



链滴

springboot 集成 dubbo 环境的两种方式

作者: [Ouyuone](#)

原文链接: <https://ld246.com/article/1565142408961>

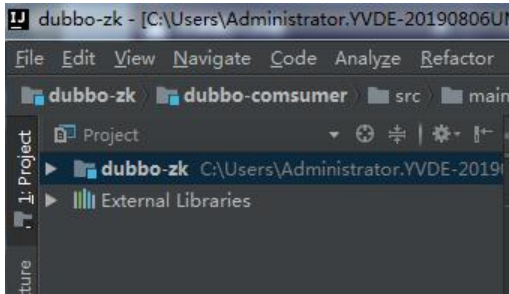
来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

1. 利用dubbo-start.jar包中@server @EnableDubbo替代使用xml

2. 利用dubbo-start.jar包中@ImportResource来使用x文件集成dubbo

第一步我们先搭建一个springboot的项目如下：



在pom文件中写入依赖：

```
<modules>
<module>dubbo-common</module>
<module>dubbo-provider</module>
</modules>
```

```
<dependencyManagement>
  <dependencies>
    <!-- dubbo 替换 dubbox-->
    <dependency>
      <groupId>com.alibaba</groupId>
      <artifactId>dubbo</artifactId>
      <version>2.6.2</version>
    </dependency>
    <!-- curator-recipes 替换 zkclient-->
    <!--<dependency>
      <groupId>org.apache.curator</groupId>
      <artifactId>curator-recipes</artifactId>
      <version>4.0.1</version>
    </dependency-->
    <!--其实我们现在使用的是curator来替换zkclient-->
    <dependency>
      <groupId>org.apache.curator</groupId>
      <artifactId>curator-framework</artifactId>
      <version>2.6.0</version>
    </dependency>
    <!--alibaba提供的启动项-->
    <dependency>
      <groupId>com.alibaba.spring.boot</groupId>
      <artifactId>dubbo-spring-boot-starter</artifactId>
      <version>2.0.0</version>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
```

```
        <artifactId>lombok</artifactId>
        <version>1.18.8</version>
    </dependency>
</dependencies>
</dependencyManagement>
```

接着我们在dubbo-zk project下新建一个module名为:



里面pom这样写:

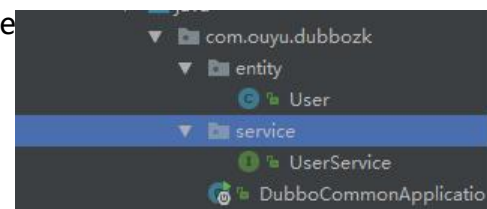
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
ven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.ouyu.dubboZk</groupId>
        <artifactId>dubbo-zk</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <artifactId>dubbo-common</artifactId>

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

    <dependencies>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
    </dependencies>

</project>
```

ok我们在common module中新建两个package一个叫entity 一个是service



```
@Data
public class User implements Serializable {
    private static final long serialVersionUID = -2257987469927365451L;
    private Integer id;
    private String userName;
    private String passWord;
    private String city;

    public User(Integer id, String userName, String passWord, String city) {
        this.id = id;
        this.userName = userName;
        this.passWord = passWord;
        this.city = city;
    }

    @Override
    public String toString() {
        return "User{" +
            "id=" + id +
            ", userName='" + userName + '\'' +
            ", passWord='" + passWord + '\'' +
            ", city='" + city + '\'' +
            '}';
    }
}
```

在service中新建个接口UserService 写入一个方法

```
public interface UserService {

    /**
     * 获取用户信息
     * @return
     */
    User getUserInfo();
}
```

ok接着来建造另一个依赖 dubbo-provider

在其pom文件中引入:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
ven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.ouyu.dubboZk</groupId>
        <artifactId>dubbo-zk</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <artifactId>dubbo-provider</artifactId>
```

```

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

<dependencies>
  <dependency>
    <groupId>com.ouyu.dubboZk</groupId>
    <artifactId>dubbo-common</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </dependency>
  <!-- dubbo 替换 dubbox-->
  <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>dubbo</artifactId>
  </dependency>
  <!-- curator-recipes 替换 zkclient-->
  <dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
  </dependency>
  <dependency>
    <groupId>com.alibaba.spring.boot</groupId>
    <artifactId>dubbo-spring-boot-starter</artifactId>
  </dependency>
  <!--<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.1.6.RELEASE</version>
  </dependency-->
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

