

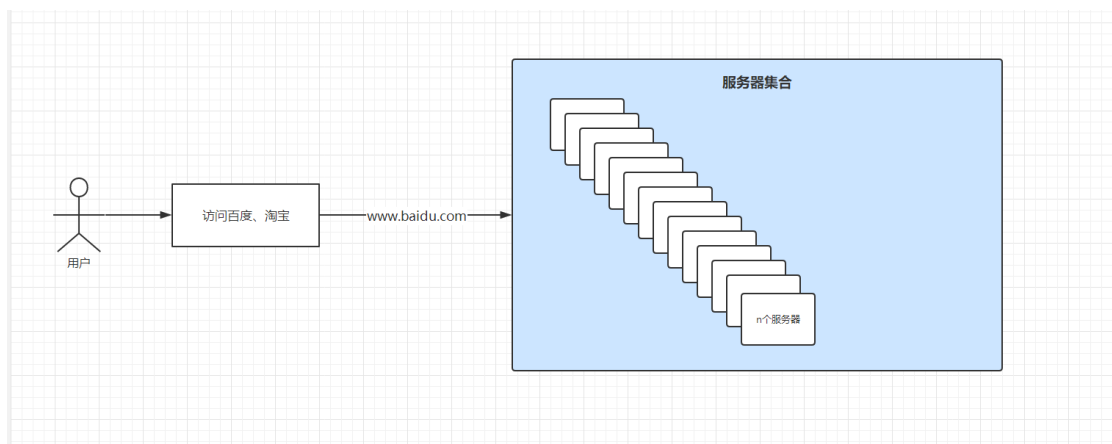
分布式 Dubbo+ZooKeeper

1 分布式理论

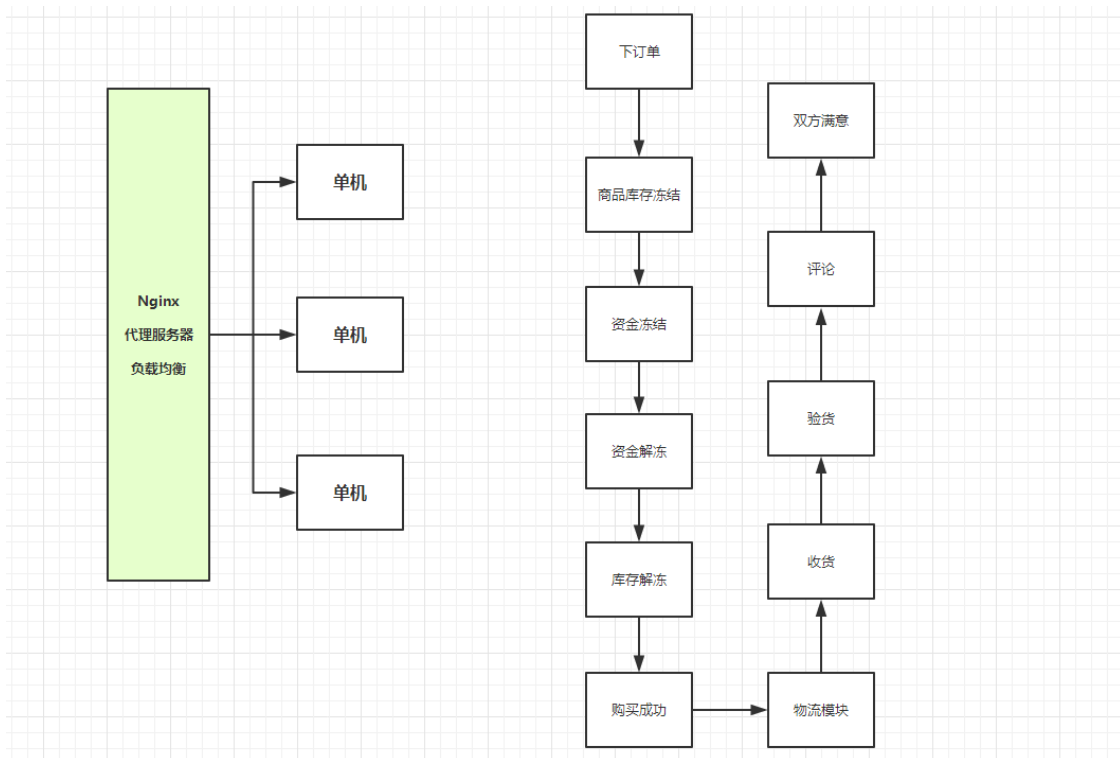
1.1 什么是分布式系统

- 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。



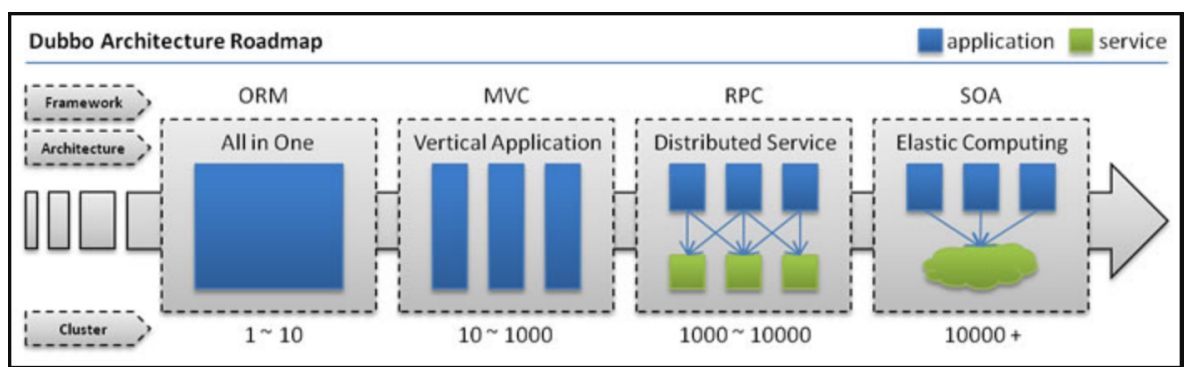
上面的流程中可以分为多个模块，每个模块都放在不同的服务器上，又可以根据模块的复杂度安排服务器（比如订单系统简单，那么可以把所有订单放在一台服务器上；物流和支付系统比较慢，就放在多台服务器上），这就将原来的单机系统分为多个服务器。

