**Dubbo**

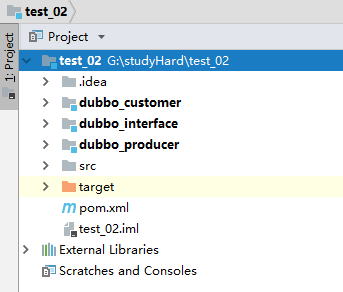
https://dubbo.gitbooks.io/dubbo-user-book/

#### Dubbo实战------管控台安装

1. 安装dubbo管控台
2. dubbo-admin下载地址：<https://github.com/apache/incubator-dubbo/releases>
3. 进入根目录运行mvn install -Dmaven.test.skip=true命令生成target目录，把 里面的war包拷贝到tomcat的ROOT（干净的）目录下
4. 启动tomcat即可访问

#### Dubbo实战------纯注解实例

1. 目录结构

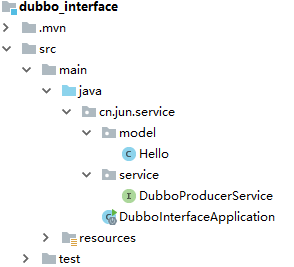


1. 首先新建一个springboot项目
2. 在该项目下新建一个model命名为dubbo\_interface，dubbo\_interface的作用是提供公共接口为服务者和消费者使用
3. 在该项目下新建一个model命名为dubbo\_provider，dubbo\_provider的作用是服务提供者，同时需要在pom.xml文件中添加dubbo与zookeeper依赖，包括对dubbo\_interface的依赖

pom.xml:

<**dependencies**>  
 <**dependency**>  
 <**groupId**>cn.jun</**groupId**>  
 <**artifactId**>dubbo\_interface</**artifactId**>  
 <**version**>0.0.1-SNAPSHOT</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 *<!-- https://mvnrepository.com/artifact/com.alibaba/dubbo -->* <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>dubbo</**artifactId**>  
 <**version**>2.5.8</**version**>  
 </**dependency**>  
 *<!-- https://mvnrepository.com/artifact/org.apache.zookeeper/zookeeper -->* <**dependency**>  
 <**groupId**>org.apache.zookeeper</**groupId**>  
 <**artifactId**>zookeeper</**artifactId**>  
 <**version**>3.4.12</**version**>  
 <**type**>pom</**type**>  
 </**dependency**>  
 *<!-- https://mvnrepository.com/artifact/org.apache.curator/curator-framework -->* <**dependency**>  
 <**groupId**>org.apache.curator</**groupId**>  
 <**artifactId**>curator-framework</**artifactId**>  
 <**version**>2.12.0</**version**>  
 </**dependency**>  
</**dependencies**>

1. 在该项目下新建一个model命名为dubbo\_customer，dubbo\_customer的作用是服务消费者，pom依赖与dubbo\_provider相同
2. 至此，项目结构搭建完成，接下来编写服务提供者与消费者。
3. 在dubbo\_interface中编写服务接口与pojo类
4. 目录结构



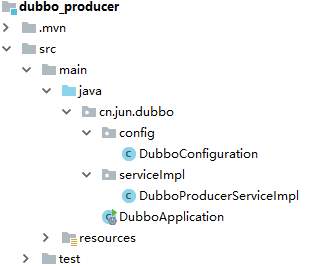
1. 创建model包和service包
2. 编写model类（注：需要序列化）

**import** java.io.Serializable;  
**public class** Hello **implements** Serializable {  
 **private** String **name**;  
 **private** String **age**;  
 **private** String **sex**;  
 **public** String getName() {  
 **return name**;  
 }  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
 **public** String getAge() {  
 **return age**;  
 }  
 **public void** setAge(String age) {  
 **this**.**age** = age;  
 }  
 **public** String getSex() {  
 **return sex**;  
 }  
 **public void** setSex(String sex) {  
 **this**.**sex** = sex;  
 }  
 @Override  
 **public** String toString() {  
 **return "Hello{"** +  
 **"name='"** + **name** + **'\''** +  
 **", age='"** + **age** + **'\''** +  
 **", sex='"** + **sex** + **'\''** +  
 **'}'**;  
 }  
}

