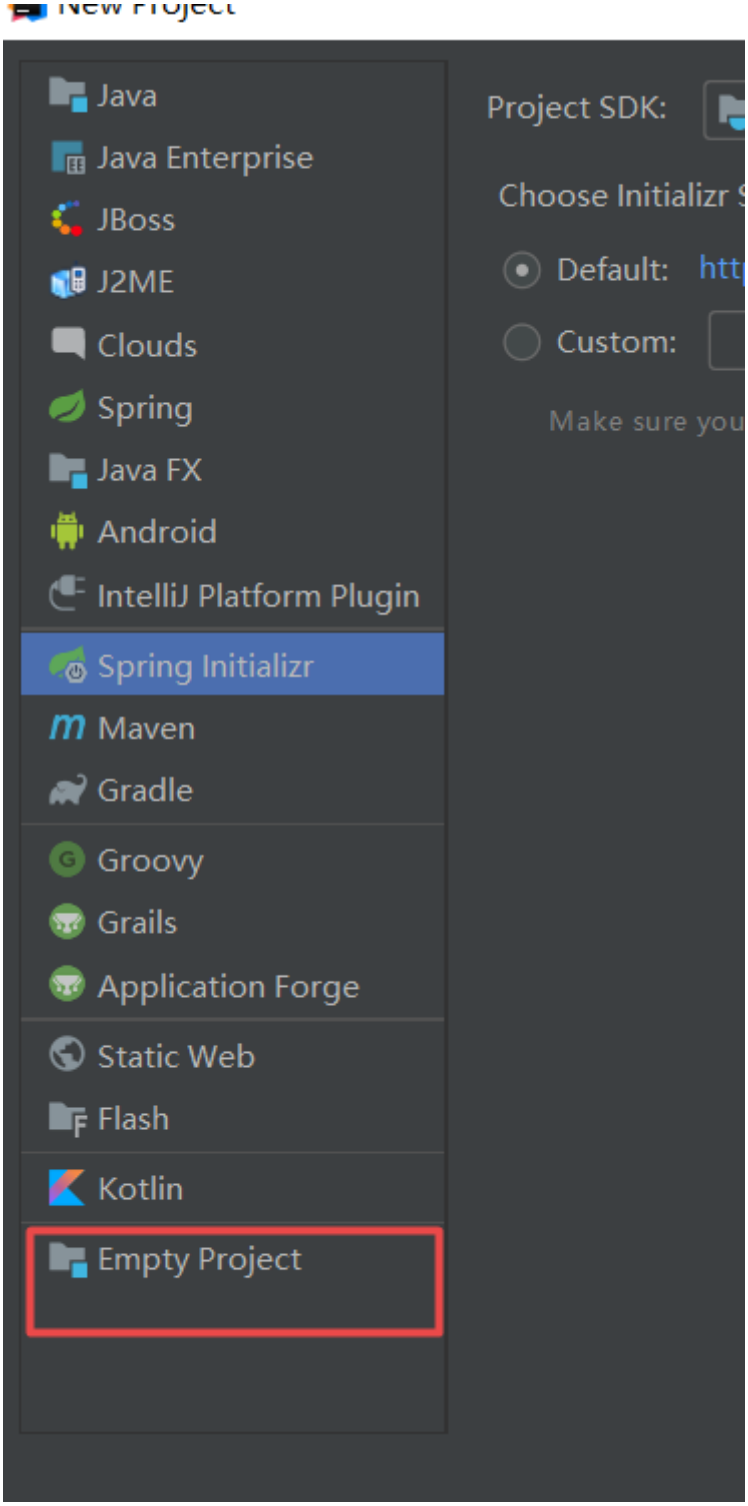
- 去访问一下 http://localhost:7001/ ， 这时候我们需要输入登录账户和密码，我们都是默认的root-root；