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。

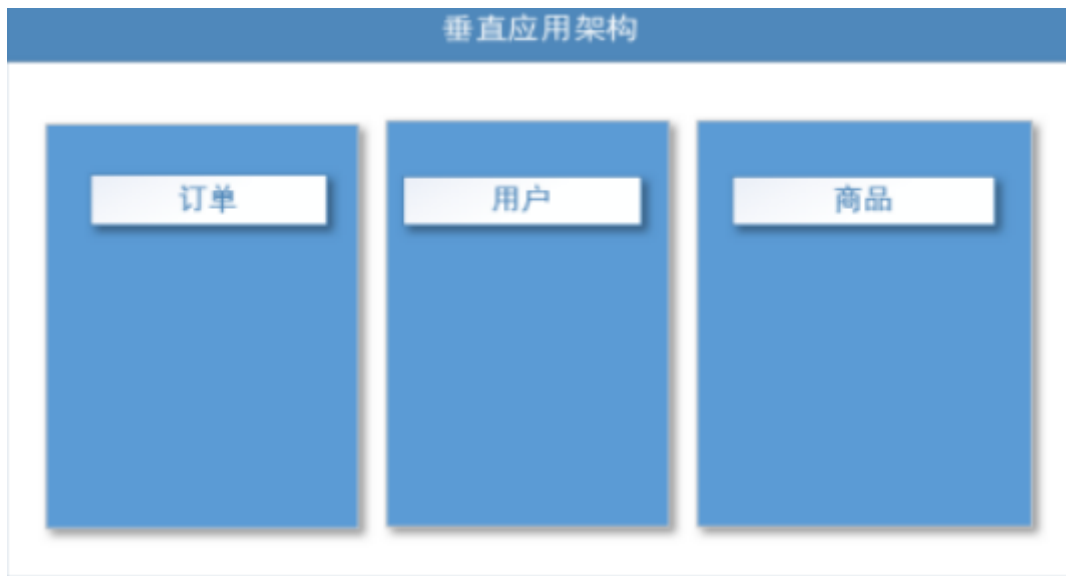


适用于小型网站，小型管理系统，将所有功能都部署到一个功能里，简单易用。

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。

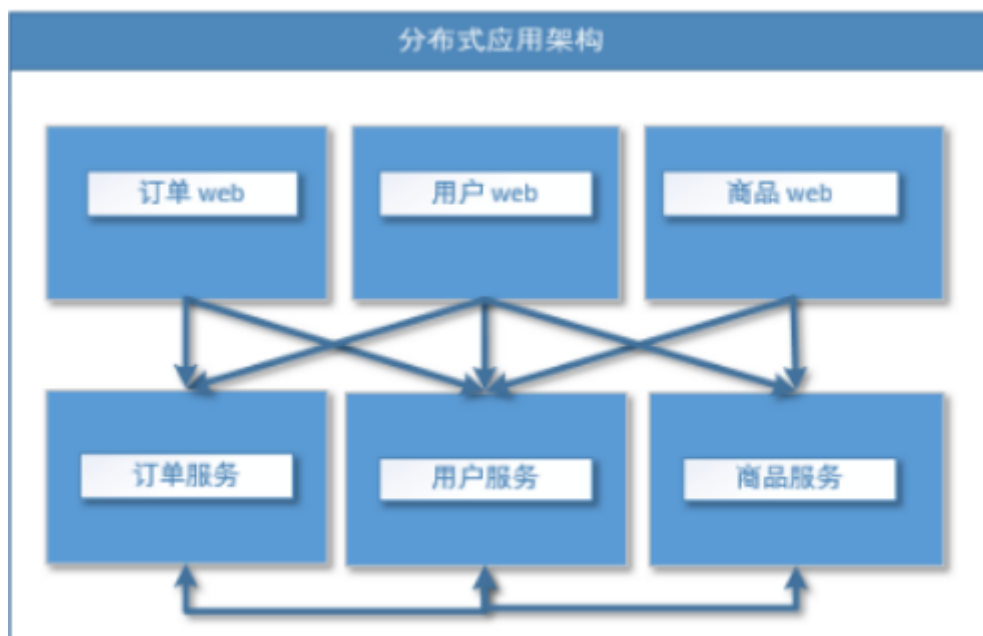