```

```
</project>
```

在propertist文件中写

```
server.port=8081
server.context-path=
```

```
spring.application.name=dubbo-spring-boot-starter
dubbo.application.name=springboot_demo
dubbo.registry.address=zookeeper://120.79.63.27:2181 #这是你的zookeeper地址
dubbo.provider.threads=10
dubbo.provider.threadpool=fixed
dubbo.provider.loadbalance=roundrobin
dubbo.server=true
```

dubbo.protocol.name=dubbo

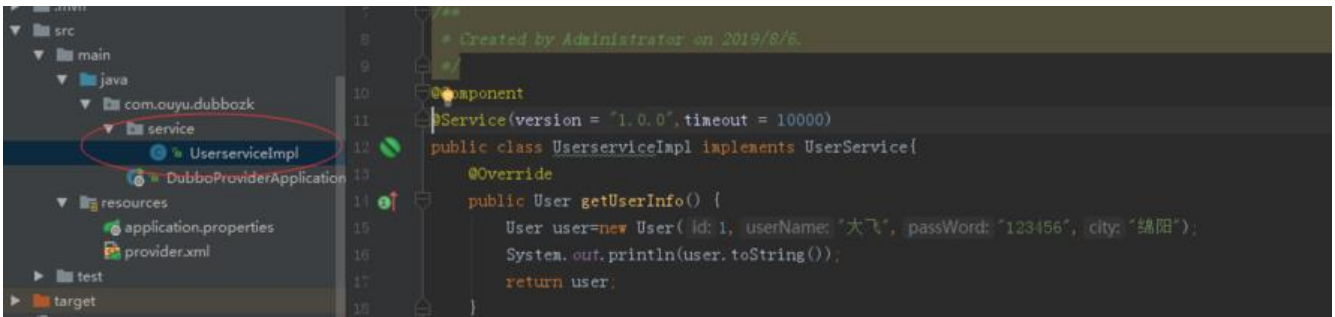
在application上加上注解@EnableDubbo

```
@SpringBootApplication
@EnableDubbo
public class DubboProviderApplication {

    public static void main(String[] args) {
        SpringApplication.run(DubboProviderApplication.class, args);
    }
}
```

在main方法中写入dubbo的启动方法Main.main(args);

接着新建servicepackage

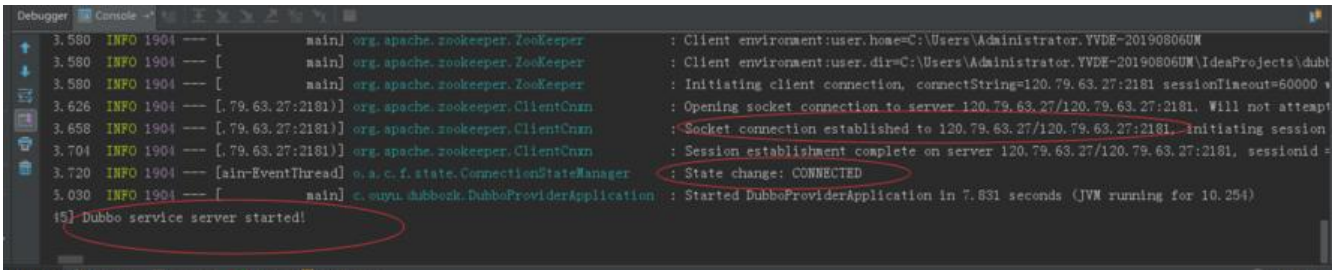


```
@Component
@Service(version = "1.0.0", timeout = 10000)
public class UserServiceImpl implements UserService {

    @Override
    public User getUserInfo() {
        User user = new User(id: 1, userName: "大飞", passWord: "123456", city: "绵阳");
        System.out.println(user.toString());
        return user;
    }
}
```

在实现类上面打上dubbo的@Service注解和spring的@Component注解以便服务的注册于发现 在@Service上写上version版本信息 等消费消费的时候会用到务必要填

