Task\_two （实用篇）搭建dubbo+zookeeper+SSM系统

1. 前言 （总结SSM系统，心得）

本篇内容是基于前一篇—构建SSM框架—之后的第二个任务，在SSM框架的基础上，集成构建dubbo与zookeeper。不得不说，此部分内容网上参考资料很多，但是真正适合我这种小白从零开始一步步入门的不多，所以，在完成任务的前提下，本人总结构建项目的全过程，一方面希望帮助需要的同学，另一方面也做一个阶段性的总结与回顾。

特别吐槽下：配置过程中不同版本的jar包与软件安装的版本问题，真的真的搞死人…花半天时间配置工程和环境，花一天时间调试…能力一般，水平有限。

1. 环境相关介绍（dubbo + zookeeper）
   1. 环境介绍

环境：jdk1.8 + Spring(4.3.10) + Mybatis(3.2.7) + oracle(11.2.0.3) + dubbo(2.5.3) + zookeeper(3.4.x; Windows环境) + apache-tomcat(8.0.52)

注：jdk的版本对于dubbo影响很大，不同版本的dubbo-admin所需要的jdk需要配置一致。网传：dubbo-2.8.4需用jdk版本为1.8，dubbo-2.5.3可以使用1.7版本的jdk。（参考链接：https://www.cnblogs.com/geekdc/p/7009947.html）

Zookeeper 配置环境用的3.4.6，安装的节点版本用的是3.4.13，由于zookeeper3.4.x版本之间影响不大，可以直接安装配置，特此说明。

* 1. 相关资源下载

项目工程已经上传到GitHub，由于后续任务的推进也会不定时更新。

整个项目工程涉及到的软件与安装包，可以在网上自行下载，若是找不到相关资源，可以通过以下链接下载：

oracle(11.2.0.3)：

dubbo(2.5.3)：

zookeeper(3.4.x)：

apache-tomcat(8.0.52)：

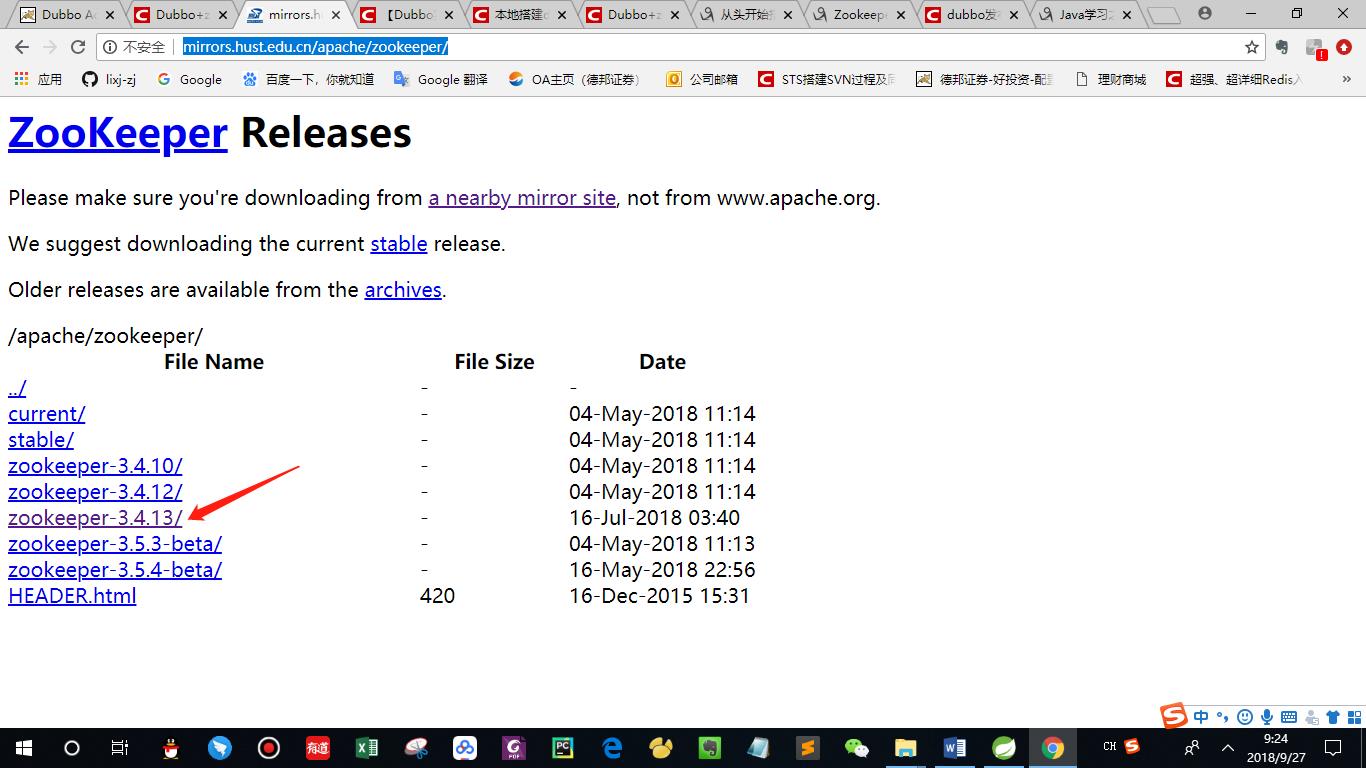
1. 搭建zookeeper节点：

　ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，是Hadoop和Hbase的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、名字服务、分布式同步、组服务等。