通过切分业务来实现各个模块独立部署，降低了维护和部署的难度，团队各司其职更易管理，性能扩展也更方便，更有针对性。

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

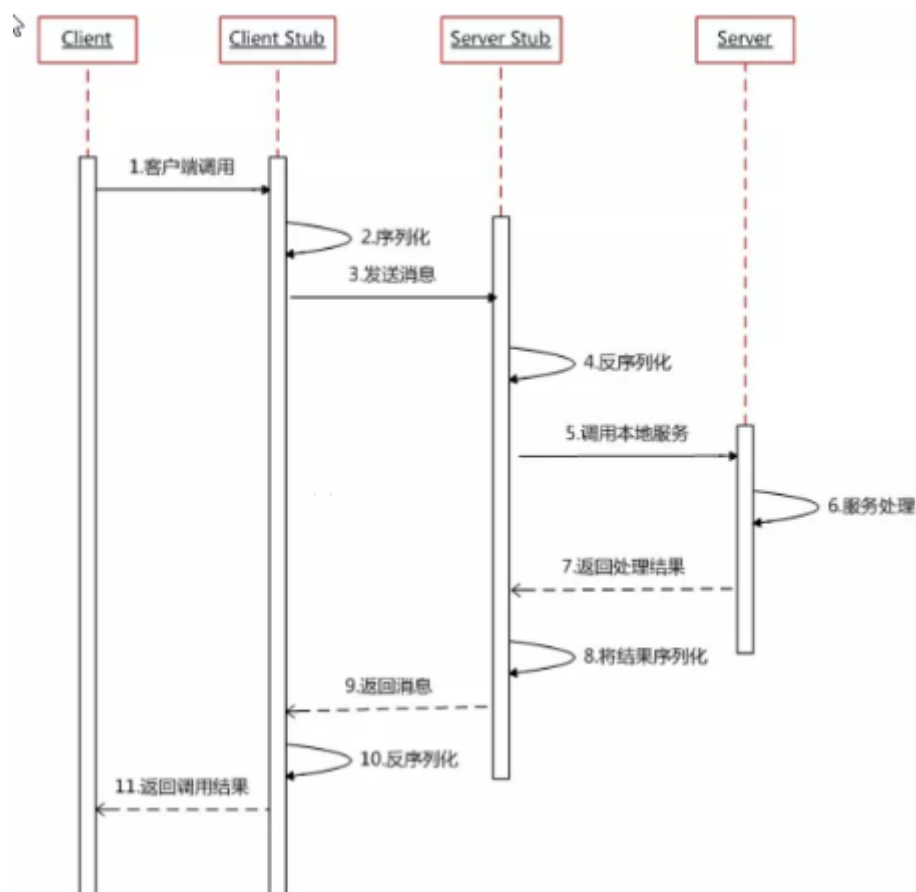
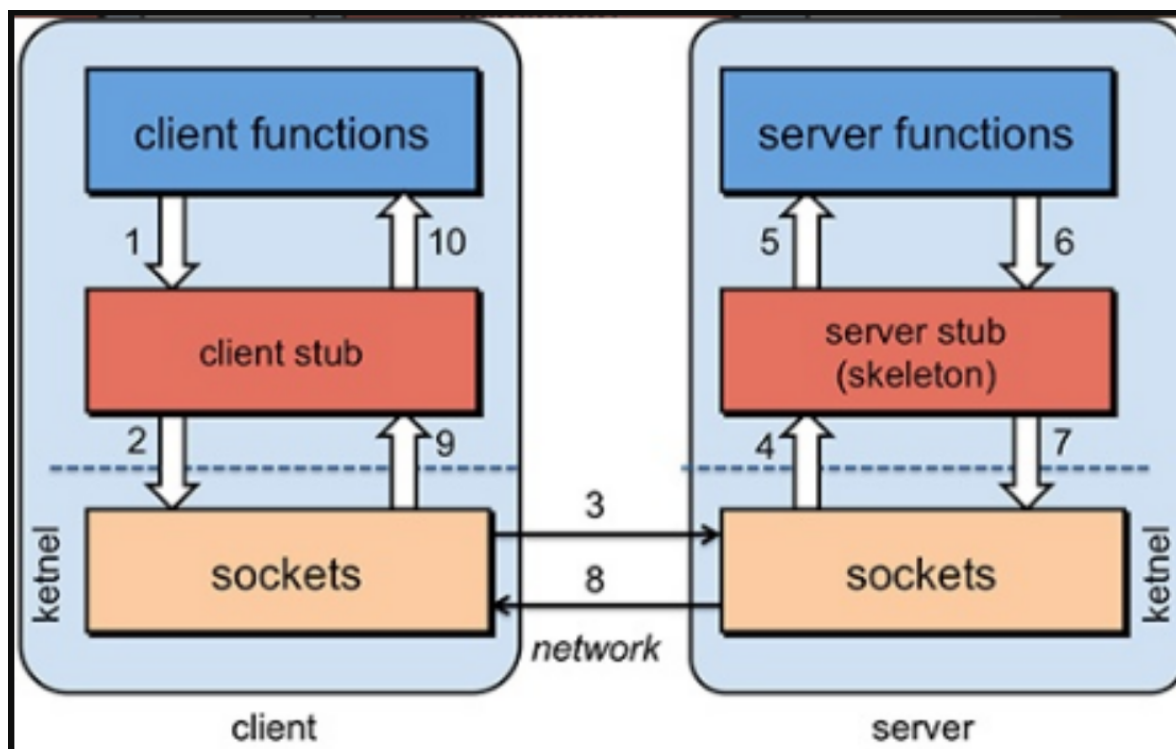
1.3.1 什么是RPC

