Dubbo-demo

## 创建 dubbo-provider maven web项目

## pom 依赖

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.lyj.dubbo</groupId>

<artifactId>dubbo-provider</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<!-- dubbo采用spring配置方式，所以需要导入spring容器依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.1.3.RELEASE</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.6.4</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>2.5.3</version>

<exclusions>

<exclusion>

<!-- 排除传递spring依赖 -->

<artifactId>spring</artifactId>

<groupId>org.springframework</groupId>

</exclusion>

</exclusions>

</dependency>

<!-- zookeeper -->

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

<version>3.3.3</version>

</dependency>

<dependency>

<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

<version>0.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<port>8187</port>

<path>/</path>

</configuration>

</plugin>

</plugins>

</build>

</project>

## Bean

**package** com.lyj.dubbo.bean;

// 使用dubbo要求传输的对象必须实现序列化接口

**public** **class** User **implements** java.io.Serializable {

/\*\*

\* serialVersionUID

\*/

**private** **static** **final** **long** ***serialVersionUID*** = -5330406141916172644L;

**private** Long id;

**private** String username;

**private** String password;

**private** Integer age;

**private** Integer gender;

**public** Integer getGender() {

**return** gender;

}

**public** **void** setGender(Integer gender) {

**this**.gender = gender;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** Integer getAge() {

**return** age;

}

**public** **void** setAge(Integer age) {

**this**.age = age;

}

}

## 创建服务接口

**public** **interface** UserService {

/\*\*

\* 查询所有用户

\* **@return**

\*/

**public** List<User> queryAll();

}

## 创建实现类

package com.lyj.dubbo.service.impl;

import java.util.ArrayList;

import java.util.List;

import com.lyj.dubbo.bean.User;

import com.lyj.dubbo.service.UserService;

public class UserServiceImpl implements UserService {

/\*\*

\* 模拟数据库查询

\*/

public List<User> queryAll() {

List<User> list = new ArrayList<User>();

for (int i = 0; i< 10; i++) {

User user = new User();

user.setId(Long.valueOf(i + 1));

user.setUsername("uname\_" + i);

user.setPassword("000000");

user.setAge(10 + i);

user.setGender(i%2);

list.add(user);

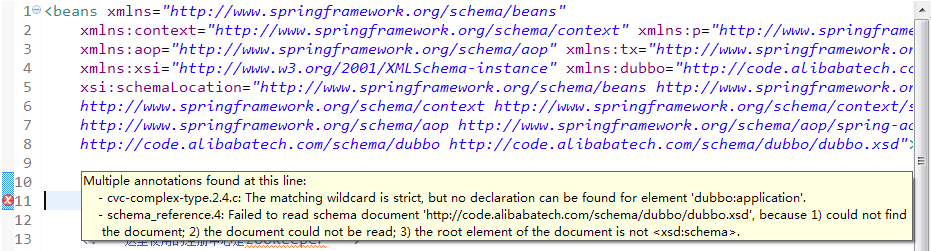
}

return list;

}

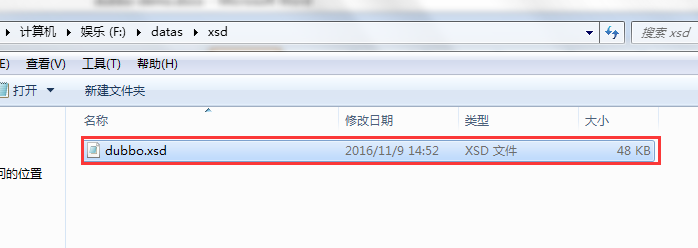
}

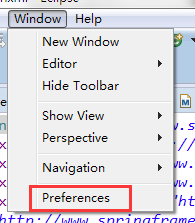
## 编写 Dubbo 配置文件

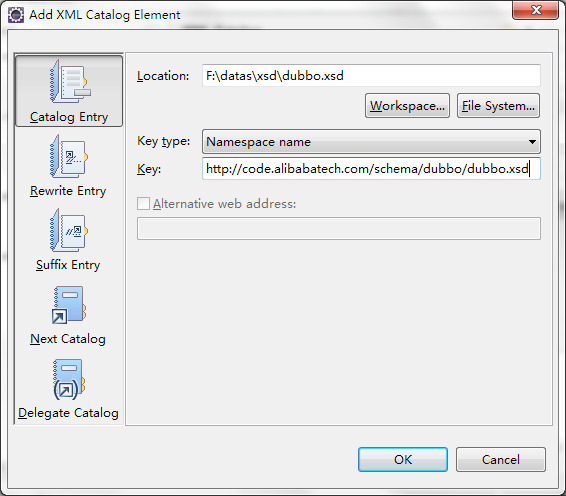


引入约束文件 xsd

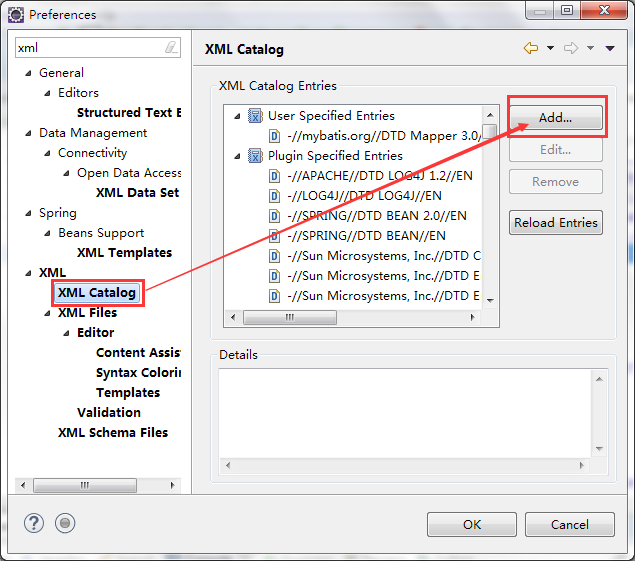
<http://code.alibabatech.com/schema/dubbo/dubbo.xsd>

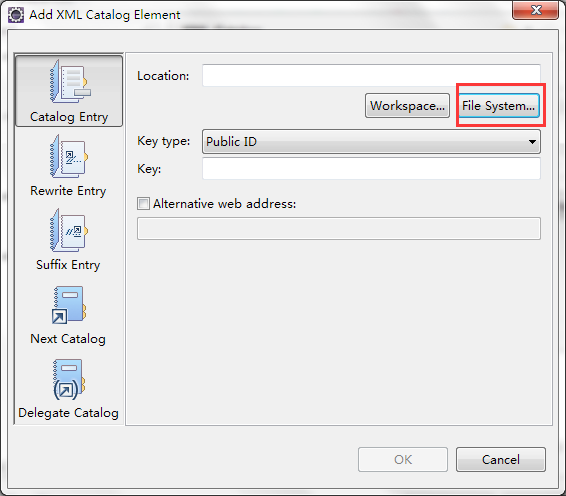






重新启动当前 Editor





选择刚刚的 dubbo.xsd 文件

## 配置 web.xml

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

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>dubbo-provider</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:dubbo/dubbo-\*.xml</param-value>