90%
localhost:7001
百度一下
常用网址
DUBBO
首页
服务治理
系统管理
机器
首页 > 机器

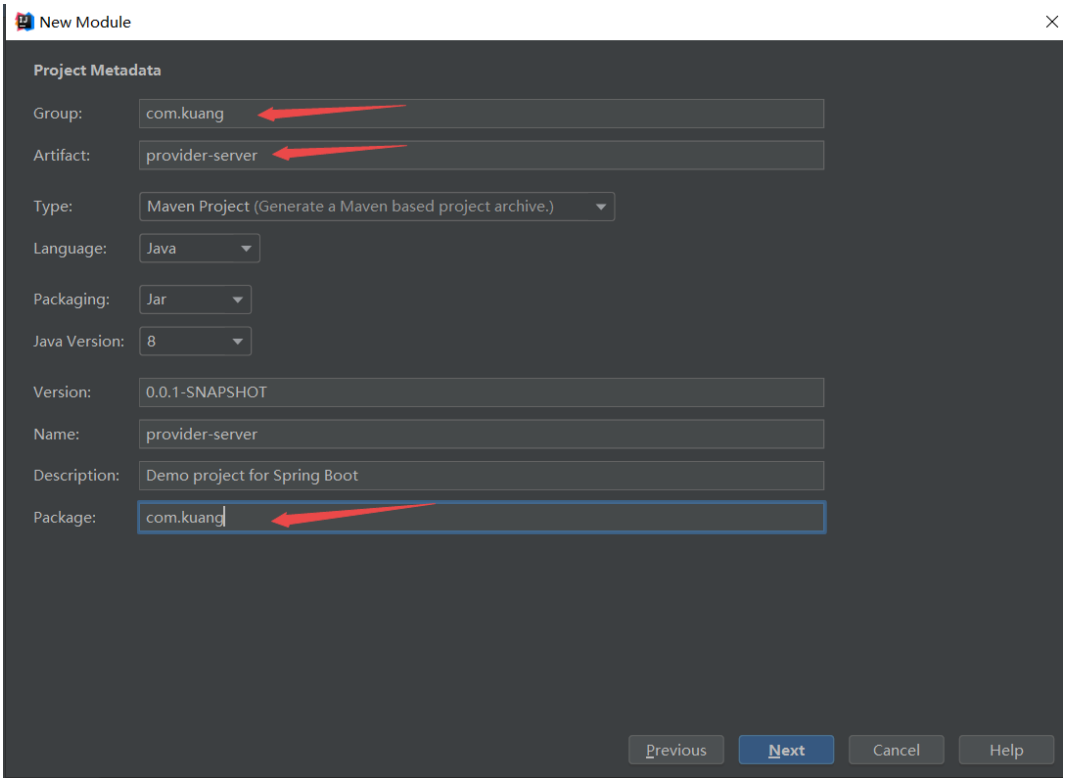

服务名
应用名
机器IP
SEARCH
新增: 提供者 路由规则 动态配置 访问控制 权重调节 负载均衡 负责人
统计: 服务数:0 应用数:0 提供者数:0 消费者数:0