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。

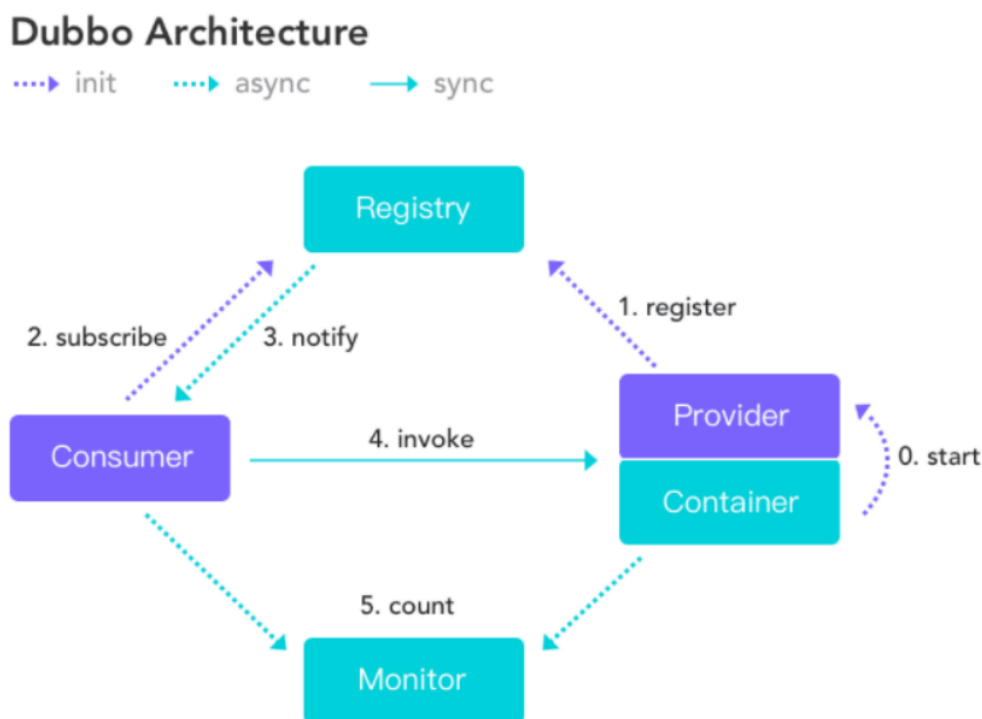
2 Dubbo和ZooKeeper安装

2.1 什么是Dubbo

Apache Dubbo | '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。

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。

2.3 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"

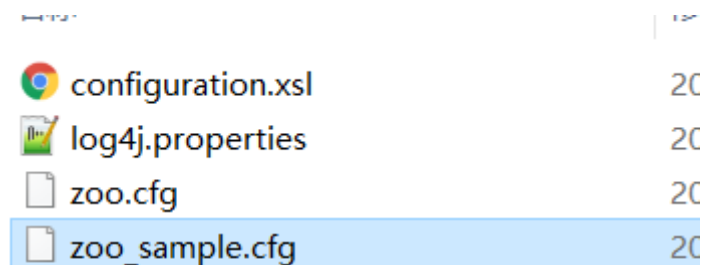
set ZOOMAIN=org.apache.zookeeper.server.quorum.QuorumPeer
echo on
call %JAVA% "-Dzookeeper.log.dir=%ZOO_LOG_DIR%" "-Dzooke
pause
endlocal
```

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\bin\..
..\" "-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\..\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
bin\..\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
p.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。

zkCleanup.sh	2019/3/7 0:50	SH 文件	2 KB
zkCli.cmd	2019/3/7 0:50	Windows 命令脚本	2 KB
zkCli.sh	2019/3/7 0:50	SH 文件	2 KB
zkEnv.cmd	2019/3/7 0:50	Windows 命令脚本	2 KB
zkEnv.sh	2019/3/7 0:50	SH 文件	3 KB
zkServer.cmd	2020/2/27 16:17	Windows 命令脚本	2 KB
zkServer.sh	2019/3/7 0:50	SH 文件	7 KB
zkTxnLogToolkit.cmd	2019/3/7 0:50	Windows 命令脚本	1 KB
zkTxnLogToolkit.sh	2019/3/7 0:50	SH 文件	2 KB

- 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

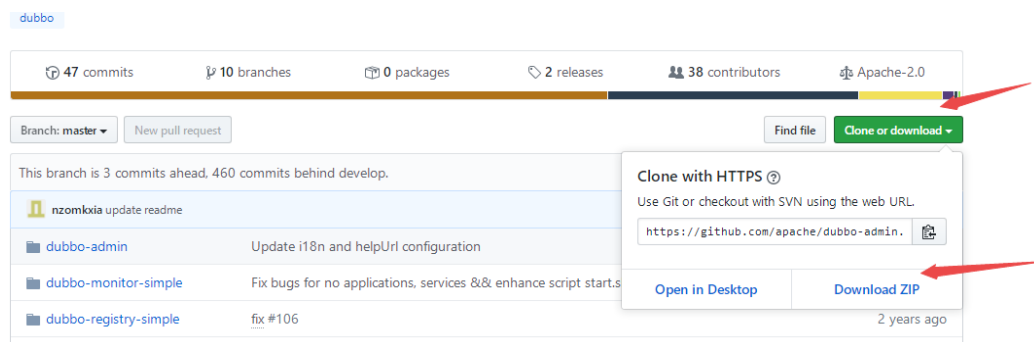
get /geek: 获取/geek节点的值

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

此电脑 > 新加卷 (F:) > dubbo-admin-master > dubbo-admin > src > main > resources				
名称	修改日期	类型	大小	
i18n	2018/9/10 22:22	文件夹		
static	2018/9/10 22:22	文件夹		
templates	2018/9/10 22:22	文件夹		
application.properties	2020/2/27 16:48	PROPERTIES 文件	2 KB	
dubbo-admin.xml	2018/9/10 22:22	XML 文档	2 KB	类型: PROPERTIES 文件 大小: 1.07 KB 修改日期: 2018/9/10 22:22
log4j.properties	2018/9/10 22:22	PROPERTIES 文件	2 KB	

查看一些配置:

```

5 # limitations under the License.
6 #
7
8 server.port=7001
9 spring.velocity.cache=false
10 spring.velocity.charset=UTF-8
11 spring.velocity.layout-url=/templates/default.vm
12 spring.messages.fallback-to-system-locale=false
13 spring.messages.basename=i18n/message
14 spring.root.password=root
15 spring.guest.password=guest
16 # 注册中心的地址
17 dubbo.registry.address=zookeeper://127.0.0.1:2181
18

```

最后的地址和Dubbo的一样。

- 在项目目录下打包dubbo-admin

项目目录:

剪贴板	组织	新建	打开
↑ > 此电脑 > 新加卷 (F:) > dubbo-admin-master			
名称	修改日期	类型	
dubbo-admin	2018/9/10 22:22	文件夹	
dubbo-monitor-simple	2018/9/10 22:22	文件夹	
dubbo-registry-simple	2018/9/10 22:22	文件夹	
.gitignore	2018/9/10 22:22	GITIGN	
.travis.yml	2018/9/10 22:22	YML 文	
pom.xml	2018/9/10 22:22	XML 文	
README.md	2018/9/10 22:22	Markd	

进入命令行:

剪贴板	组织
cmd F:\dubbo-admin-master	

```
F:\dubbo-admin-master>
```

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

```
INFO] Building tar: F:\dubbo-admin-master\dubbo-registry-simple\target\dubbo-registry-simple-2.0.0-a
INFO] Reactor Summary:
INFO] dubbo-admin 0.0.1-SNAPSHOT ..... SUCCESS [ 59.408 s]
INFO] dubbo-ops 2.0.0 ..... SUCCESS [ 0.135 s]
INFO] dubbo-monitor-simple 2.0.0 ..... SUCCESS [ 35.605 s]
INFO] dubbo-registry-simple 2.0.0 ..... SUCCESS [ 1.487 s]
INFO] BUILD SUCCESS
INFO] Total time: 01:46 min
INFO] Finished at: 2020-02-27T16:53:55+08:00
INFO]
```

打包成功!

此电脑 > 新加卷 (F:) > dubbo-admin-master > dubbo-admin > target

名称	修改日期	类型
classes	2020/2/27 16:53	文件夹
generated-sources	2020/2/27 16:53	文件夹
maven-archiver	2020/2/27 16:53	文件夹
maven-status	2020/2/27 16:53	文件夹
dubbo-admin-0.0.1-SNAPSHOT.jar	2020/2/27 16:53	Executable Jar File
dubbo-admin-0.0.1-SNAPSHOT.jar.o...	2020/2/27 16:53	ORIGINAL 文件

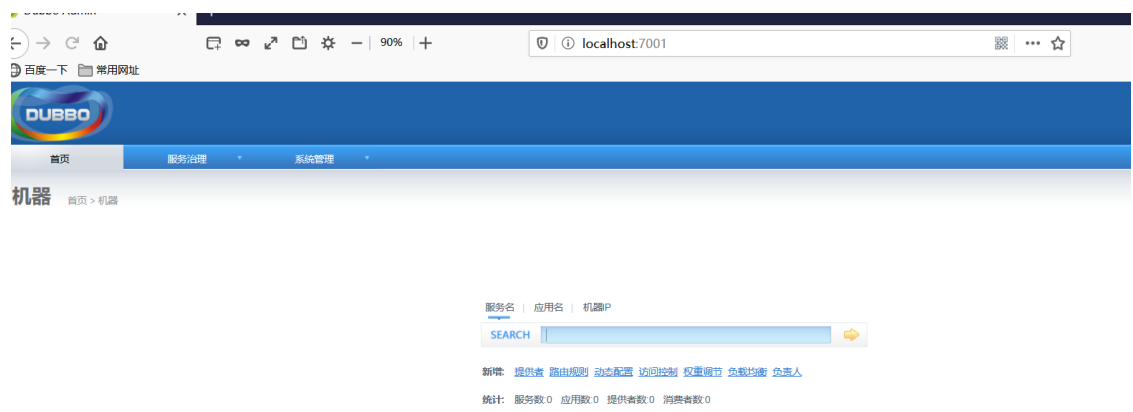
- 执行打包完的jar包
- 打开zkServer.cmd。【一定要打开ZooKeeper服务!】

名称	修改日期	类型	大小
classes	2020/2/27 16:53	文件夹	
generated-sources	2020/2/27 16:53	文件夹	
maven-archiver	2020/2/27 16:53	文件夹	
maven-status	2020/2/27 16:53	文件夹	
dubbo-admin-0.0.1-SNAPSHOT.jar	2020/2/27 16:53	Executable Jar File	30,996 KB
dubbo-admin-0.0.1-SNAPSHOT.jar.o...	2020/2/27 16:53	ORIGINAL 文件	858 KB