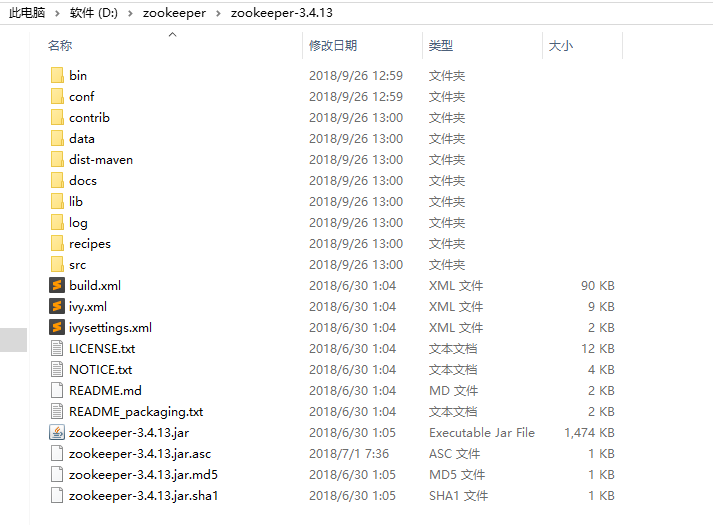
ZooKeeper的目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

　　ZooKeeper官网为：http://zookeeper.apache.org/

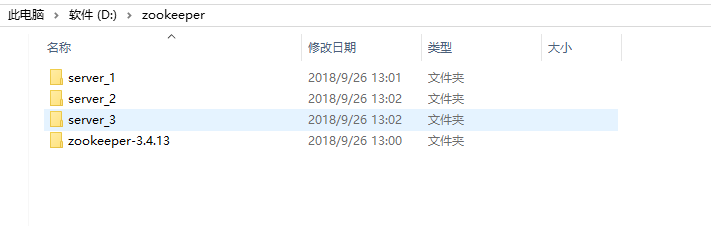
3.1 Zookeeper官网下载地址为：<http://mirrors.hust.edu.cn/apache/zookeeper/>



3.2 点击下载之后，获得zookeeper压缩包，在E盘新建zookeeper目录，解压至此。



3.3 将此zookeeper解压文件复制三份，模拟三个节点，如图：



3.3 Zookeeper 的配置文件在conf目录下，这个目录下有 zoo\_sample.cfg 和 log4j.propert ies，将zoo\_sample.cfg 改名为 zoo.cfg，Zookeeper 的节点在启动时会找这个文件作为默认配置文件。

打开 D:\zookeeper\server\_1\conf文件夹中的zoo.cfg文件，修改后的内容为：

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=D:\\zookeeper\\server\_1\\data

dataLogDir=D:\\zookeeper\\server\_1\\log

# the port at which the clients will connect

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=192.168.130.43:2887:3887

server.2=192.168.130.43:2888:3888

server.3=192.168.130.43:2889:3889

主要修改以下几点：

dataDir=D:\\zookeeper\\server\_1\\data

dataLogDir=D:\\zookeeper\\server\_1\\log

clientPort=2181

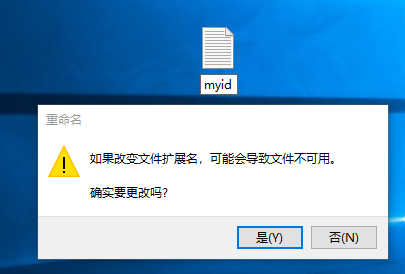
server.1=192.168.130.43:2887:3887

server.2=192.168.130.43:2888:3888

server.3=192.168.130.43:2889:3889

3.4 在D:\zookeeper\server\_1 路径下新建data与log文件夹对应zoo.cfg文件中的dataDir与dataLogDir，分别存放此节点的标识文件与日志文件。

3.5在D:\zookeeper\server\_1\data路径下，新建txt文本文件，写入数字1后保存关闭，重命名文件为myid。注意！此时文件没有后缀！在Windows10下显示如下：



C:\Users\tebon\AppData\Local\Temp\1538014855(1).png

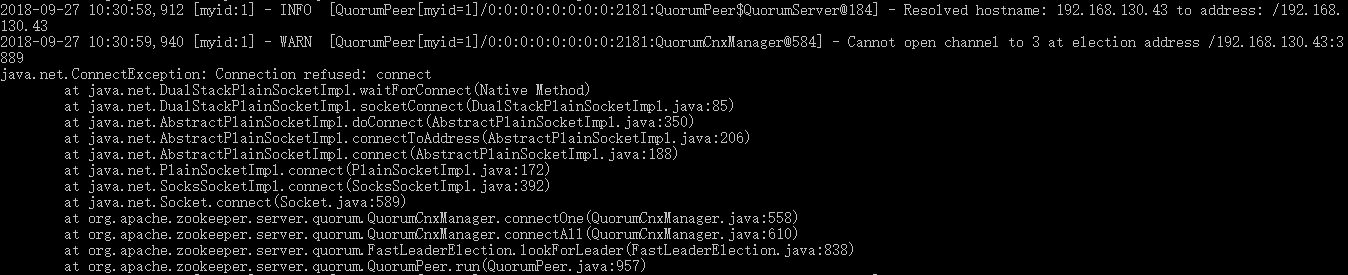
3.6 对应余下的两个节点server\_2与server\_3，同样操作即可。注意server\_2与server\_3 zoo.cfg中的dataDir与dataLogDir的对应本节点的路径、clientPort分别对应2182与2183、myid分别写入2与3。