</context-param>

<!--Spring的ApplicationContext 载入 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

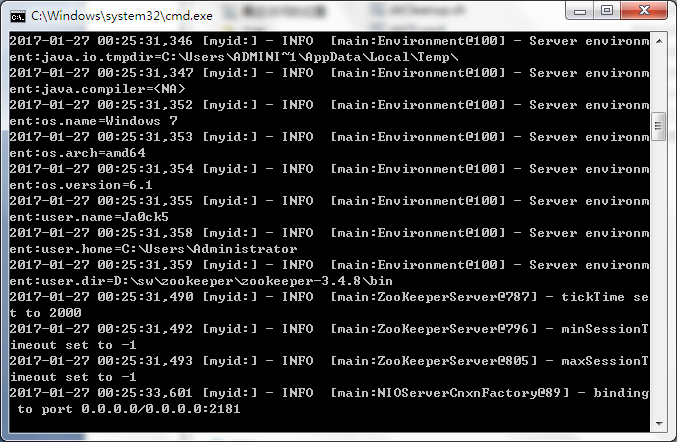
<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

## 启动 zookeeper



## 启动项目

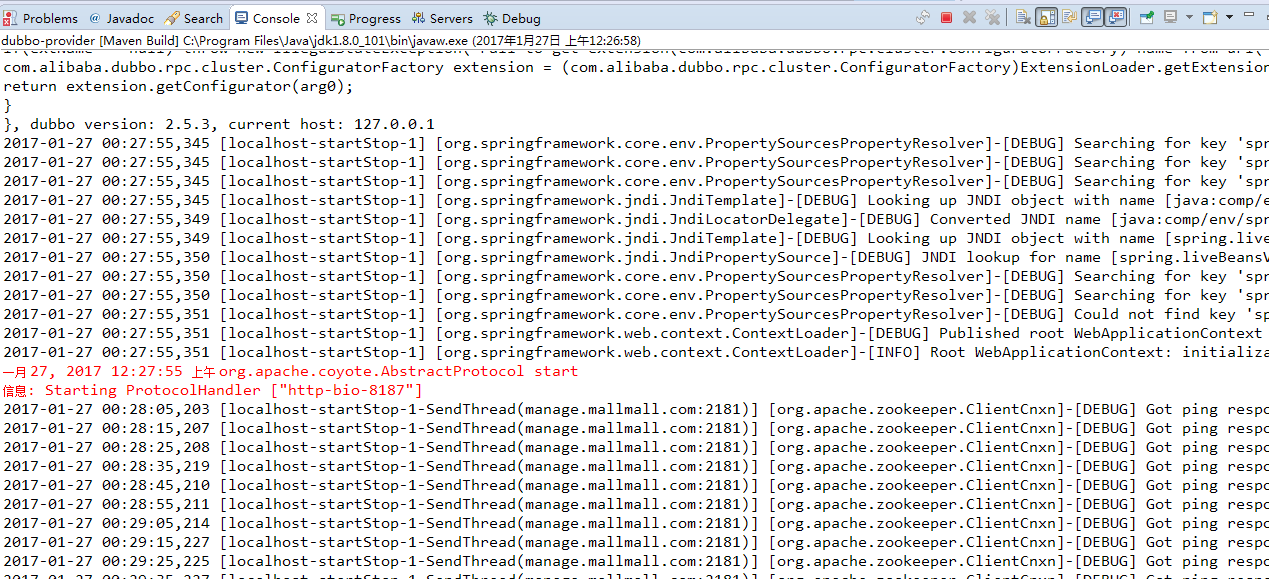
2017-01-27 00:27:53,976 [localhost-startStop-1] [com.alibaba.dubbo.config.AbstractConfig]-[INFO] [DUBBO] Export dubbo service com.lyj.dubbo.service.UserService to local registry, dubbo version: 2.5.3, current host: 127.0.0.1

2017-01-27 00:27:53,977 [localhost-startStop-1] [com.alibaba.dubbo.config.AbstractConfig]-[INFO] [DUBBO] Export dubbo service com.lyj.dubbo.service.UserService to url dubbo://192.168.64.1:20880/com.lyj.dubbo.service.UserService?anyhost=true&application=dubbo-provider-server&dubbo=2.5.3&interface=com.lyj.dubbo.service.UserService&methods=queryAll&pid=4708&side=provider&timestamp=1485448073650, dubbo version: 2.5.3, current host: 127.0.0.1

2017-01-27 00:27:53,977 [localhost-startStop-1] [com.alibaba.dubbo.config.AbstractConfig]-[INFO] [DUBBO] Register dubbo service com.lyj.dubbo.service.UserService url dubbo://192.168.64.1:20880/com.lyj.dubbo.service.UserService?anyhost=true&application=dubbo-provider-server&dubbo=2.5.3&interface=com.lyj.dubbo.service.UserService&methods=queryAll&pid=4708&side=provider&timestamp=1485448073650 to registry registry://127.0.0.1:2181/com.alibaba.dubbo.registry.RegistryService?application=dubbo-provider-server&client=zkclient&dubbo=2.5.3&pid=4708&registry=zookeeper&timestamp=1485448073622, dubbo version: 2.5.3, current host: 127.0.0.1

2017-01-27 00:27:54,011 [localhost-startStop-1] [com.alibaba.dubbo.common.extension.ExtensionLoader]-[DEBUG] [DUBBO] package com.alibaba.dubbo.registry;

## 启动成功



## 消费者

创建消费者 dubbo-consumer

pom

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.lyj.dubbo</groupId>

<artifactId>dubbo-consumer</artifactId>

<version>1.0.0-SNAPSHOT</version>

<dependencies>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.10</version>

<scope>test</scope>

</dependency>

<!-- dubbo采用spring配置方式，所以需要导入spring容器依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.1.3.RELEASE</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.6.4</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>2.5.3</version>

<exclusions>

<exclusion>

<!-- 排除传递spring依赖 -->

<artifactId>spring</artifactId>

<groupId>org.springframework</groupId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

<version>3.3.3</version>

</dependency>

<dependency>

<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

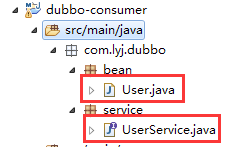
<version>0.1</version>

</dependency>

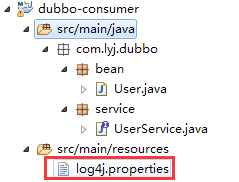
</dependencies>

</project>

## Consumer (Copy)使用Provider 的User 对象和 UserService 接口



引入日志



## 编写dubbo配置文件 dubbo-consumer.xml

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

xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*

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-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