```
F:\dubbo-admin-master\dubbo-admin\target>java -jar dubbo-admin-0.0.1-SNAPSHOT.jar
```

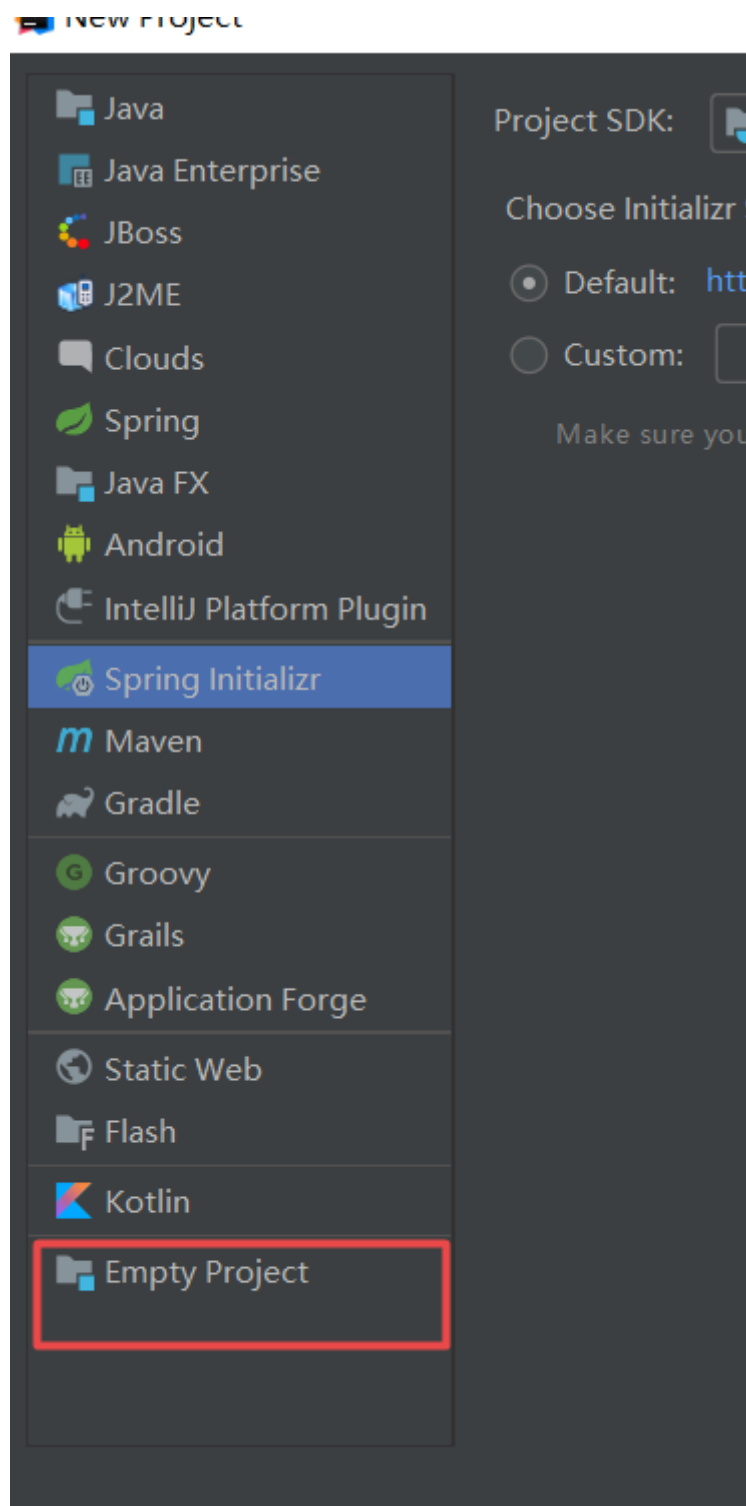
java -jar dubbo-admin-0.0.1-SNAPSHOT.jar

- 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;



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)

New Module

Project Metadata

Group: com.kuang

Artifact: provider-server

Type: Maven Project (Generate a Maven based project archive.)

Language: Java

Packaging: Jar

Java Version: 8

Version: 0.0.1-SNAPSHOT

Name: provider-server

Description: Demo project for Spring Boot

Package: com.kuang

Previous Next Cancel Help

○ 购买者 (Consumer)

Project Metadata

Group: com.kuang

Artifact: consumer-server

Type: Maven Project (Generate a Maven based project archive.)

Language: Java

Packaging: Jar

Java Version: 8

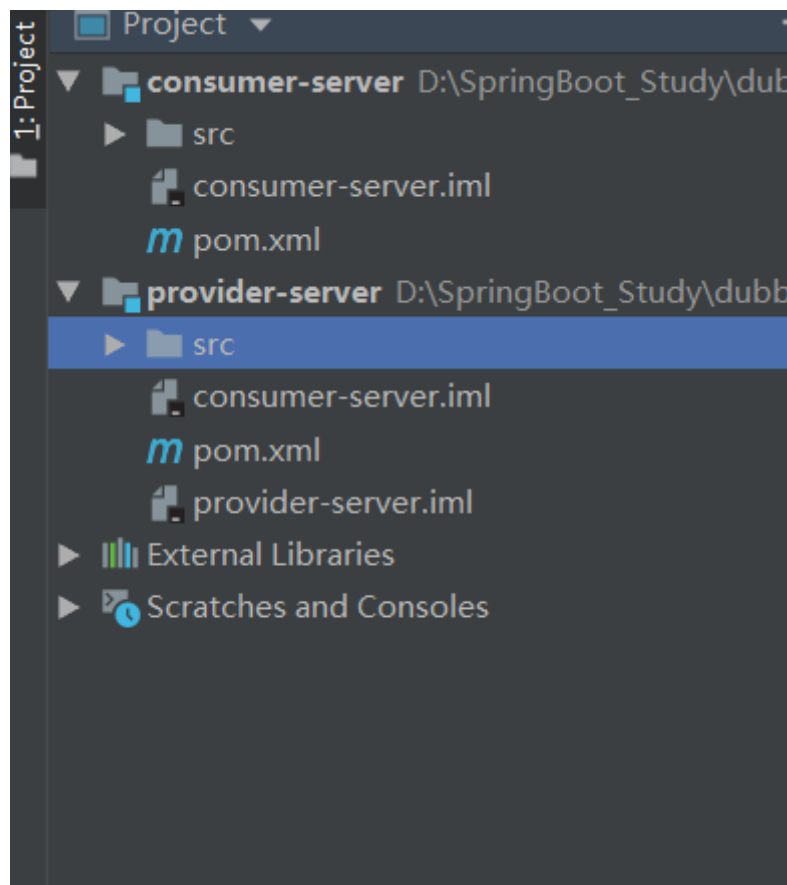
Version: 0.0.1-SNAPSHOT

Name: consumer-server

Description: Demo project for Spring Boot

Package: com.kuang

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