3.7 至此，三个节点的zookeeper集群已经配置完毕，验证节点如下：

进入bin目录下启动服务，运行指令：zkServer

C:\Users\tebon\AppData\Local\Temp\1538015354(1).png

在一个节点启动后，可以发现cmd中出现如下问题：java.net.ConnectException: Connection refused: connect

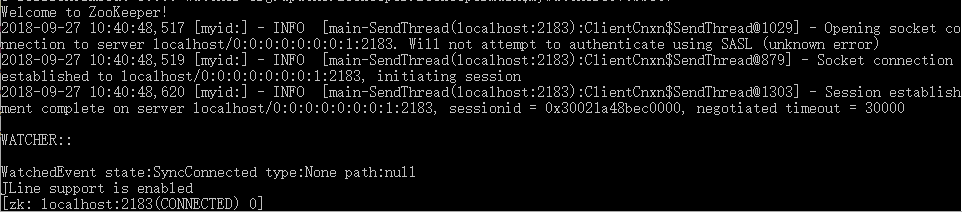


导致此问题的原因很多，包括防火墙、配置文件、端口等等（参考链接：https://blog.csdn.net/qq\_34841911/article/details/79165778）。但是当余下的两个节点依次开启时，此连接异常的问题将恢复正常。

同理，启动余下两个节点服务。

运行bin目录下启动客户端，运行指令：zkCli.cmd -server localhost:2181。注意！指定**每个节点分别对应的**端口号！

C:\Users\tebon\AppData\Local\Temp\1538016132(1).png



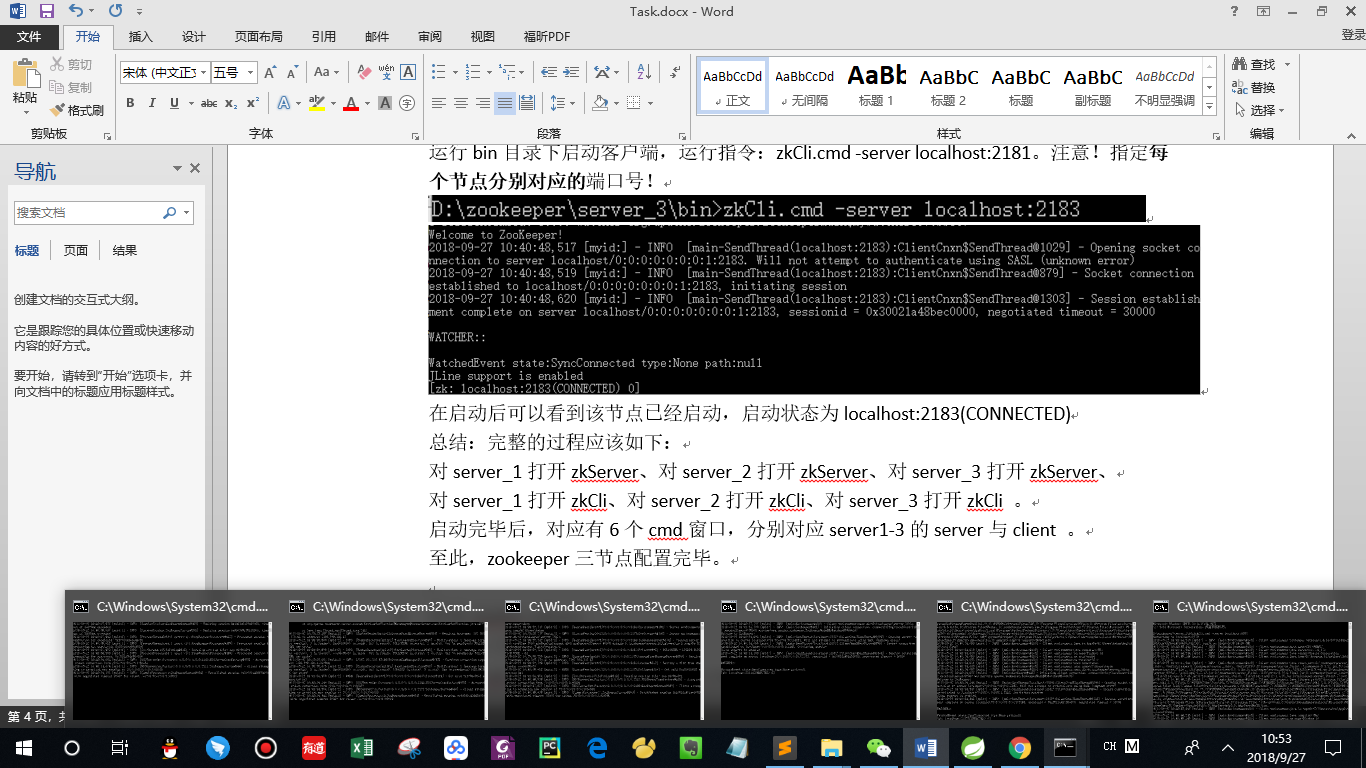
在启动后可以看到该节点已经启动，启动状态为localhost:2183(CONNECTED)