3 整合项目

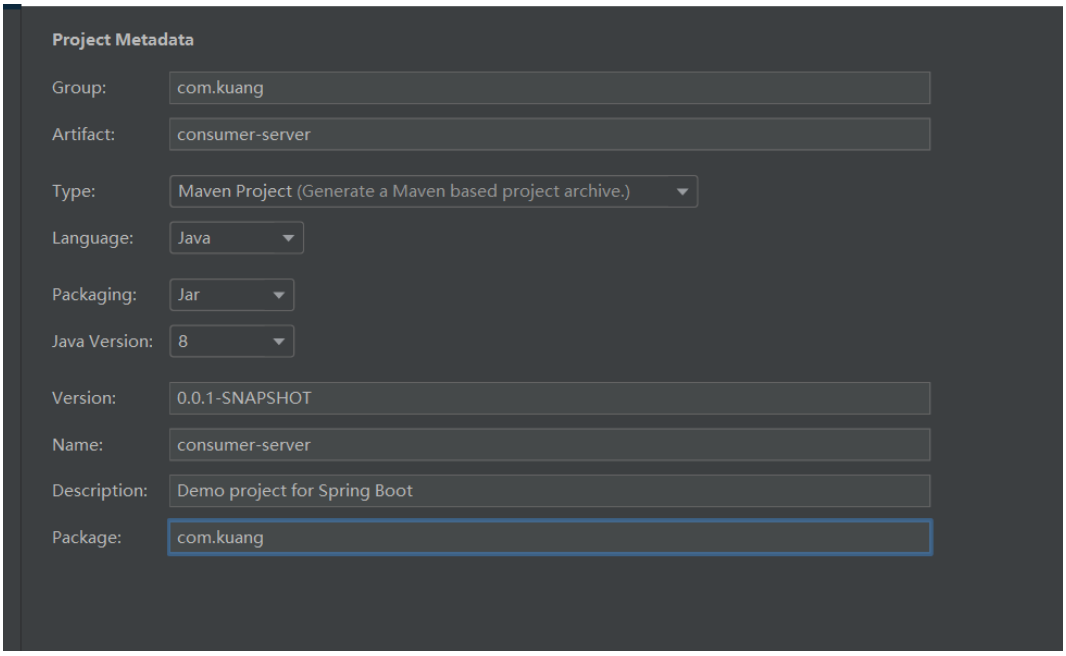
- 空的项目



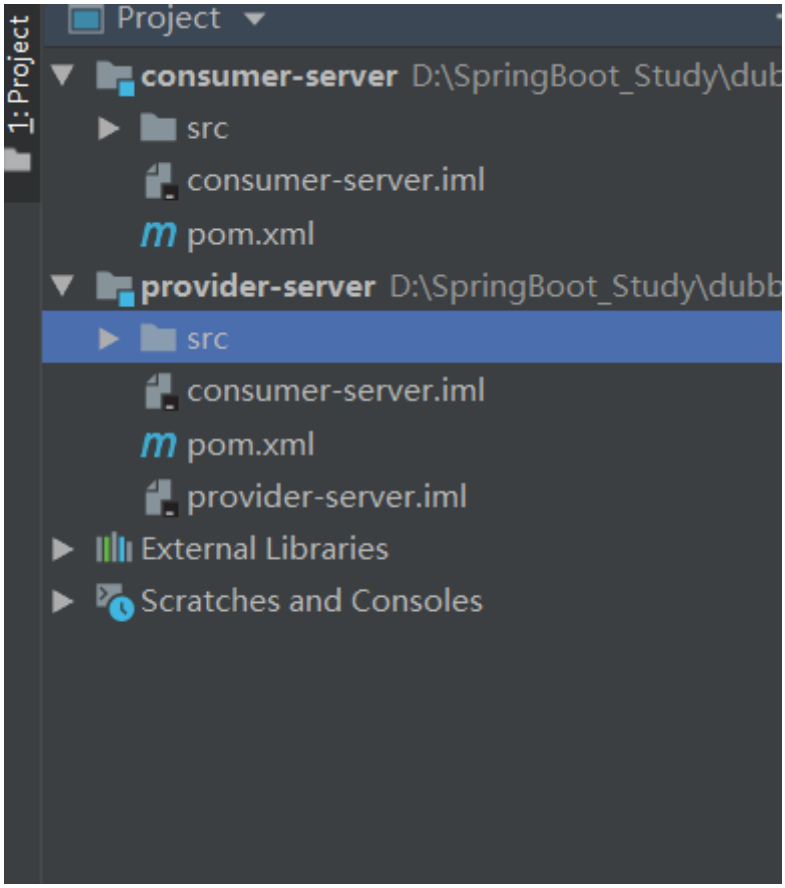
- 在该项目下创建两个SpringBoot的Module，都添加Web支持
  - 提供者（Provider）