```
1  <!--导入Dubbo和ZooKeeper的依赖-->
2  <!-- https://mvnrepository.com/artifact/org.apache.dubbo/dubbo-spring-
3  boot-starter -->
4  <dependency>
5      <groupId>org.apache.dubbo</groupId>
6      <artifactId>dubbo-spring-boot-starter</artifactId>
7      <version>2.7.3</version>
8  </dependency>
9  <!-- https://mvnrepository.com/artifact/com.github.sgroschupf/zkclient
10 -->
11 <dependency>
12     <groupId>com.github.sgroschupf</groupId>
13     <artifactId>zkclient</artifactId>
14     <version>0.1</version>
15 </dependency>
16 <!--新版的坑: ZooKeeper及其依赖包, 解决日志冲突, 这是导入ZooKeeper的坑-->
17 <!-- 引入zookeeper -->
18 <dependency>
19     <groupId>org.apache.curator</groupId>
20     <artifactId>curator-framework</artifactId>
21     <version>2.12.0</version>
22 </dependency>
23 <dependency>
24     <groupId>org.apache.curator</groupId>
25     <artifactId>curator-recipes</artifactId>
26     <version>2.12.0</version>
27 </dependency>
28 <dependency>
29     <groupId>org.apache.zookeeper</groupId>
```

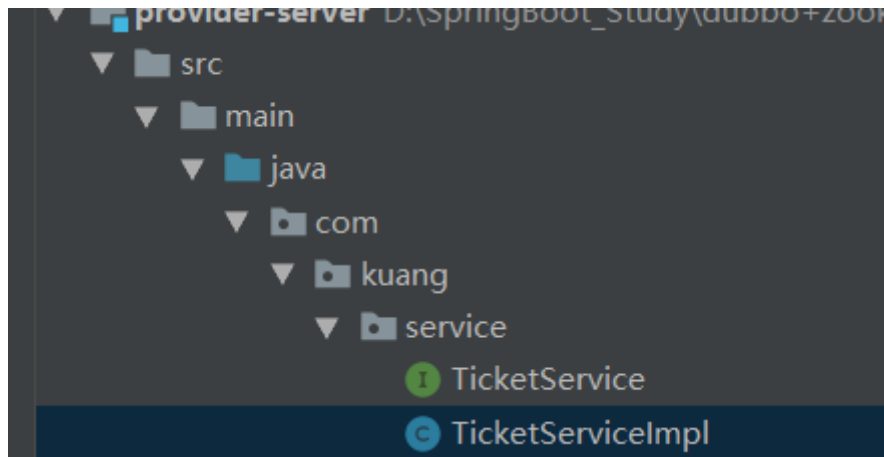
```

29     <artifactId>zookeeper</artifactId>
30     <version>3.4.14</version>
31     <!--排除这个slf4j-log4j12-->
32     <exclusions>
33         <exclusion>
34             <groupId>org.slf4j</groupId>
35             <artifactId>slf4j-log4j12</artifactId>
36         </exclusion>
37     </exclusions>
38 </dependency>

```

3.1 提供者

在provider-server项目下：



```

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

```

```

1 package com.kuang.service;
2
3 import org.apache.dubbo.config.annotation.Service;
4 import org.springframework.stereotype.Component;
5
6 //Zookeeper: 服务注册与发现
7
8 @Service //可以被扫描到，在项目一启动就自动注册到注册中心 import
9 org.apache.dubbo.config.annotation.Service;
10 @Component //使用了Dubbo，最好不要用@Service注解
11 public class TicketServiceImpl implements TicketService {
12     @Override
13     public String getTicket() {
14         return "Study Dubbo + ZooKeeper";
15     }
16 }

```

注意：@service不要导错包===》是要导Dubbo中的。

配置application.properties:

```

1 server.port=8001
2
3
4 # 服务应用名字
5 dubbo.application.name=provider-server
6 # 注册中心地址
7 dubbo.registry.address=zookeeper://127.0.0.1:2181
8 # 哪些服务要被注册（扫描包）
9 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。

```

1 package com.kuang.service;
2
3 import org.apache.dubbo.config.annotation.Reference;
4 import org.springframework.stereotype.Service;
5
6 @Service //放到容器中注册，导入的是Spring的包
7 public class UserService {
8     //想拿到provider-server提供的票
9     //要去注册中心拿

```



```

10  @Reference //引用,可以使用Pom坐标或者 定义路径相同的接口名
11  TicketService ticketService;
12
13  public void buyTicket() {
14      String ticket = ticketService.getTicket();
15      System.out.println("在注册中心拿到====>" + ticket);
16  }
17
18  }