总结：完整的过程应该如下：

对server\_1打开zkServer、对server\_2打开zkServer、对server\_3打开zkServer、

对server\_1打开zkCli、对server\_2打开zkCli、对server\_3打开zkCli 。

启动完毕后，对应有6个cmd窗口，分别对应server1-3的server与client 。



至此，zookeeper三节点配置完毕。

另附zoo.cfg配置文件主要参数说明：

tickTime：zookeeper中使用的基本时间单位, 毫秒值这个时间是作为Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔，也就是每个 tickTime 时间就会发送一个心跳。

dataDir：数据目录. 可以是任意目录，默认情况下，Zookeeper 将写数据的日志文件也保存在这个目录里。

clientPort：这个端口就是客户端连接 Zookeeper 服务器的端口，Zookeeper 会监听这个端口，接受客户端的访问请求。

1. 搭建dubbo-admin

Dubbo是Alibaba开源的分布式服务框架，使得应用可通过高性能的 RPC 实现服务的输出和输入功能，可以和Spring框架无缝集成。它最大的特点是按照分层的方式来架构，使用这种方式可以使各个层之间解耦合（或者最大限度地松耦合）。从服务模型的角度来看，Dubbo采用的是一种非常简单的模型，要么是提供方提供服务，要么是消费方消费服务，所以基于这一点可以抽象出服务提供方（Provider）和服务消费方（Consumer）两个角色。关于注册中心、协议支持、服务监控等内容。

主要核心部件

**Remoting:** 网络通信框架，实现了 sync-over-async 和  request-response 消息机制.

**RPC:** 一个[远程过程调用](https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E8%BF%87%E7%A8%8B%E8%B0%83%E7%94%A8" \t "_blank)的抽象，支持[负载均衡](https://baike.baidu.com/item/%E8%B4%9F%E8%BD%BD%E5%9D%87%E8%A1%A1)、[容灾](https://baike.baidu.com/item/%E5%AE%B9%E7%81%BE)和[集群](https://baike.baidu.com/item/%E9%9B%86%E7%BE%A4)功能

**Registry:** 服务目录框架用于服务的注册和服务事件发布和订阅

Dubbo官网为：http://dubbo.io/

Dubbo架构图如图所示：



Provider

暴露服务方称之为“服务提供者”。

Consumer

调用远程服务方称之为“服务消费者”。

Registry

服务注册与发现的中心目录服务称之为“服务注册中心”。

Monitor

统计服务的调用次数和调用时间的日志服务称之为“服务监控中心”。

(1) 连通性：

注册中心负责服务地址的注册与查找，相当于目录服务，服务提供者和消费者只在启动时与注册中心交互，注册中心不转发请求，压力较小

监控中心负责统计各服务调用次数，调用时间等，统计先在内存汇总后每分钟一次发送到监控中心服务器，并以报表展示

服务提供者向注册中心注册其提供的服务，并汇报调用时间到监控中心，此时间不包含网络开销

服务消费者向注册中心获取服务提供者地址列表，并根据负载算法直接调用提供者，同时汇报调用时间到监控中心，此时间包含网络开销

注册中心，服务提供者，服务消费者三者之间均为长连接，监控中心除外

注册中心通过长连接感知服务提供者的存在，服务提供者宕机，注册中心将立即推送事件通知消费者

注册中心和监控中心全部宕机，不影响已运行的提供者和消费者，消费者在本地缓存了提供者列表

注册中心和监控中心都是可选的，服务消费者可以直连服务提供者

(2) 健壮性：

监控中心宕掉不影响使用，只是丢失部分采样数据

数据库宕掉后，注册中心仍能通过缓存提供服务列表查询，但不能注册新服务

注册中心对等集群，任意一台宕掉后，将自动切换到另一台

注册中心全部宕掉后，服务提供者和服务消费者仍能通过本地缓存通讯

服务提供者无状态，任意一台宕掉后，不影响使用

服务提供者全部宕掉后，服务消费者应用将无法使用，并无限次重连等待服务提供者恢复

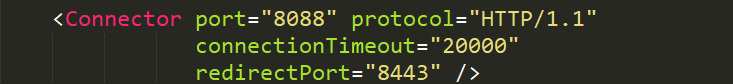
(3) 伸缩性：

注册中心为对等集群，可动态增加机器部署实例，所有客户端将自动发现新的注册中心

服务提供者无状态，可动态增加机器部署实例，注册中心将推送新的服务提供者信息给消费者

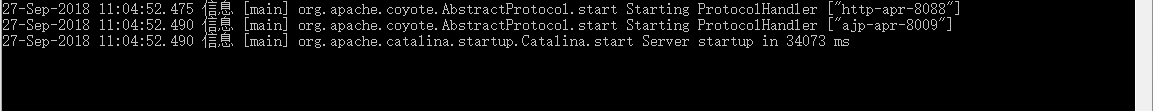
* 1. 由于jdk版本不同，dubbo-admin.war无法通用，网上大多war包基本都是基于jdb1.7的，本文使用的war包是jdk1.8版本，tomcat是apache-tomcat-8.0.52版本，具体下载：
  2. 下载好dubbo-admin.war后，按照常用的web部署方式部署。

先修改本地tomcat 端口，因为zookeeper会用到8080的端口，为了防止端口冲突，需要修改conf下的文件 server.xml，把Tomcat的端口改为8088。

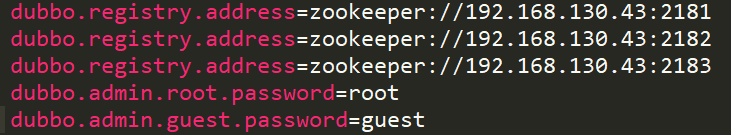


把war包放到apache-tomcat-8.0.52的webapps目录下，在bin目录中双击startup.bat启动tomcat即可。

启动成功后如下：



在启动成功后，还需要修改dubbo-admin的相关配置，进入apache-tomcat-8.0.52-dubbo-admin\webapps\dubbo\WEB-INF目录下，打开dubbo.properties文件，