- 购买者（Consumer）



注意后面选择放的地方的时候，一定不要放在provider-server下。



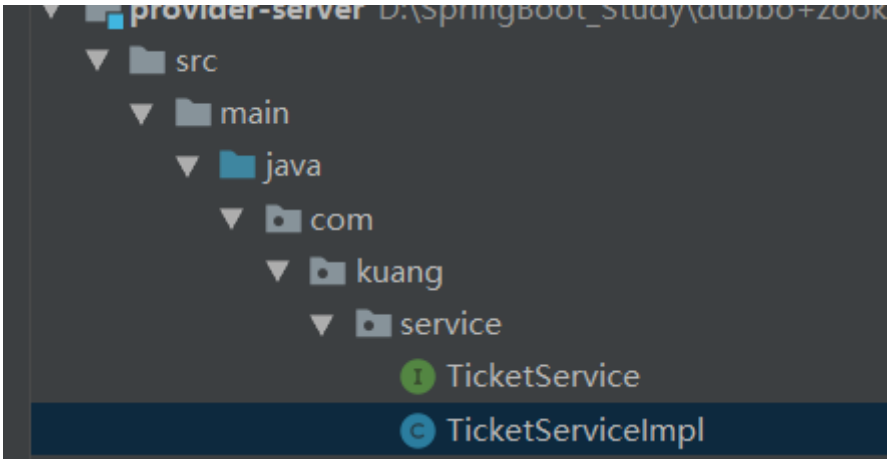
- 添加依赖

```
<!--导入Dubbo和ZooKeeper的依赖-->
<!-- https://mvnrepository.com/artifact/org.apache.dubbo/dubbo-spring-boot-starter -->
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo-spring-boot-starter</artifactId>
    <version>2.7.3</version>
</dependency>
<!-- 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，这是导入Zookeeper的坑-->
<!-- 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>
```

3.1 提供者

在provider-server项目下：




```java
package com.kuang.service;

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

```java
package com.kuang.service;

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

//ZooKeeper：服务注册与发现

@Service //可以被扫描到，在项目一启动就自动注册到注册中心 import
org.apache.dubbo.config.annotation.Service;
@Component //使用了Dubbo，最好不要用@Service注解
public class TicketServiceImpl implements TicketService {
    @Override
    public String getTicket() {
        return "Study Dubbo + ZooKeeper";
    }
}
```

注意：@service不要导错包===》是要导Dubbo中的。

配置application.properties:

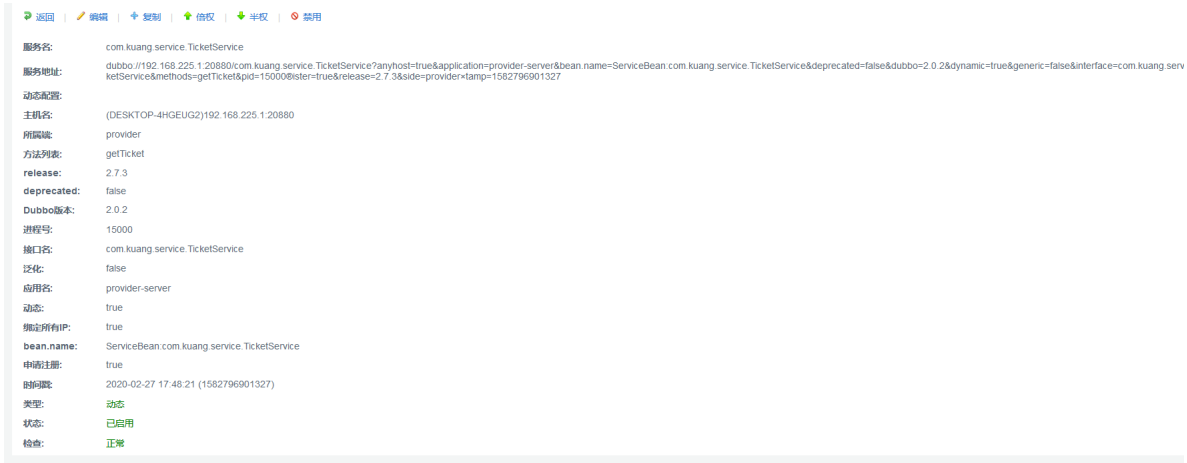
```properties
server.port=8001