```

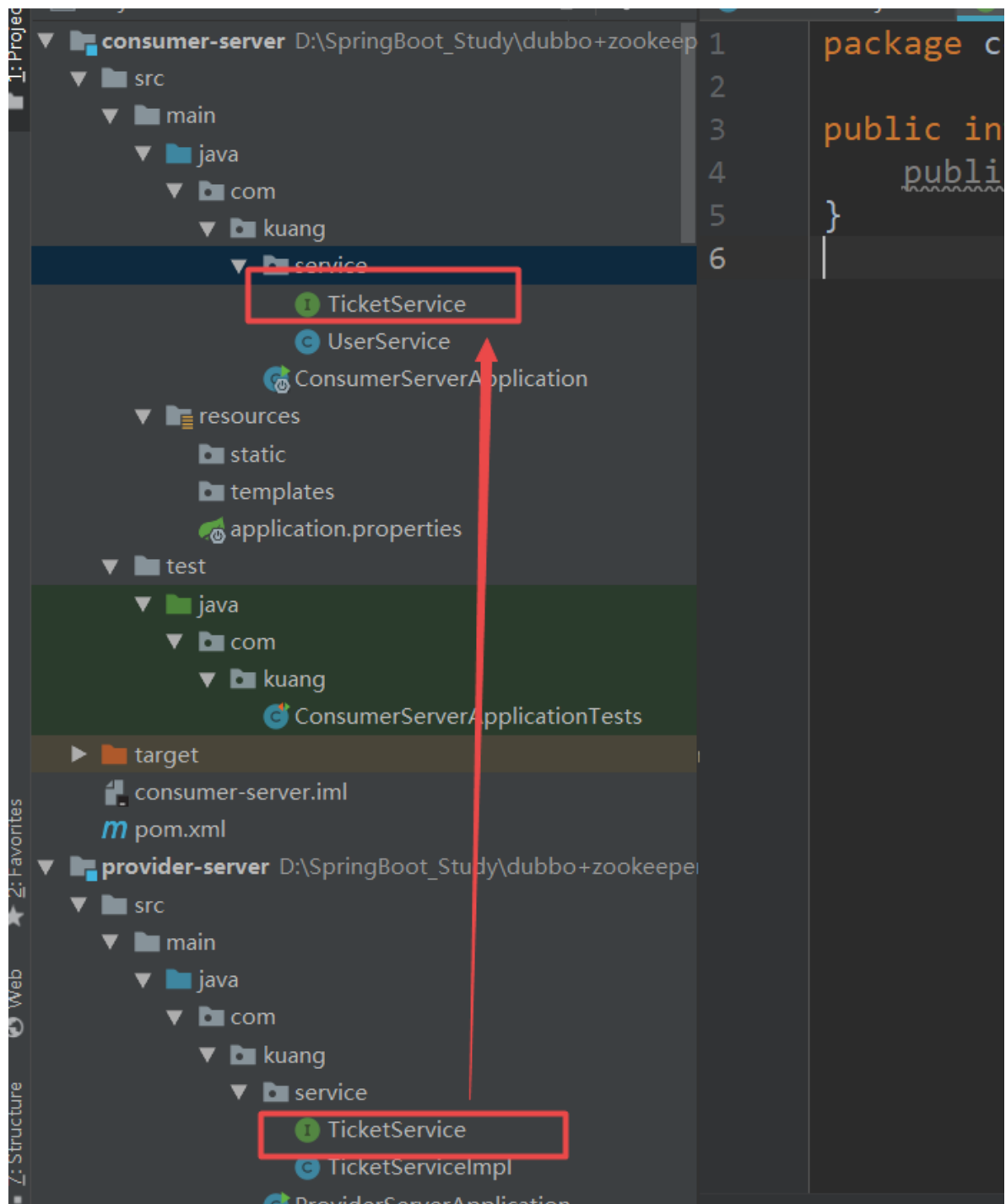
application.properties:

```

1  server.port=8002
2
3  # 消费者去哪里拿服务【去注册中心拿】
4  # 消费者拿服务时，必须暴露自己的名字
5  dubbo.application.name=consumer-server
6  # 注册中心的地址
7  dubbo.registry.address=zookeeper://127.0.0.1:2181
8
9  # 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:

```
1 @SpringBootTest
2 class ConsumerServerApplicationTests {
3     @Autowired
4     private UserService userService;
5
6     @Test
7     void contextLoads() {
8         userService.buyTicket();
9     }
10
11 }
```

```

2020-02-27 18:09:13.517 INFO 4560 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Ini
2020-02-27 18:09:14.182 INFO 4560 --- [main] c.kuang.ConsumerServerApplicationTests : Sta
seconds (JVM running for 9.495)
在注册中心拿到====>Study Dubbo + ZooKeeper
2020-02-27 18:09:15.094 INFO 4560 --- [extShutdownHook] .b.c.e.AwaitingNonWebApplicationListener : [D
shutdown...
2020-02-27 18:09:15.151 INFO 4560 --- [tor-Framework-0] o.a.c.f.imps.CuratorFrameworkImpl : bac
2020-02-27 18:09:15.208 INFO 4560 --- [extShutdownHook] org.apache.zookeeper.ZooKeeper : Ses
2020-02-27 18:09:15.208 INFO 4560 --- [ain-EventThread] org.apache.zookeeper.ClientCnxn : Eve
2020-02-27 18:09:15.222 INFO 4560 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shu
2020-02-27 18:09:15.223 INFO 4560 --- [extShutdownHook] f.a.ReferenceAnnotationBeanPostProcessor : org
destroying!
2020-02-27 18:09:15.224 INFO 4560 --- [extShutdownHook] f.a.ReferenceAnnotationBeanPostProcessor : cla
.annotation.ReferenceAnnotationBeanPostProcessor was destroying!

```

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