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

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name=*"dubbo-a-consumer"*/>

<!-- 这里使用的注册中心是zookeeper -->

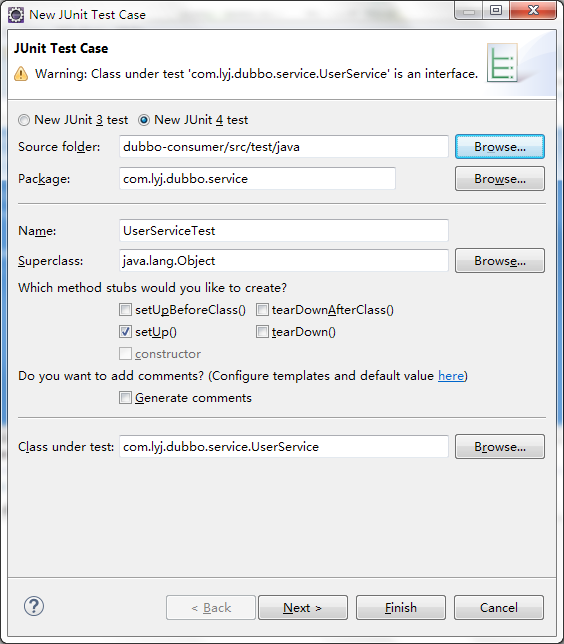
<dubbo:registry address=*"zookeeper://127.0.0.1:2181"* client=*"zkclient"*/>

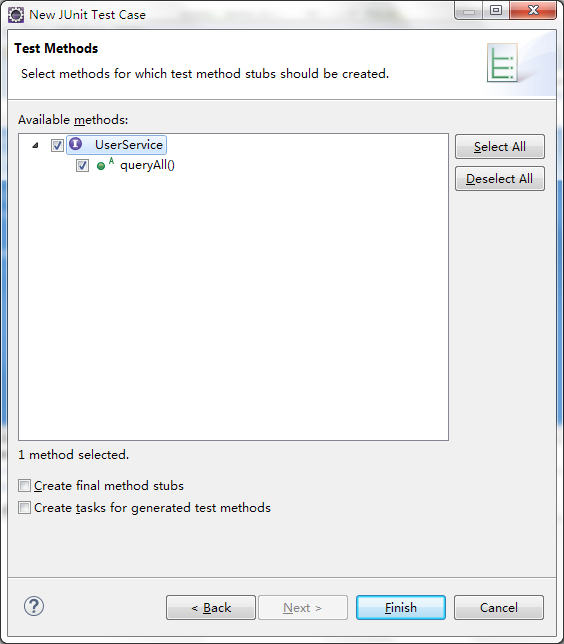
<!-- 从注册中心中查找服务 -->

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

</beans>

## 测试





测试代码：

**package** com.lyj.dubbo.service;

**import** java.util.List;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.lyj.dubbo.bean.User;

**public** **class** UserServiceTest {

**private** UserService userService;

@Before

**public** **void** setUp() **throws** Exception {

ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("classpath:dubbo/dubbo-\*.xml");

**this**.userService = applicationContext.getBean(UserService.**class**);

}

@Test

**public** **void** testQueryAll() {

List<User> list = **this**.userService.queryAll();

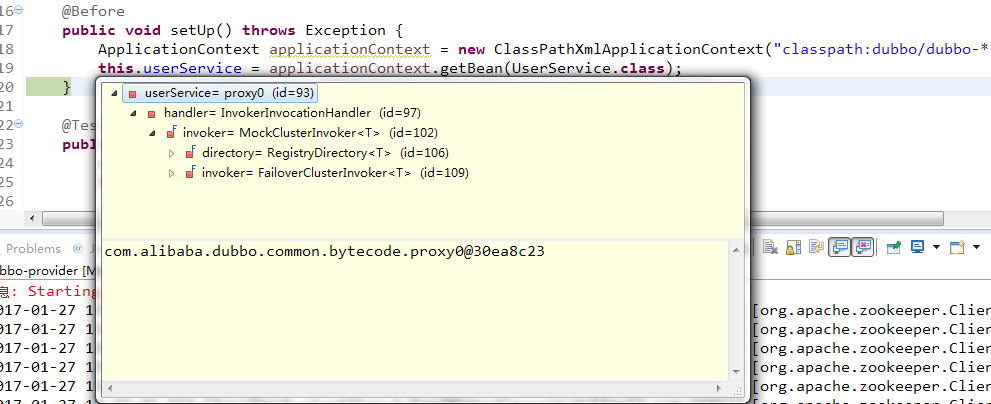
**for** (User user : list) {

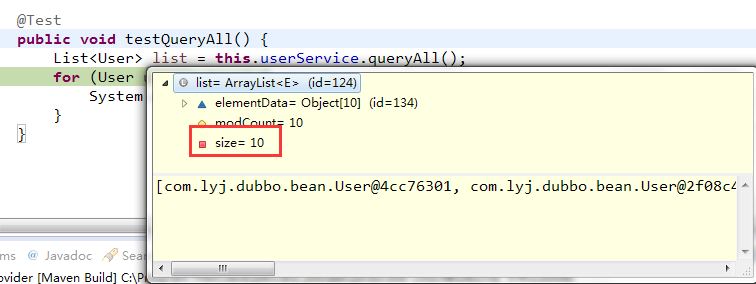
System.***out***.println(user);