这里的dubbo.registry.address后面对应的IP地址是我们的zookeeper对应的IP与端口地址，通常初始状态对应本机的IP地址。

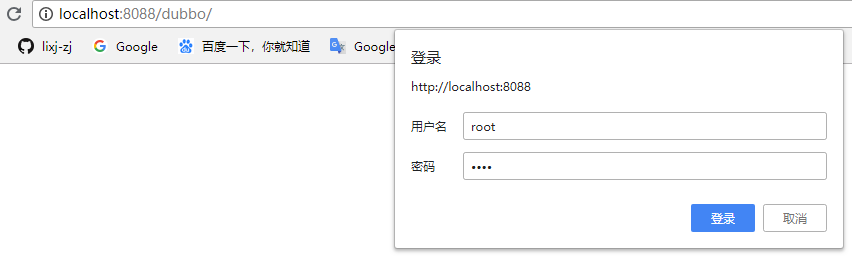
**dubbo.admin. root .password=root与dubbo.admin.guest .password=guest**

分别是默认的两个用户root与guest，默认密码与用户名相同。

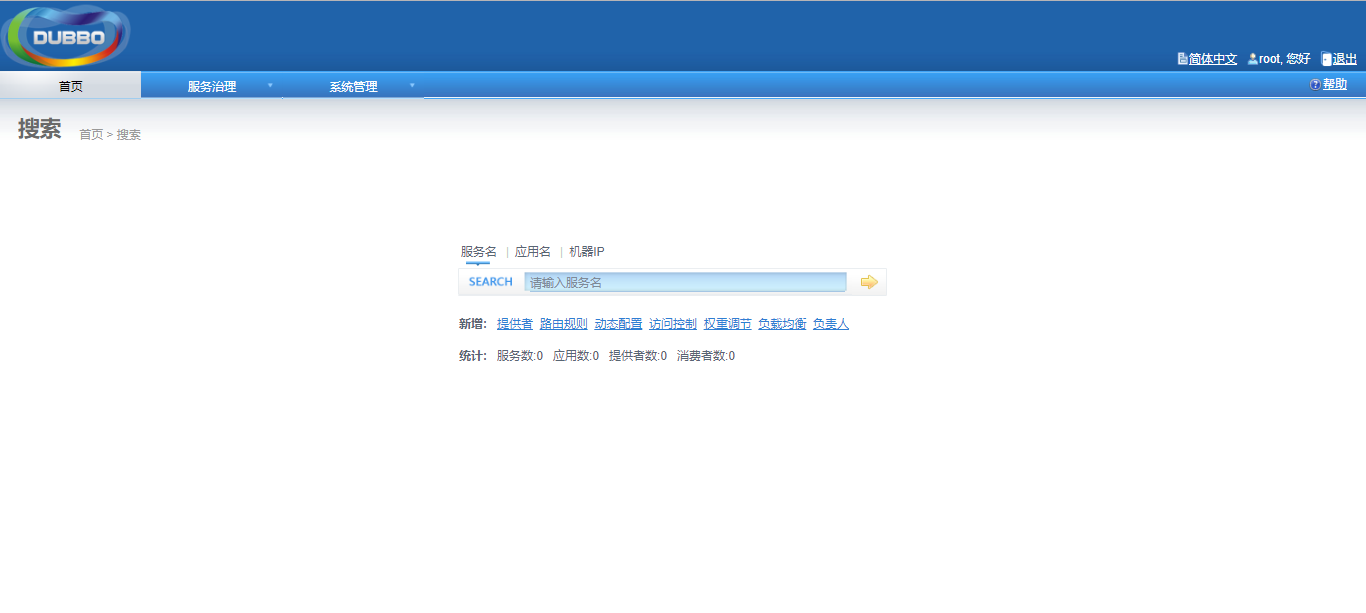
4.3 启动顺序：先启动zookeeper，再启动tomcat！

在tomcat启动完毕后，访问http: //ip地址:端口号/dubbo/ 或localhost:8088/dubbo/

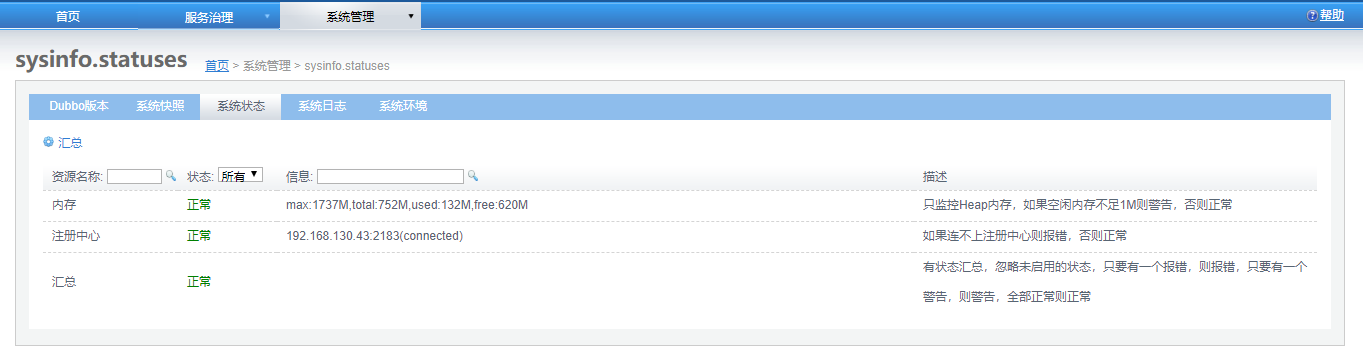
注意！此名称dubbo对应webapp下的工程名，名称要一致，如果一切正常，则会弹出登录界面，如下图所示。