# 服务应用名字
dubbo.application.name=provider-server
# 注册中心地址
dubbo.registry.address=zookeeper://127.0.0.1:2181
# 哪些服务要被注册（扫描包）
dubbo.scan.base-packages=com.kuang.service
```

启动ZooKeeper服务：zkServer.cmd

启动主启动类。

打开浏览器：http://localhost:7001/

发现提供者已经监听到了。



3.2 消费者

导入和上面一样的依赖。

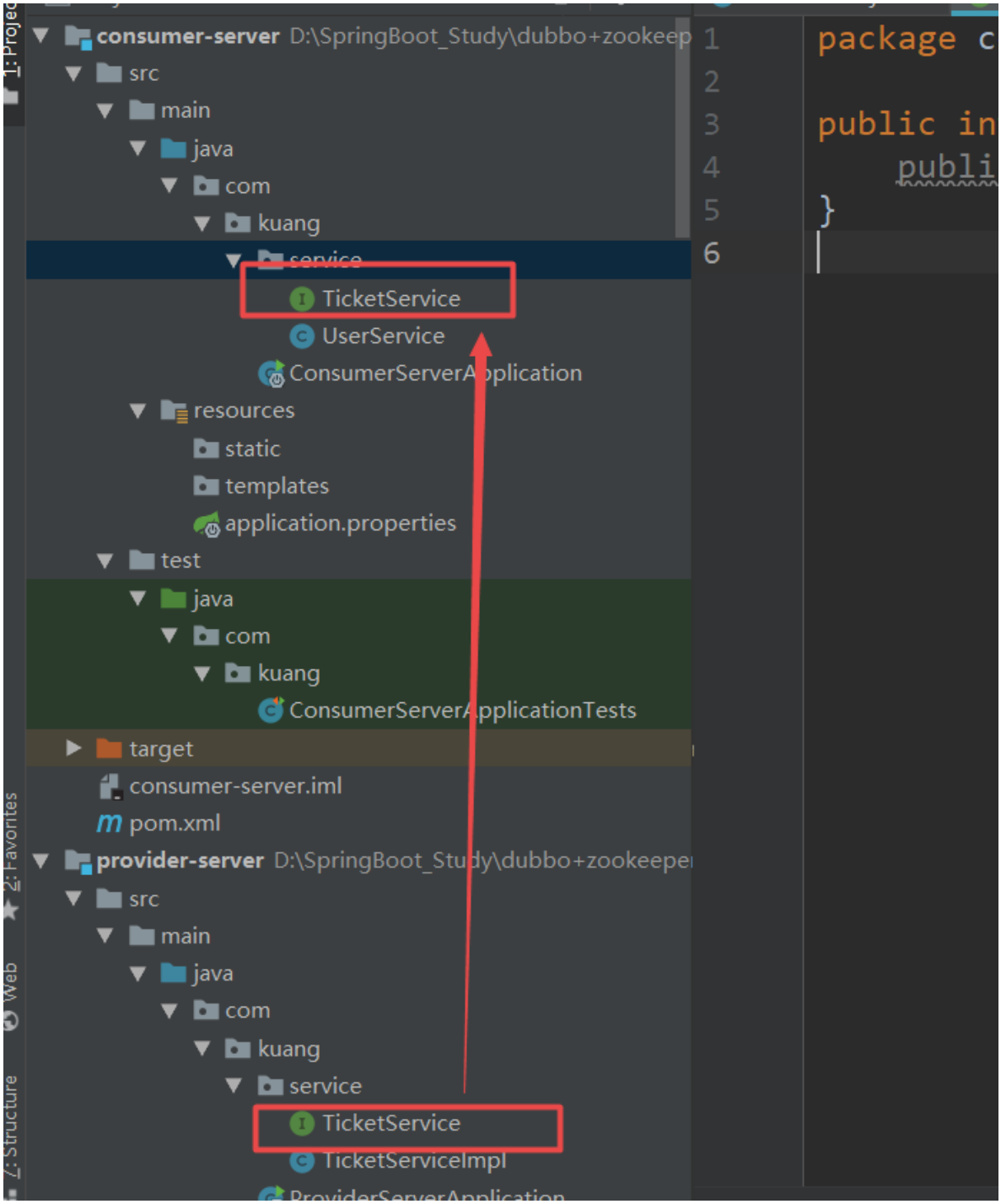
```java
package com.kuang.service;