}

}

}



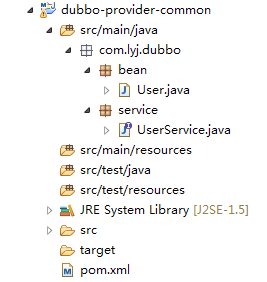


远程调用成功。

## 解决代码重复问题

将重复代码抽取成一个maven 项目，打包成 jar 在使用到的项目中引入依赖即可。

新建项目 dubbo-provider-common



在 dubbo-provider dubbo-consumer 中引入依赖

<dependency>

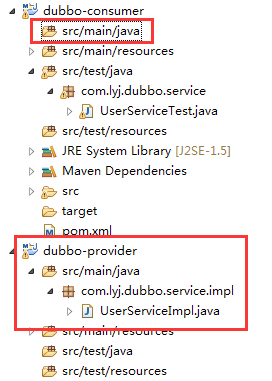
<groupId>com.lyj.dubbo</groupId>

<artifactId>dubbo-provider-common</artifactId>

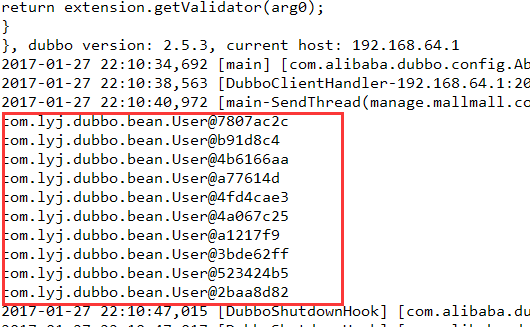
<version>1.0.0-SNAPSHOT</version>

</dependency>

删除重复代码

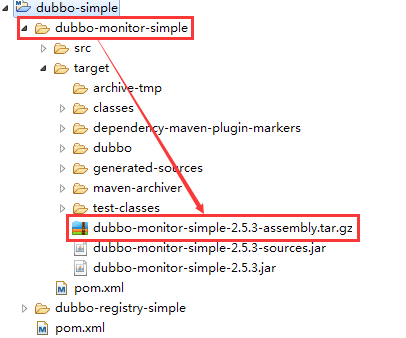


测试：



获得测试的User 对象。

## 配置监控中心

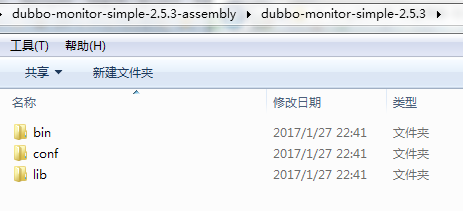


将构建好的 dubbo 项目中的 dubbo-simple 下的 dubbo-monitor-simple 下 target 下编译好的文件

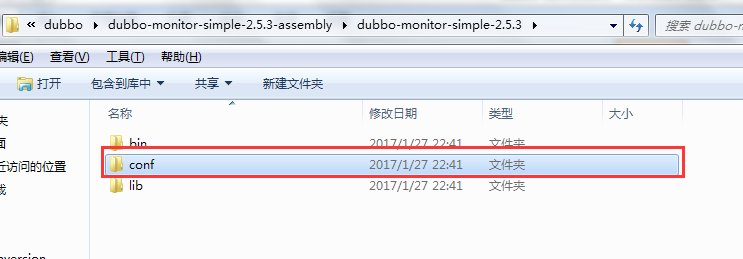
dubbo-monitor-simple-2.5.3-assembly.tar.gz

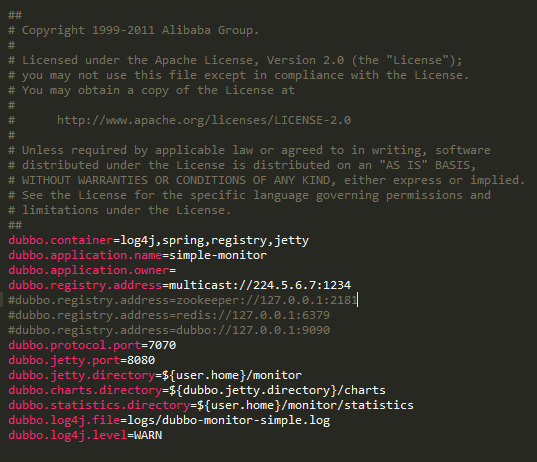
获取到

解压得到目录

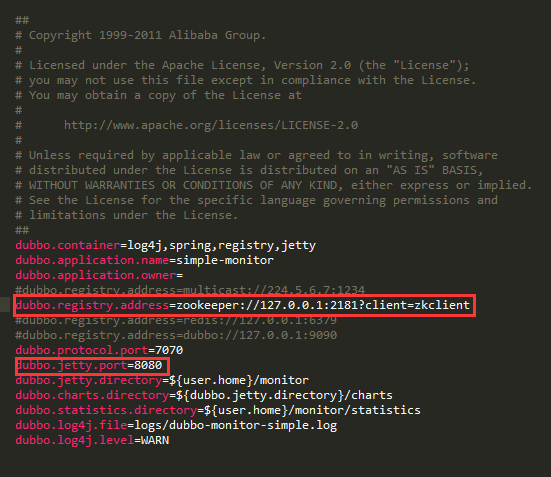


Conf 目录下修改配置文件





修改为



##

# Copyright 1999-2011 Alibaba Group.

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

##

dubbo.container=log4j,spring,registry,jetty

dubbo.application.name=simple-monitor

dubbo.application.owner=

#dubbo.registry.address=multicast://224.5.6.7:1234

dubbo.registry.address=zookeeper://127.0.0.1:2181?client=zkclient

#dubbo.registry.address=redis://127.0.0.1:6379

#dubbo.registry.address=dubbo://127.0.0.1:9090

dubbo.protocol.port=7070

dubbo.jetty.port=8080

dubbo.jetty.directory=${user.home}/monitor

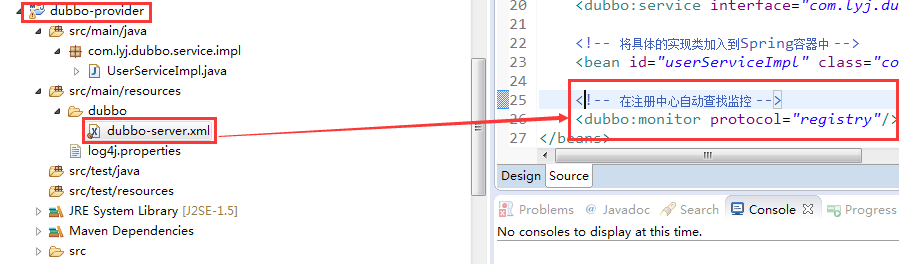
dubbo.charts.directory=${dubbo.jetty.directory}/charts

dubbo.statistics.directory=${user.home}/monitor/statistics

dubbo.log4j.file=logs/dubbo-monitor-simple.log

dubbo.log4j.level=WARN

## 在 dubbo-provider 中配置 在注册中心自动查找监控



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

xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*

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-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

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

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name=*"dubbo-provider-server"*/>

<!-- 这里使用的注册中心是zookeeper -->

<dubbo:registry address=*"zookeeper://127.0.0.1:2181"* client=*"zkclient"*/>

<!-- 用dubbo协议在20880端口暴露服务 -->

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

<!-- 将该接口暴露到dubbo中 -->

<dubbo:service interface=*"com.lyj.dubbo.service.UserService"* ref=*"userServiceImpl"*/>