服务提供端我们就完成了 现在开始启动试试



```
3.580 INFO 1904 --- [main] org.apache.zookeeper.ZooKeeper : Client environment:user.home=C:\Users\Administrator.YVDE-20190806UM
3.580 INFO 1904 --- [main] org.apache.zookeeper.ZooKeeper : Client environment:user.dir=C:\Users\Administrator.YVDE-20190806UM\IdeaProjects\dubbo
3.580 INFO 1904 --- [main] org.apache.zookeeper.ZooKeeper : Initiating client connection, connectString=120.79.63.27:2181 sessionTimeout=60000
3.626 INFO 1904 --- [79.63.27:2181] org.apache.zookeeper.ClientCnxn : Opening socket connection to server 120.79.63.27/120.79.63.27:2181. Will not attempt
3.658 INFO 1904 --- [79.63.27:2181] org.apache.zookeeper.ClientCnxn : Socket connection established to 120.79.63.27/120.79.63.27:2181. Initiating session
3.704 INFO 1904 --- [79.63.27:2181] org.apache.zookeeper.ClientCnxn : Session establishment complete on server 120.79.63.27/120.79.63.27:2181, sessionId =
3.720 INFO 1904 --- [main-EventThread] o.a.c.f.state.ConnectionStateManager : State change: CONNECTED
5.030 INFO 1904 --- [main] c.ouyu.dubbozk.DubboProviderApplication : Started DubboProviderApplication in 7.831 seconds (JVM running for 10.254)
[INFO] Dubbo service server started!
```

看到对应圈住的信息就说明ok

现在我们来把消费者搞起来 新建module dubbo-comsumer

pom中的依赖跟provider中其实都一样 只是增加了一个web的框架

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.1.6.RELEASE</version>
</dependency>
```

在properties中

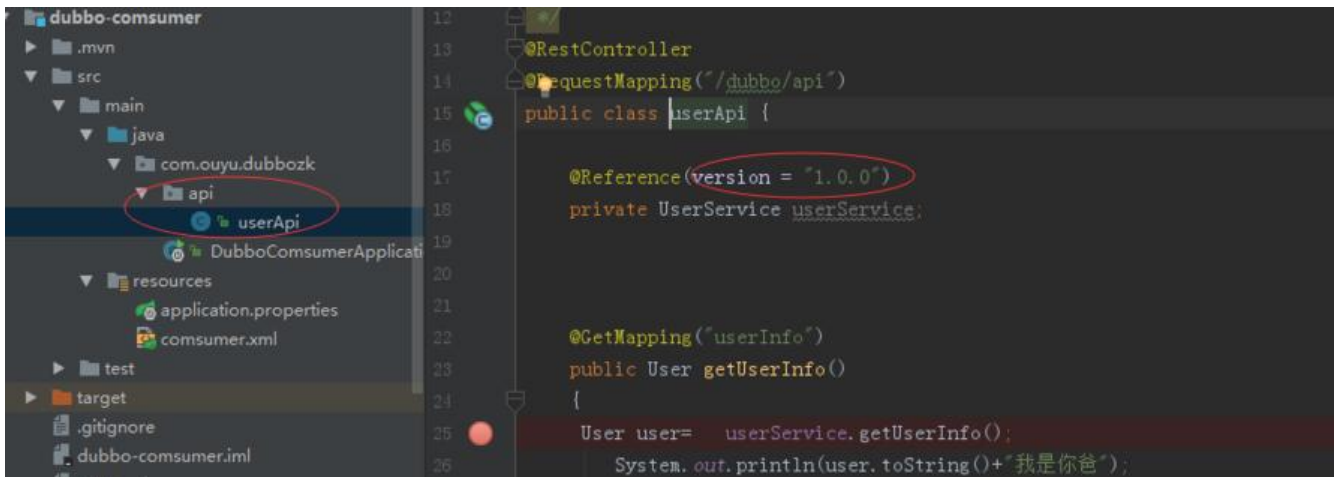
```
#server
server.port=8082
server.context-path=/
#dubbo
```