输入用户名密码都是root后，进入管理页面如下图：



在系统管理->系统状态中可以查看系统当前的状态：



至此，zookeeper与dubbo的整个部署完成！接下来开始应用！

1. Maven工程项目

task\_one, dubbo-api, dubbo-provider, dubbo-consumer

工程分为4个结构

task\_one工程，主工程，主要导入公共jar包等

dubbo-api 工程，公共接口

dubbo-provider 工程，服务提供者

dubbo-consumer 工程，服务消费者

注：每个工程都是独立的，通过maven 引入父子关系）

5.1修改task\_one的maven 主工程

创建步骤参见上一篇文章。在此引入父工程时，主要新加相关jar包，包括zookeeper、dubbo依赖包，更改打包方式为pom，添加三个子项目的所属关系：

<packaging>pom</packaging>

<profiles>

<profile>

<modules>

<module>dubbo-api</module>

<module>dubbo-provider</module>

<module>dubbo-consumer</module>

</modules>

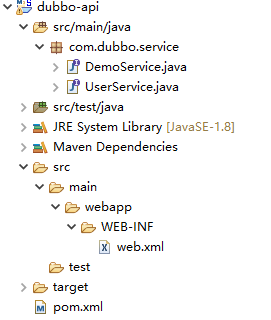
</profile>

</profiles>

5.2创建dubbo-api 的maven 接口项目

该项目主要定义接口，供dubbo-provider与dubbo-consumer项目调用，并配置好pom.xml。

该项目结构如下：



Pom.xml文件中依赖父项目：

<parent>

<artifactId>task\_one</artifactId>

<groupId>com.lixj.work</groupId>

<version>0.0.1-SNAPSHOT</version>

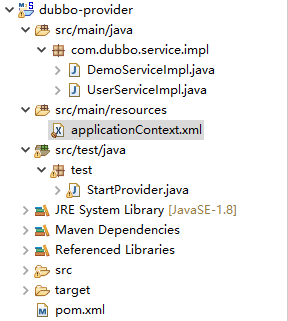
</parent>

dubbo-api代码连接：

5.2创建dubbo-provider 的服务提供者项目

该项目主要实现公共接口，但是对消费者隐藏实现过程。

该项目结构如下：



Pom.xml文件中调用公共接口：

<dependency>

<groupId>com.lixj.work</groupId>

<artifactId>dubbo-api</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

注意！<groupId>是父项目task\_one所对应的groupid。

ApplicationContext.xml声明dubbo的配置服务如下。主要配置zookeeper的注册中心，声明需要暴露的接口（即dubbo-api的公共接口）与具体实现的bean。

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://code.alibabatech.com/schema/dubbo

http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

<!-- 定义了提供方应用信息，用于计算依赖关系。在dubbo-admin 会显示这个名字，方便识别 -->

<dubbo:application name="admin-provider" owner="dubbo" organization="dubbox"/>

<!-- 使用zookeeper 注册中心暴露服务，注意要先开启 zookeeper -->

<dubbo:registry protocol="zookeeper" address="localhost:2181,localhost:2182,localhost:2183"/>

<!-- 用dubbo协议在20880端口暴露服务 -->

<dubbo:protocol name="dubbo" port="20880"/>

<!-- 声明需要暴露的服务接口 -->

<dubbo:service interface="com.dubbo.service.DemoService" ref="demoService" protocol="dubbo"/>

<dubbo:service interface="com.dubbo.service.UserService" ref="userService" protocol="dubbo"/>

<!-- 具体实现接口的bean -->

<bean id="demoService" class="com.dubbo.service.impl.DemoServiceImpl"/>

<bean id="userService" class="com.dubbo.service.impl.UserServiceImpl"/>

</beans>

测试类如下。主要加载applicationContext.xml配置文件开启服务。

package test;

import java.io.IOException;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class StartProvider {

public static void main(String[] args) throws IOException {

ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

System.out.println(context.getDisplayName() + ": here");

context.start();

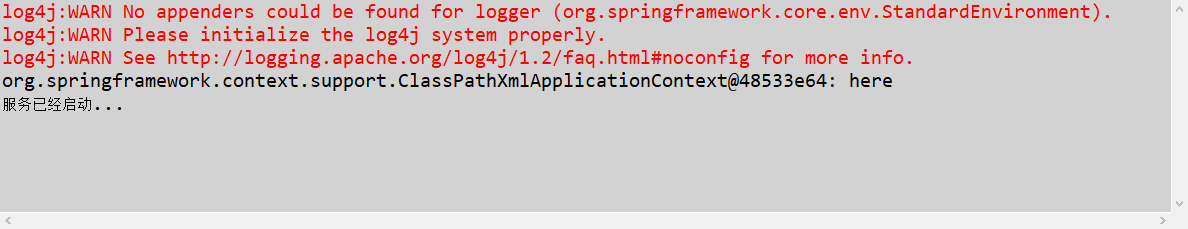
System.out.println("服务已经启动...");

System.in.read();

}

}

将测试类 run as -> java application后，结果如下（前提：开启zookeeper！！！）。若发生错误（如 Error creating bean with name….等），参考错误集锦。

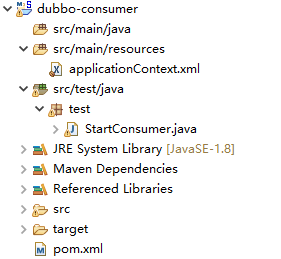


dubbo-provider代码连接：

5.3创建dubbo-consumer 的服务消费者项目

该项目主要调用服务，即调用dubbo-provider提供的服务。

该项目结构如下：



Pom.xml文件中调用公共接口：

<dependency>

<groupId>com.lixj.work</groupId>

<artifactId>dubbo-api</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

注意！<groupId>是父项目task\_one所对应的groupid。

ApplicationContext.xml声明dubbo的配置服务如下。主要用于生成远程服务代理，可以像使用本地bean一样使用Service，并且使用dubbo 协议调用定义好的 api 接口。