<!-- 将具体的实现类加入到Spring容器中 -->

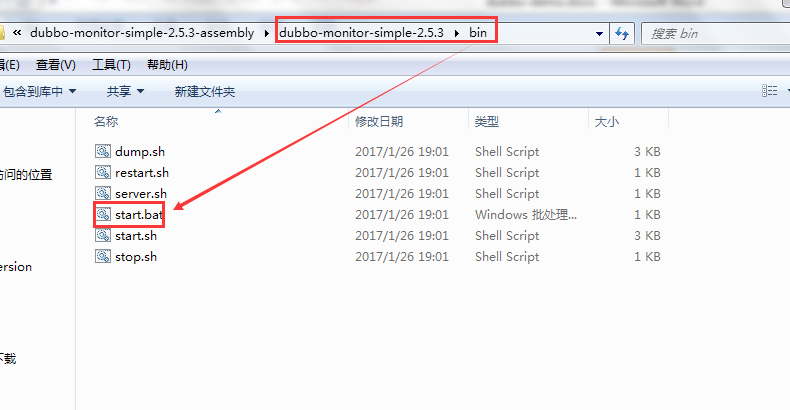
<bean id=*"userServiceImpl"* class=*"com.lyj.dubbo.service.impl.UserServiceImpl"*/>