1. 编写service接口

**import** cn.jun.service.model.Hello;  
**public interface** DubboProducerService {  
 **public** Hello sayHello();  
}

1. 在dubbo\_provider中编写服务提供者
2. 目录结构



1. 首先创建config与serviceImpl两个包
2. 在config包里面编写dubbo配置类DubboConfiguration

本配置把服务发布到zookeeper注册中心上，其中@DubboComponentScan的注解是标注扫描dubbo的配置  
**import** com.alibaba.dubbo.config.ApplicationConfig;  
**import** com.alibaba.dubbo.config.RegistryConfig;  
**import** com.alibaba.dubbo.config.spring.context.annotation.DubboComponentScan;  
**import** org.springframework.context.annotation.Bean;  
**import** org.springframework.context.annotation.Configuration;  
  
@Configuration  
@DubboComponentScan(**"cn.jun.dubbo.serviceImpl"**)  
**public class** DubboConfiguration {  
  
 @Bean  
 **public** ApplicationConfig applicationConfig() {  
 ApplicationConfig applicationConfig = **new** ApplicationConfig();  
 applicationConfig.setName(**"provider-test"**);  
 **return** applicationConfig;  
 }  
 @Bean  
 **public** RegistryConfig registryConfig() {  
 RegistryConfig registryConfig = **new** RegistryConfig();  
 registryConfig.setAddress(**"zookeeper://192.168.81.135:2181"**);  
 registryConfig.setClient(**"curator"**);  
 **return** registryConfig;  
 }  
}

1. 在serviceImpl里面编写服务实现类，实现dubbo\_interface中的公共接口

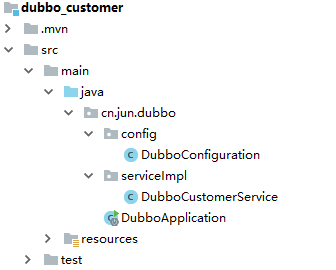
其中，@Service注解为dubbo的注解而不是spring的注解，该注解的作用是发布服务

**package** cn.jun.dubbo.serviceImpl;  
**import** cn.jun.service.model.Hello;  
**import** cn.jun.service.service.DubboProducerService;  
**import** com.alibaba.dubbo.config.annotation.Service;  
@Service  
**public class** DubboProducerServiceImpl **implements** DubboProducerService {  
 @Override  
 **public** Hello sayHello() {  
 Hello hello = **new** Hello();  
 hello.setName(**"小花"**);  
 hello.setAge(**"19"**);  
 hello.setSex(**"男"**);  
 **return** hello;  
 }  
}

1. Springboo的启动类，@ComponentScan扫描spring的注解

**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
**import** org.springframework.context.annotation.ComponentScan;  
@SpringBootApplication  
@ComponentScan(**"cn.jun.dubbo.config"**)  
**public class** DubboApplication {  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(DubboApplication.**class**, args);  
 }  
}