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://code.alibabatech.com/schema/dubbo

http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

<!-- 定义了消费方应用信息，用于计算依赖关系。在dubbo-admin 会显示这个名字，方便识别 -->

<dubbo:application name="dubbo-consumer" owner="dubbo" organization="dubbox"/>

<!-- 向zookeeper 订阅provider 的地址，由zookeeper 定时推送 -->

<dubbo:registry protocol="zookeeper" address="localhost:2181,localhost:2182,localhost:2183"/>

<!-- 生成远程服务代理，可以像使用本地bean一样使用Service -->

<!-- 使用dubbo 协议调用定义好的 api 接口 -->

<dubbo:reference id="userService" interface="com.dubbo.service.UserService"/>

<dubbo:reference id="demoService" interface="com.dubbo.service.DemoService"/>

</beans>

测试类如下。主要加载applicationContext.xml配置文件调用服务。

package test;

import java.io.IOException;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.dubbo.service.DemoService;

public class StartConsumer {

public static void main(String[] args) throws IOException {

ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

context.start();

System.out.println("消费已经启动。。。");

DemoService demoService = (DemoService)context.getBean("demoService");

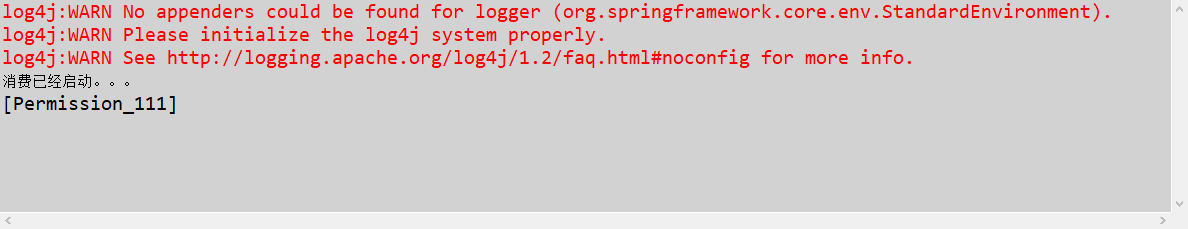
System.out.println(demoService.getPermissions(111L));

System.in.read();

}

}

将测试类 run as -> java application后，结果如下（前提：开启zookeeper与dubbo provider！！！）。若发生错误，参加错误集锦。