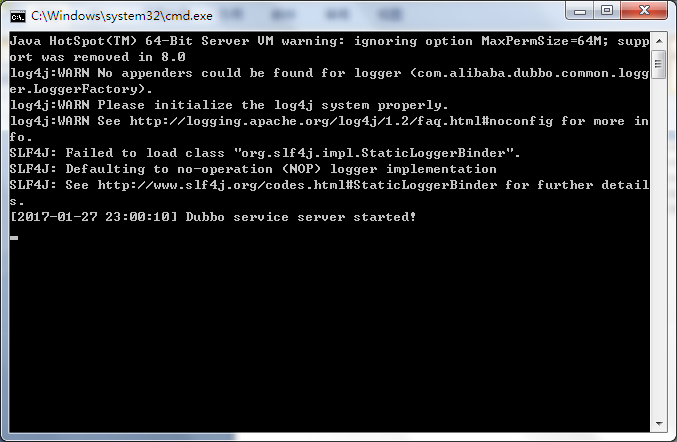
<!-- 在注册中心自动查找监控 -->

<dubbo:monitor protocol=*"registry"*/>

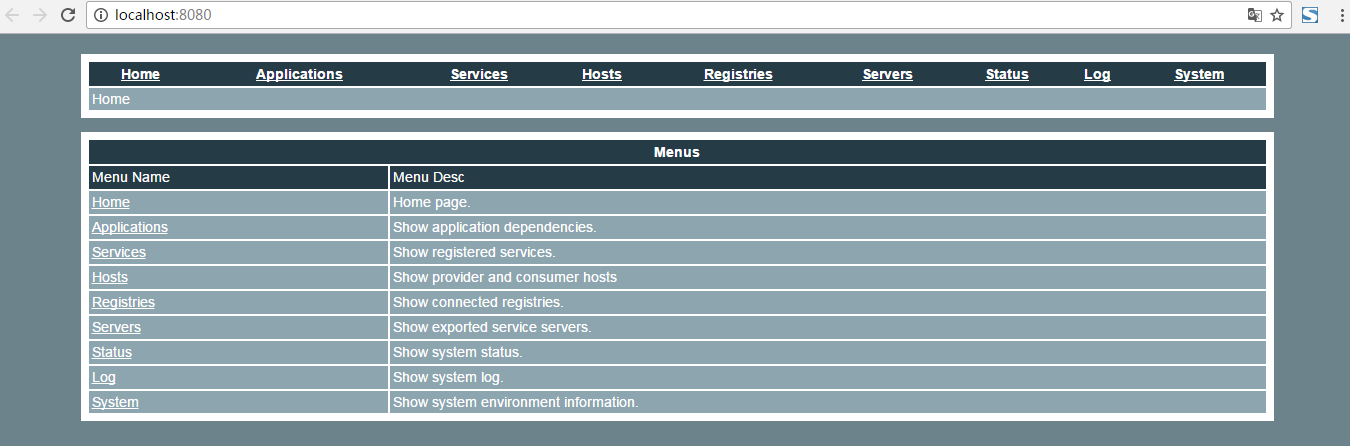
</beans>

## 启动监控：

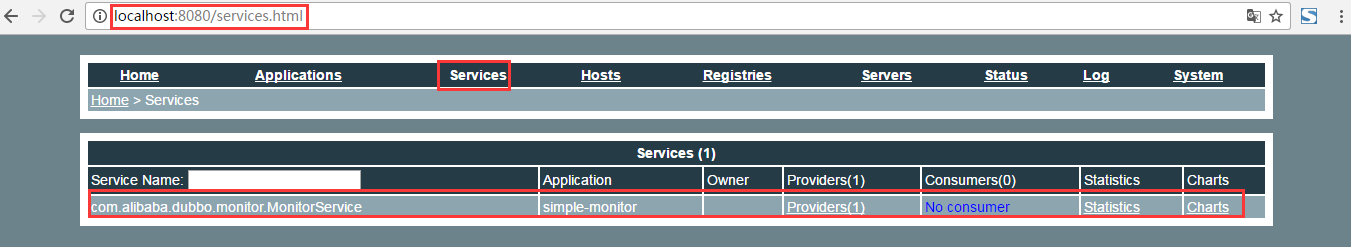




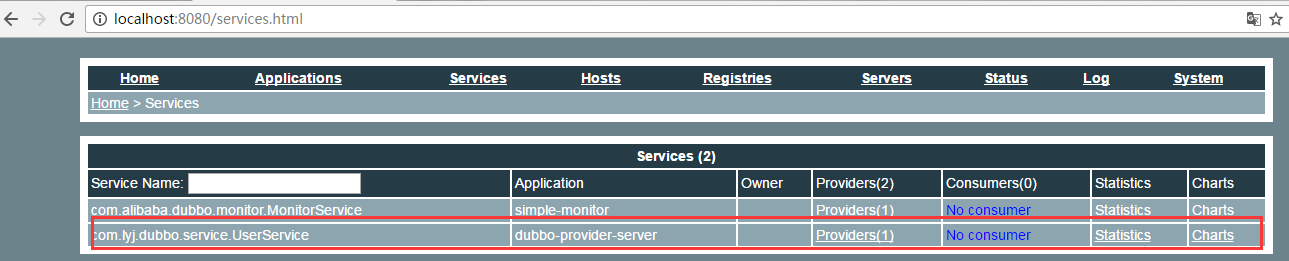
访问地址：http://localhost:8080/



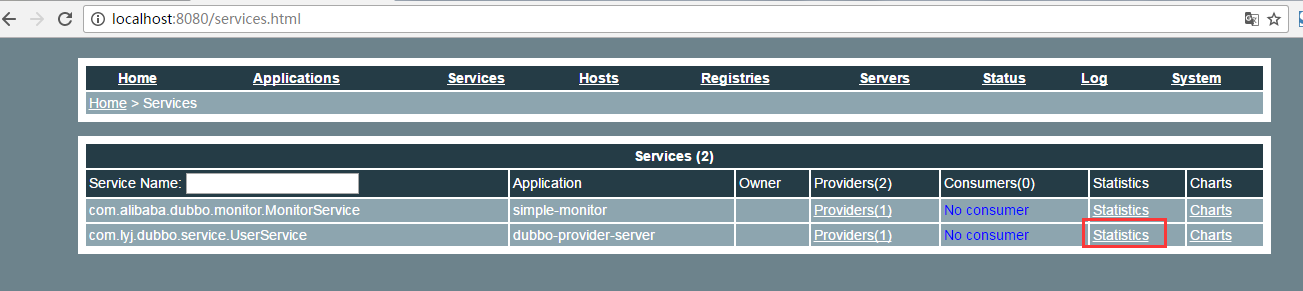
暂时只有 监控本身 一个服务

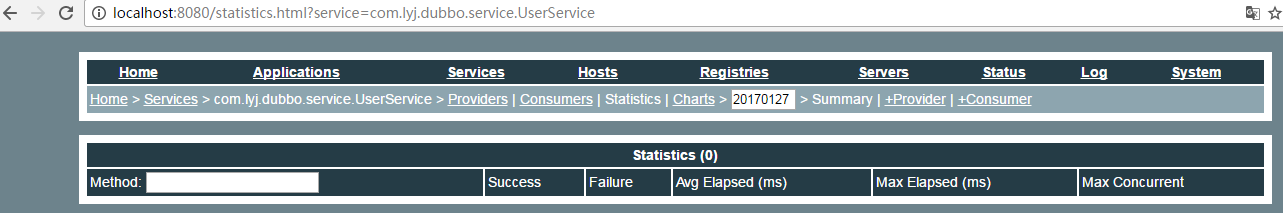


## 启动 dubbo-provider

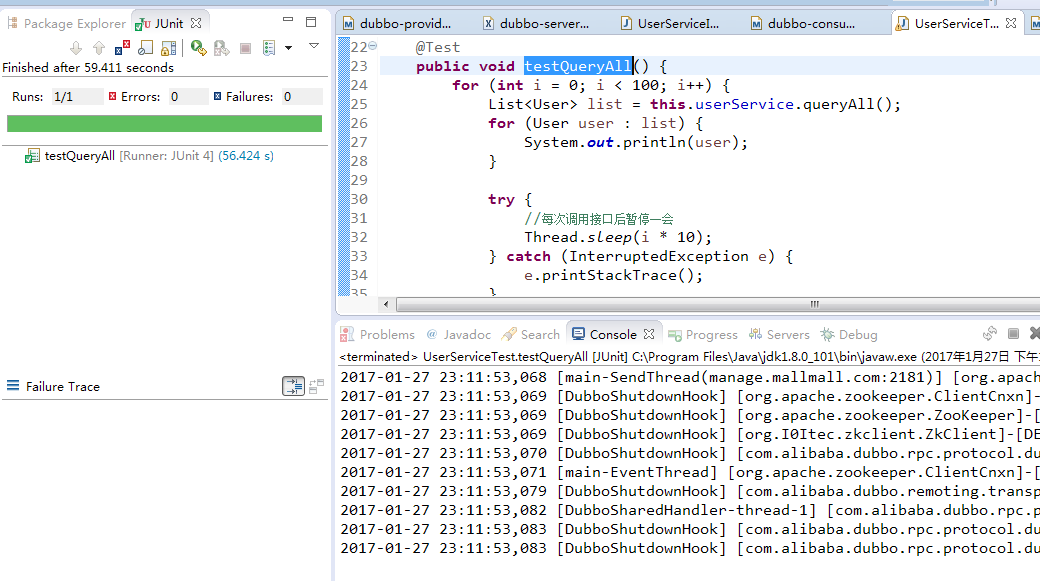


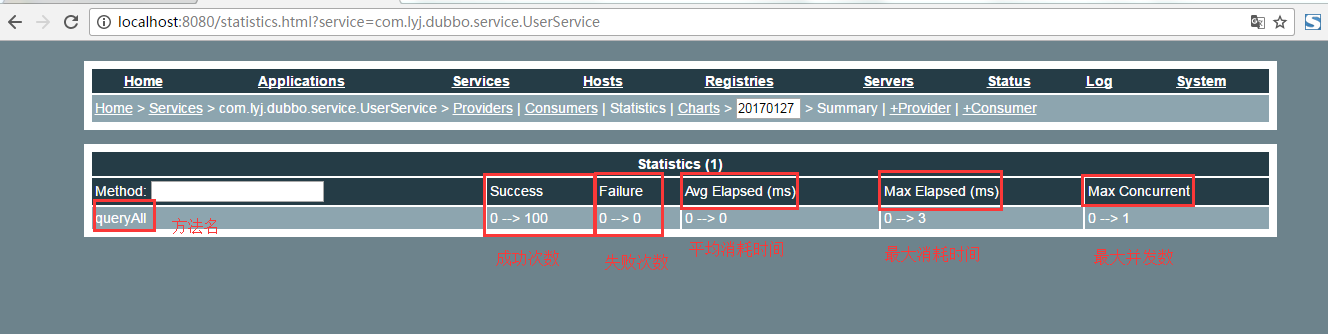
## 执行 dubbo-consumer 查看统计



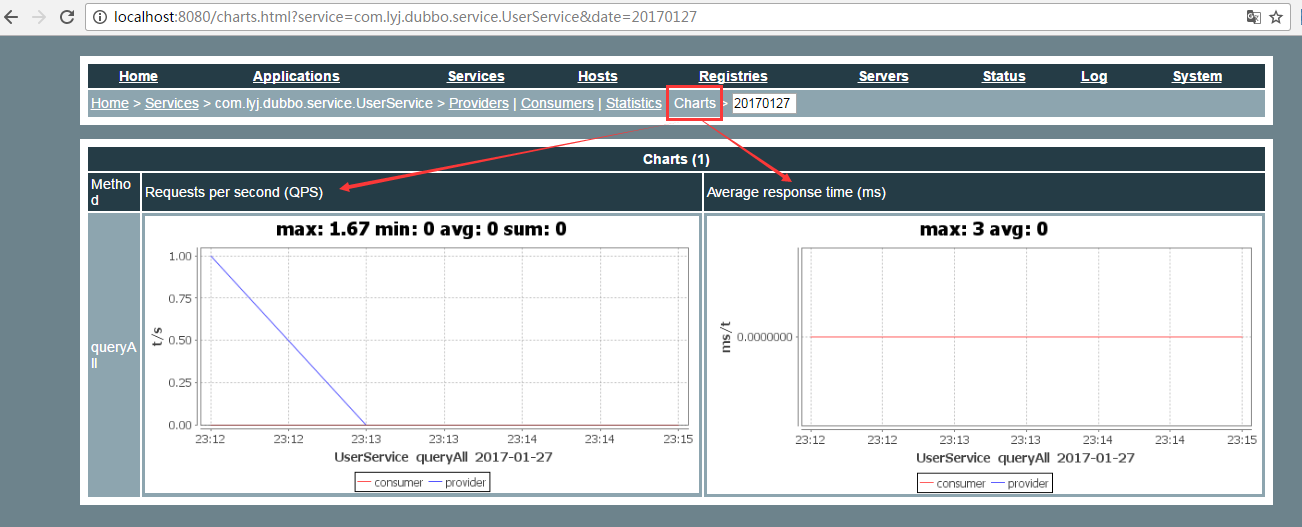


执行后：