import org.apache.dubbo.config.annotation.Reference;
import org.springframework.stereotype.Service;

@Service //放到容器中注册，导入的是Spring的包
public class UserService {
    //想拿到provider-server提供的票
    //要去注册中心拿
```

```
    @Reference //引用,可以使用Pom坐标或者 定义路径相同的接口名
    TicketService ticketService;

    public void buyTicket() {
        String ticket = ticketService.getTicket();
        System.out.println("在注册中心拿到====>" + ticket);
    }

}
```

application,properties:

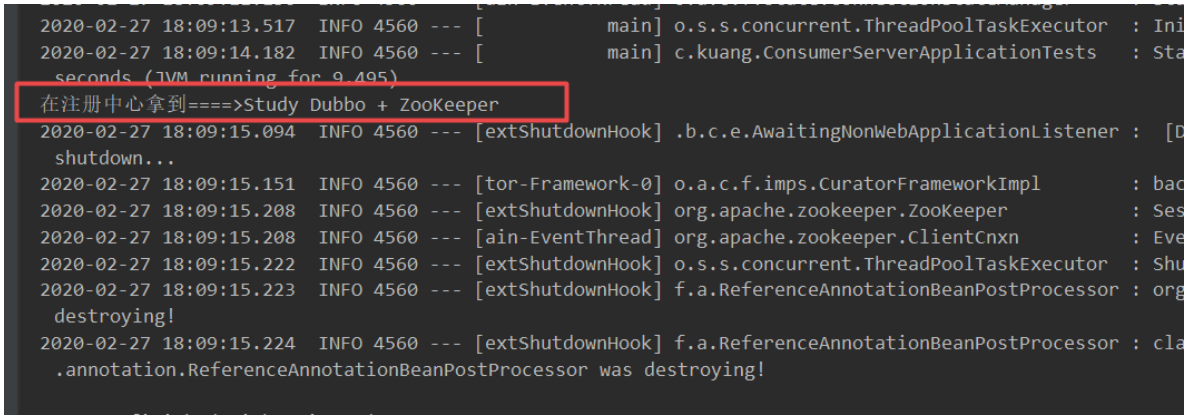
```
server.port=8002

# 消费者去哪里拿服务【去注册中心拿】
# 消费者拿服务时，必须暴露自己的名字
dubbo.application.name=consumer-server
# 注册中心的地址
dubbo.registry.address=zookeeper://127.0.0.1:2181

# 消费者不需要写扫描包，因为自己只是去服务那边“取货”，扫描包是为了“发货”
```

这里我们使用第二种方法进行引用：定义路径相同的接口名===》将service接口复制过来，路径要一样【正常开发中我们不会这么做，这里只是展示一下】



测试一下：

```java
@SpringBootTest
class ConsumerServerApplicationTests {
    @Autowired
    private UserService userService;

    @Test
    void contextLoads() {
        userService.buyTicket();
    }

}
```



测试成功！我们这里的TicketService并没有实现类，但是却可以执行，是因为远程引入了。

3.3 总结

前提是：ZooKeeper服务已开启

1、提供者提供服务

- 导入依赖
- 配置注册中心的地址，服务发现名，要扫描的包（要发的货）
- 在想要被注册的服务（货源）上面增加注解@Service【Dubbo包下的】

2、消费者如何消费

- 导入依赖
- 配置注册中心的地址，配置自己的服务名
- 从远程注入服务@Reference