dubbo-consumer代码链接：

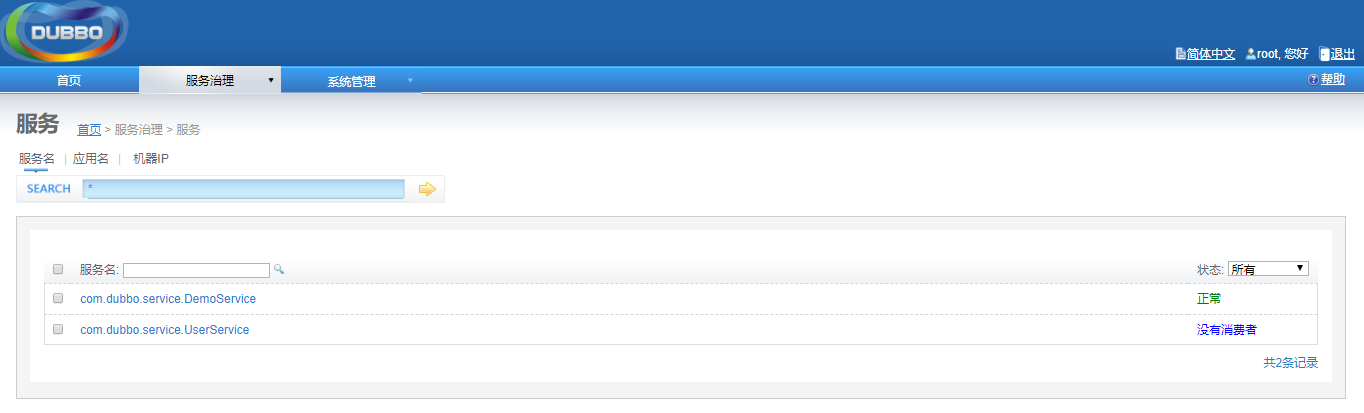
至此，基于zookeeper与dubbo的maven项目搭建完成！

1. 结果展示

经过安装配置zookeeper、dubbo、maven项目后，可以通过dubbo-admin访问当前提供者数与消费者数，结果如下：

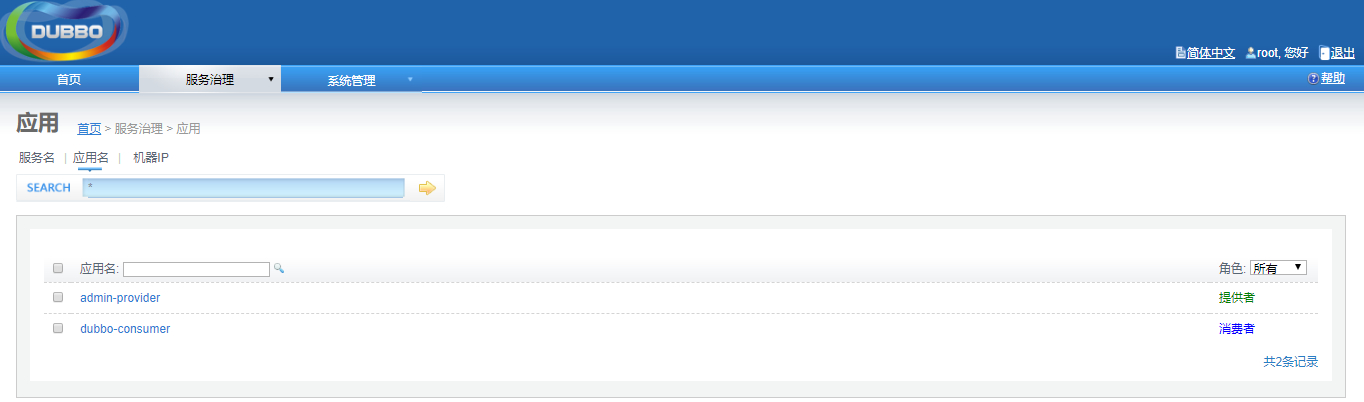


点击服务治理 –> 服务，可以看到当前提供的服务名：



可以看到，由于dubbo-consumer中只对demoService进行调用，所以com.dubbo.service.DemoService状态为正常，而com.dubbo.service.UserService显示没有消费者。

点击应用名后，可以看到当前提供者与消费者的信息：



其中，admin-provider是dubbo-provider的applicationContext.xml配置文件中的应用名；dubbo-consumer是dubbo-consumer的applicationContext.xml配置文件中的应用名。

1. 总结

本人菜鸟一个，水平一般，能力有限。从接触zookeeper与dubbo到本文完成，耗时约三天，算是很慢上手的了，在搭建的过程中采坑不少，仅错误集锦就高达十几处之多。一方面给有需要的同学参考，另一方面是阶段性的记录与总结。在项目搭建的过程中，印象最深的在于无法启动服务（即无法加载applicationContext.xml文件），dubbo-provider死活注册不了，纠结了好久，期间更改过配置文件applicationContext中的dubbo:service标签（改为dubbo:reference），虽然可以启动服务，但是却无法注册（最终解决方法见错误集锦15，坑爹的javassist！！！）。究其原委，还是对于dubbo不了解，对未知的事物不了解！代码的复制粘贴虽然在一定程度上可以提高速度，有时却无法提高效率。所以，要慎重的考虑每个复制的代码，慎重的查看每个出现的错误或异常。

与君同勉！

1. 错误集锦

1. pom.xml出现web.xml is missing and <failOnMissingWebXml> is set to true解决方案

https://blog.csdn.net/sinat\_22911279/article/details/77454139

2. Updating Maven Project”. java.lang.NullPointerException

https://stackoverflow.com/questions/19522897/an-internal-error-occurred-during-updating-maven-project-java-lang-nullpoint

3. ZooKeeper异常之无法找到myid文件

https://blog.csdn.net/cflys/article/details/76598413

自己总结：新建txt文件，名称为myid，写入数字 1，然后删除文件后缀.txt，即文件名为myid，文件类型为 文件 ！

4. Address already in use: bind

端口被占用，同一个机器上，conf文件中的clientPort 从2181递增 2182 2183.。。。

5. Will not attempt to authenticate using SASL (unknown error)

https://blog.csdn.net/tengdazhang770960436/article/details/53895497

1. conf文件中修改本地IP： https://blog.csdn.net/tengdazhang770960436/article/details/53895497

server.1=192.168.130.43:2888:3888

2. 或者 myid文件中的值与conf文件中定义的不一致。

6. 查看端口被调用：

https://jingyan.baidu.com/article/d8072ac4855b38ec95cefdbf.html

7. dubbo-admin 无响应或者tomcat访问无响应

在启动dubbo-admin之前，先启动本地的所有zookeeper service服务，即开启三个节点的zkServer，由于在dubbo.properties中，设置的dubbo.registry.address=zookeeper://127.0.0.1:2181

所以还需要启动service 2181 的client，即启动zkCli.cmd -server localhost:2181

重启tomcat，即可访问dubbo-admin

原因：zookeeper没有启动成功

参考：https://blog.csdn.net/u010297957/article/details/51702076#commentBox

8. Address already in use: bind 程序端口被占用

https://blog.csdn.net/jikefzz1095377498/article/details/79628319

9. Failed to execute goal on project dubbo-provider:

https://blog.csdn.net/jiangyu1013/article/details/79314027

10. jar:1.9.0-SNAPSHOT is missing, no dependency information available

maven clean，生成jar包或war包等，在子项目中调用

https://blog.csdn.net/jiangyu1013/article/details/79314027

11. 错误: 找不到或无法加载主类

https://www.cnblogs.com/wushuai2014/p/7468954.html

12. pom child module does not exist

https://stackoverflow.com/questions/26021141/maven-child-module-does-not-exist

https://blog.csdn.net/alizhuana/article/details/79419948

13. Tomcat 部署项目的三种方法

https://www.cnblogs.com/ysocean/p/6893446.html#\_label2

14. Failed to initialize end point associated with ProtocolHandler ["http-apr-8080"]

<https://www.cnblogs.com/onlycxue/p/3573154.html>

15. 无法产生bean实例

<https://blog.csdn.net/wwrzyy/article/details/79603421>

1. 参考链接

<https://blog.csdn.net/weijifeng_/article/details/79775738>

<http://www.cnblogs.com/huyong/p/6678475.html>

<https://blog.csdn.net/hua1586981/article/details/79195111>

<https://blog.csdn.net/hejingyuan6/article/details/47403299>