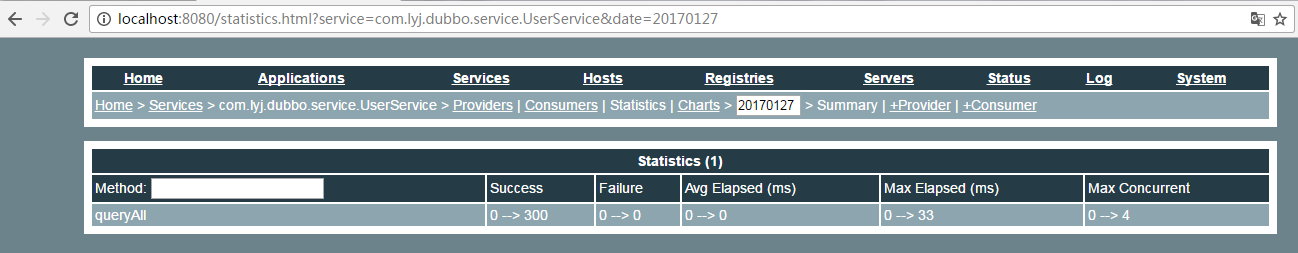


Charts图表



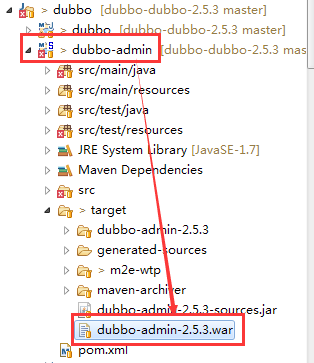
QPS（Query Per Second 每秒查询率QPS是对一个特定的查询服务器在规定时间内所处理流量多少的衡量标准） 和平均响应时间

## 同时多次执行 dubbo-consumer

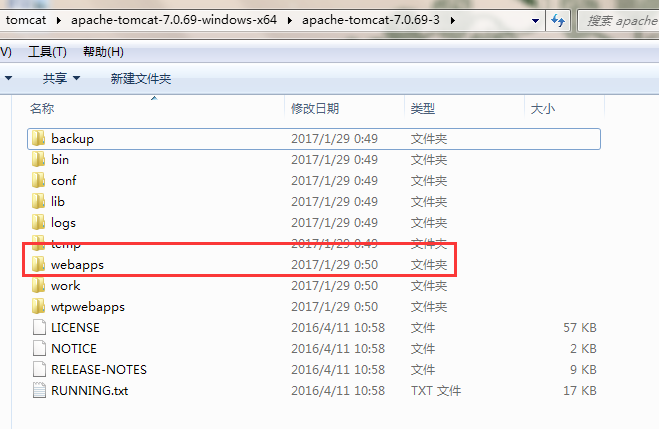


## dubbo-admin

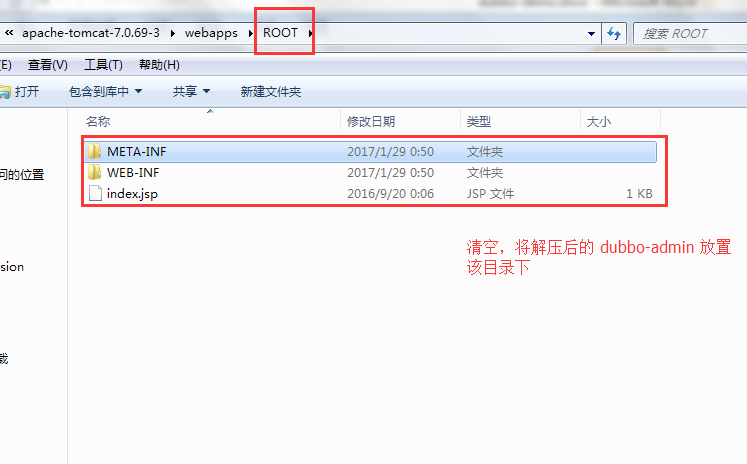
在之前构建好的 dubbo 项目下找到 dubbo-admin-2.5.3.war

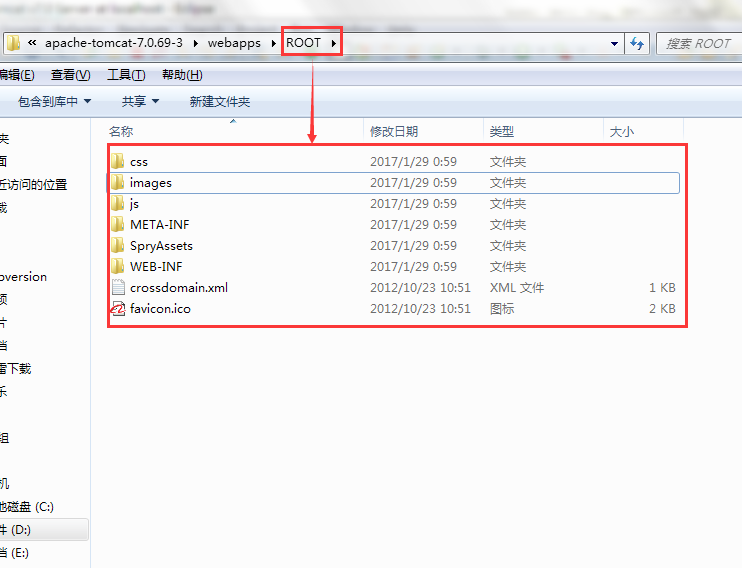


## 解压放到tomcat 下

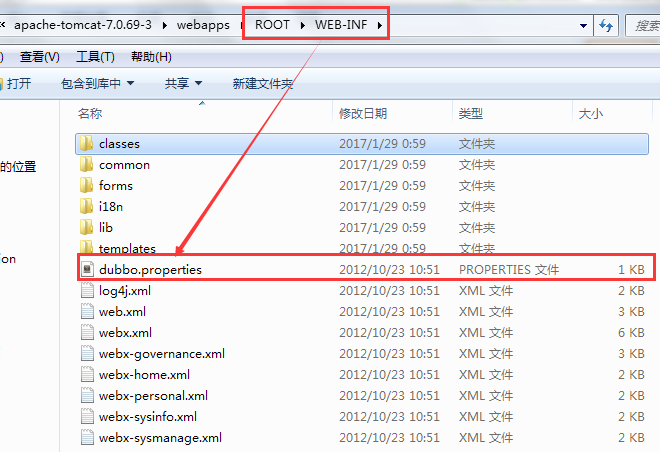


这里放到 ROOT





## 配置文件





修改成与以上配置的监控中心一致

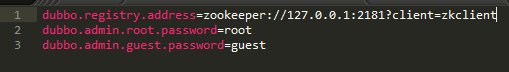
dubbo.registry.address=zookeeper://127.0.0.1:2181?client=zkclient