1. 至此服务提供者搭建完毕
2. 在dubbo\_consumer中编写服务消费者
3. 目录结构



1. 首先创建config包与serviceImpl包
2. 在config包下编写消费者配置类

**import** com.alibaba.dubbo.config.ApplicationConfig;  
**import** com.alibaba.dubbo.config.ConsumerConfig;  
**import** com.alibaba.dubbo.config.RegistryConfig;  
**import** com.alibaba.dubbo.config.spring.context.annotation.DubboComponentScan;  
**import** org.springframework.context.annotation.Bean;  
**import** org.springframework.context.annotation.Configuration;  
@Configuration  
@DubboComponentScan(basePackages = **"cn.jun.dubbo.serviceImpl"**)  
**public class** DubboConfiguration {  
 @Bean  
 **public** ApplicationConfig applicationConfig() {  
 ApplicationConfig applicationConfig = **new** ApplicationConfig();  
 applicationConfig.setName(**"consumer-test"**);  
 System.***out***.println(**"aa"**);  
 **return** applicationConfig;  
 }  
 @Bean  
 **public** ConsumerConfig consumerConfig() {  
 ConsumerConfig consumerConfig = **new** ConsumerConfig();  
 consumerConfig.setTimeout(3000);  
 System.***out***.println(**"bb"**);  
 **return** consumerConfig;  
 }  
 @Bean  
 **public** RegistryConfig registryConfig() {  
 RegistryConfig registryConfig = **new** RegistryConfig();  
registryConfig.setAddress(**"zookeeper://192.168.81.135:2181"**);  
 registryConfig.setClient(**"curator"**);  
 System.***out***.println(**"cc"**);  
 **return** registryConfig;  
 }  
}

1. 在serviceImpl包下编写服务消费者，其中@Service的注解是正常的spring注解

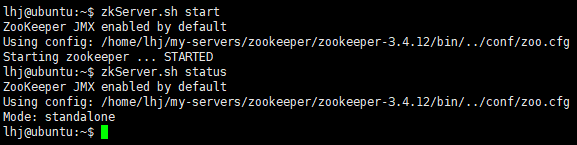
引用dubbo的服务需要@Reference注解，按照dubbo帮助文档写的时候，@Reference没有加url参数是会报空指针异常的，后来加上url参数可以正常获取服务，其中20880端口为dubbo服务的默认端口，该端口也可以自己指定。

**import** cn.jun.service.model.Hello;  
**import** cn.jun.service.service.DubboProducerService;  
**import** com.alibaba.dubbo.config.annotation.Reference;  
**import** org.springframework.stereotype.Service;  
@Service  
**public class** DubboCustomerService{  
 @Reference(url = **"dubbo://127.0.0.1:20880"**)  
 DubboProducerService **dubboProducerService**  
 **public void** hello() {  
 Hello hello = **dubboProducerService**.sayHello();  
 System.***out***.println(**"你好呀！我是"**+hello.getName()+**"！我今年"**+hello.getAge()+**"岁了！我是"**+hello.getSex()+**"生哦！"**);  
 }  
}