```
spring.application.name=dubbo-spring-boot-starter
```

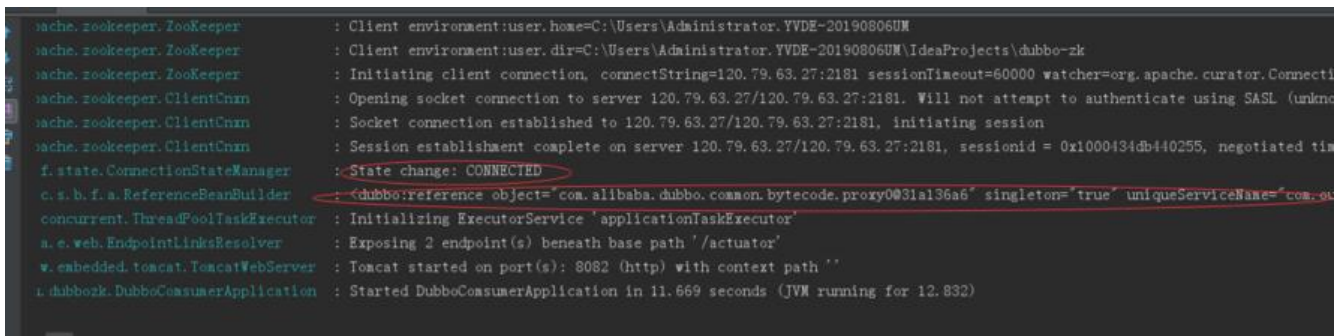
```
dubbo.rpotocol.port=20880
```

```
spring.dubbo.application.id=springboot_demo  
dubbo.application.name=springboot_demo  
dubbo.registry.address=zookeeper://120.79.63.27:2181  
dubbo.protocol.name=dubbo
```

在application中写上@EnableDubbo注解就行

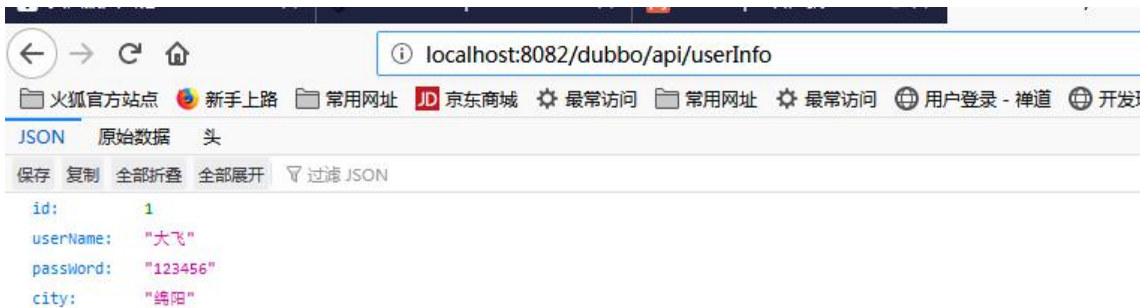


我们看到我们userService变量上加上了@Reference注解上面写上了对应服务提供者的版本信息
ok现在启动测试下



看到上面圈住的信息 就说明没毛病老铁

打开浏览器输入<http://localhost:8082/dubbo/api/userInfo>




得到返回json 大功告成 基于springboot的dubbo 没使用xml实现

二. 我们使用xml

在provider模块中把properties都注释掉了端口号 在下面新建provider.xml文件

```
<?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://dubbo.apache.org/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
  <dubbo:application name="hello-world-app" />
  <dubbo:registry address="zookeeper://120.79.63.27:2181" />
  <dubbo:protocol name="dubbo" port="20880" />
  <bean id="userService" class="com.ouyu.dubbozk.service.UserserviceImpl"/>
  <dubbo:service interface="com.ouyu.dubbozk.service.UserService" ref="userService" version="1.0.0"/>
</beans>
```

在application中隐藏点@EnableDubbo注解增加@ImportResource注解



```
@SpringBootApplication
//@EnableDubbo
@ImportResource(value = {"classpath:provider.xml"})
public class DubboProviderApplication {
```

在UserServiceImpl中把@Component @Service删掉

启动时 信息跟基于注解的一样 没问题

重点在于服务消费者模块上面

一样把他下面的properties中信息给注释掉新建consumer.xml

```
<?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://dubbo.apache.org/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
  <dubbo:application name="hello-world-app1"/>
  <dubbo:registry address="zookeeper://120.79.63.27:2181"/>
  <dubbo:reference id="userService" check="false" interface="com.ouyu.dubbozk.service.UserService" version="1.0.0"/>
</beans>
```

这个里面的version可写可不写

在application中只增加@ImportResource注解不删除@EnableDubbo注解


```
import ...

@SpringBootApplication
@EnableDubbo
@ImportResource(value = {"classpath:consumer.xml"})
public class DubboConsumerApplication {

    public static void main(String[] args) { SpringApplication
```

api提供那个地方一定不要改了 就使用@Reference(version = "1.0.0")记住加上版本信息

启动规矩的很 稳不稳的 跟注解访问一样

好了今天的讲解就到这里为止