#root 用户密码

dubbo.admin.root.password=root

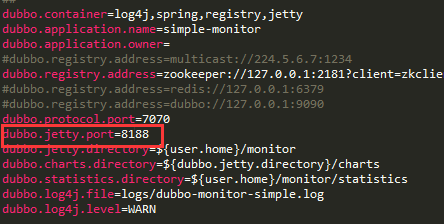
#guest 用户密码

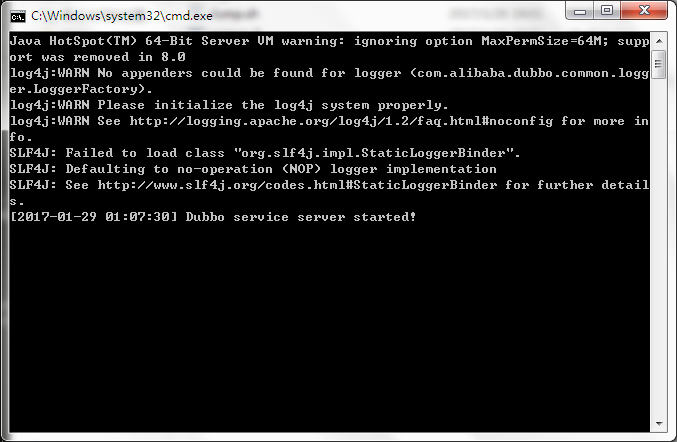
dubbo.admin.guest.password=guest



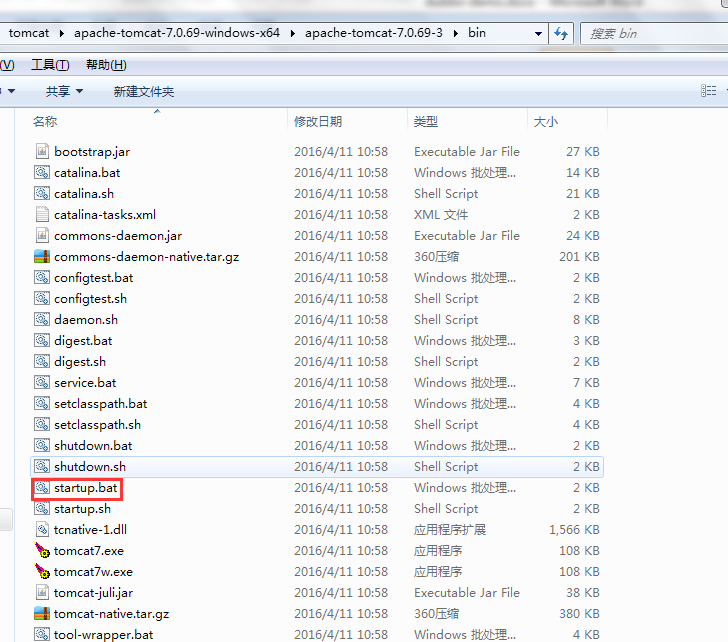
## 启动监控

修改监控端口与管理端口不冲突

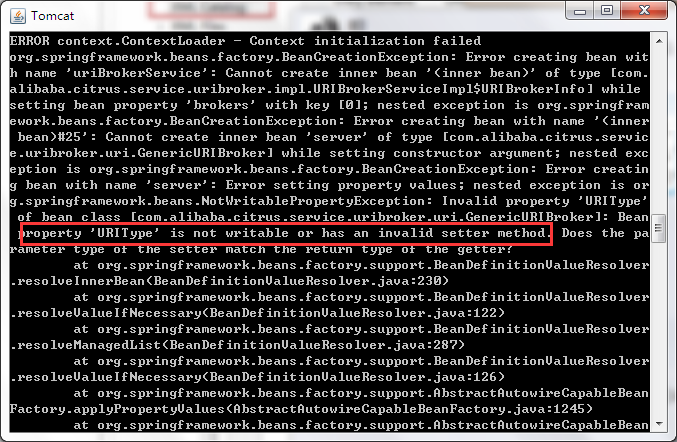




## 启动管理



### 报错 URIType is not writable…



原因是使用的 jdk8 不兼容。

### 解决方法

1. 将 jdk1.8 换成 1.7 的
2. 更改文件 重新安装 jdk1.7 到 另一个目录
3. 1) 修改bin文件夹下面的catalina.bat文件，把如下内容  
   rem ----- Execute The Requested Command ----------------------------------  
   echo Using CATALINA\_BASE: %CATALINA\_BASE%   
   echo Using CATALINA\_HOME: %CATALINA\_HOME%   
   echo Using CATALINA\_TMPDIR: %CATALINA\_TMPDIR%   
   echo Using JAVA\_HOME: **%JAVA\_HOME%**  
     
   修改为：  
   echo Using CATALINA\_BASE: %CATALINA\_BASE%   
   echo Using CATALINA\_HOME: %CATALINA\_HOME%   
   echo Using CATALINA\_TMPDIR: %CATALINA\_TMPDIR%   
   echo Using JAVA\_HOME: **D:\Java\jdk1.7.0\_79(重新安装的另一个 jdk )**  
     
   2) 修改bin文件夹下面的setclasspath.bat文件，把如下内容：   
   rem Set standard command for invoking Java.   
     
   rem Note that NT requires a window name argument when using start.   
   rem Also note the quoting as JAVA\_HOME may contain spaces.   
   set \_RUNJAVA="%JAVA\_HOME%\bin\java"   
   set \_RUNJAVAW="%JAVA\_HOME%\bin\javaw"   
   set \_RUNJDB="%JAVA\_HOME%\bin\jdb"   
   set \_RUNJAVAC="%JAVA\_HOME%\bin\javac"   
     
   修改为:   
   rem Set standard command for invoking Java.   
     
   rem Note that NT requires a window name argument when using start.   
   rem Also note the quoting as JAVA\_HOME may contain spaces.   
   set \_RUNJAVA=**"D:\Java\jdk1.7.0\_79\bin\java.exe"**

set \_RUNJDB=**"D:\Java\jdk1.7.0\_79\bin\jdb.exe"**

### 解决后