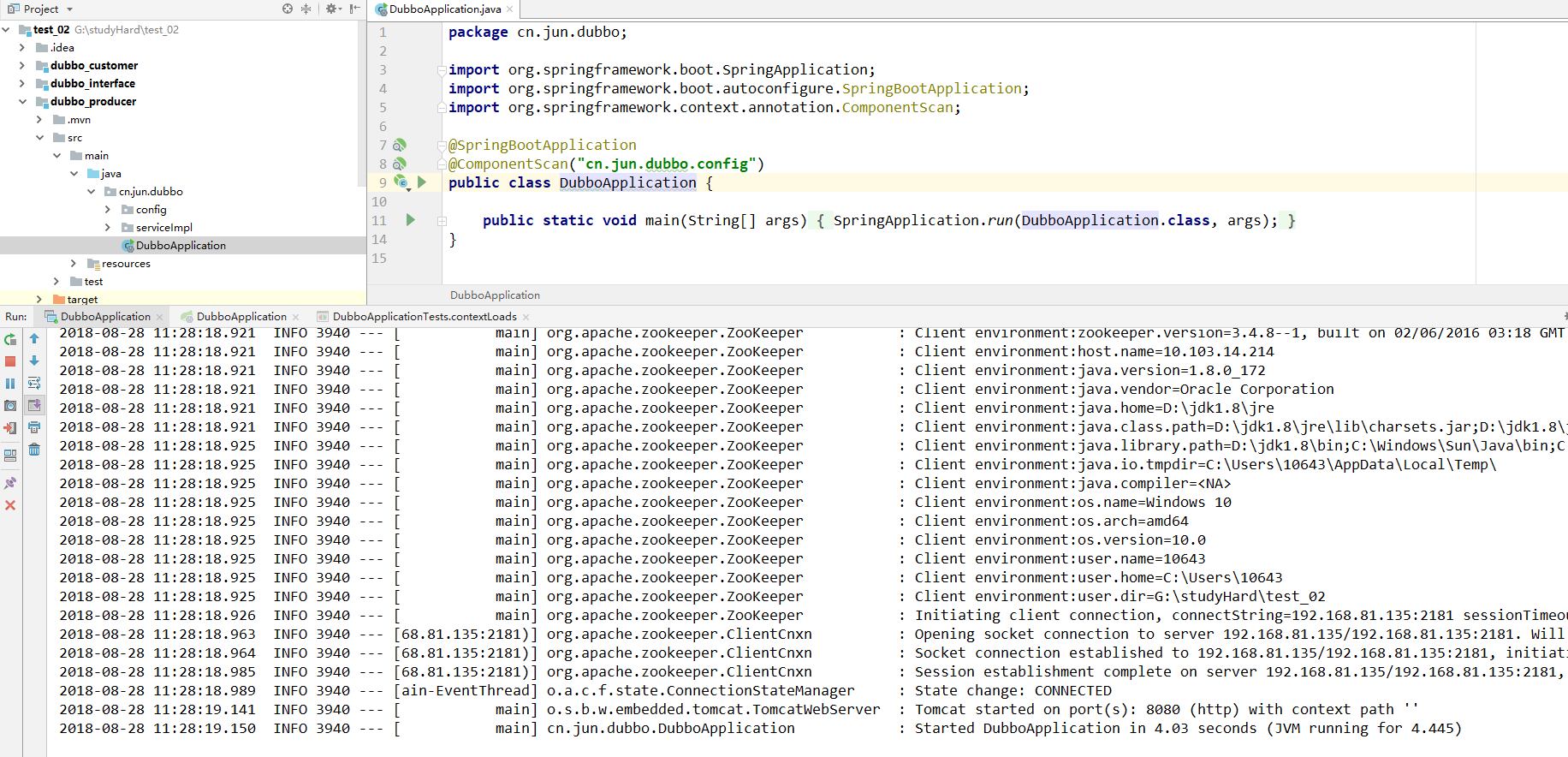
1. 编写单元测试

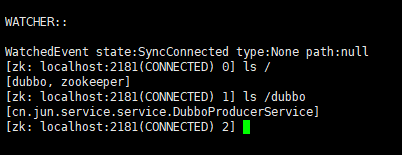
**import** cn.jun.dubbo.serviceImpl.DubboCustomerService;  
**import** org.junit.Test;  
**import** org.junit.runner.RunWith;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.boot.test.context.SpringBootTest;  
**import** org.springframework.context.annotation.ComponentScan;  
**import** org.springframework.test.context.junit4.SpringRunner;  
@RunWith(SpringRunner.**class**)  
@SpringBootTest  
@ComponentScan(**"cn.jun.dubbo"**)  
**public class** DubboApplicationTests {  
 @Autowired  
 DubboCustomerService **dubboCustomerService**;  
 @Test  
 **public void** contextLoads() {  
 **dubboCustomerService**.hello();  
 }  
}

1. 至此服务消费者搭建完毕
2. 现在进行测试
3. 首先启动zookeeper服务

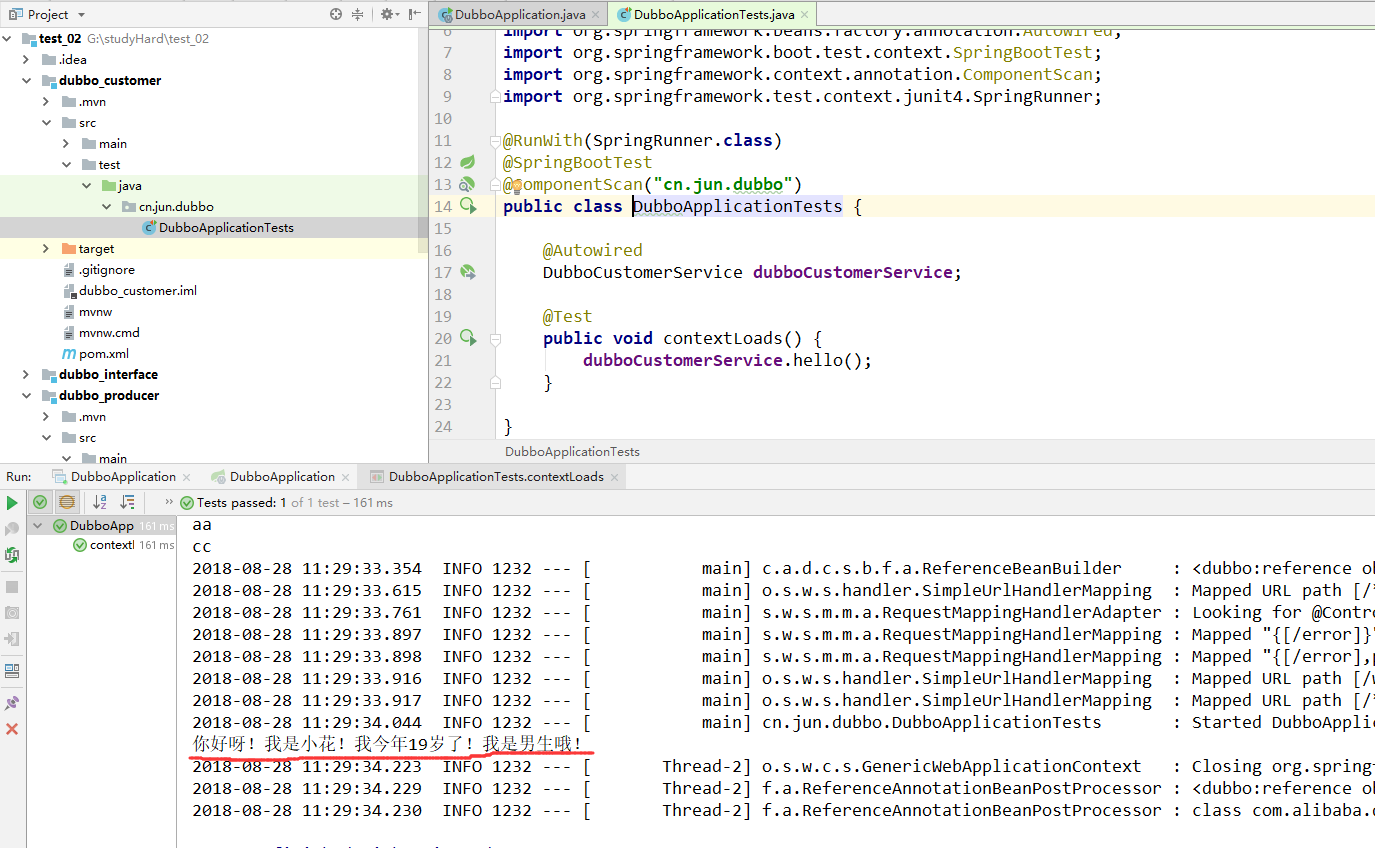


1. 启动服务提供者，可以看到运行正常，同时在服务器上进入zookeeper客户端可以看到服务已经注册成功





1. 启动服务消费者的单元测试类，可以看到正常输出了结果



1. 至此，一个完整的基于springboot+dubbo+zookeeper的纯注解的helloworld实例搭建完毕。