Dubbo使用Quick-start

# 简单案例

Dubbo官网：<http://dubbo.apache.org/>

This guide gets you started with dubbo **in Java** with a simple working example. You could find the complete working samples from directory ‘dubbo-demo’ in [dubbo project](https://github.com/apache/incubator-dubbo) on github.

It’s officially suggested that we should use XML configuration file **because it depends on a Spring container** (currently Spring 4.3.10).

**官方建议使用xml配置文件，因为Dubbo依赖Spring Container**。

# 用户手册：

英文：<http://dubbo.io/books/dubbo-user-book-en/>

中文：<http://dubbo.io/books/dubbo-user-book/>

# Dubbo测试需要的Maven依赖

## dubbo依赖

## zookeeper依赖与ziClient依赖

在Maven的Repository：搜索zookeeper、zkclient即可。

<dependency>

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

<artifactId>**zookeeper**</artifactId>

<version>3.4.11</version>

</dependency>

<dependency>

<groupId>**com.101tec**</groupId>

<artifactId>**zkclient**</artifactId>

<version>0.10</version>

</dependency>

## 必须添加org.apache.curato依赖，否则报错



**<dependency>**

**<groupId>org.apache.curator</groupId>**

**<artifactId>curator-framework</artifactId>**

**<version>4.0.0</version>**

**</dependency>**

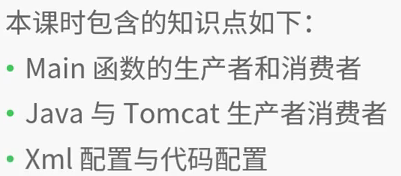
## 还需要spring的相应依赖（dubbo中也依赖spring包，注意版本冲突问题）

spring-core、spring-beans、spring-context、spring-beans、spring-aop等

在Maven的Repository网站搜索即可。Maven仓库：<http://mvnrepository.com/>

## 需要junit测试的，需要依赖junit

1. **Dubbo使用方式**

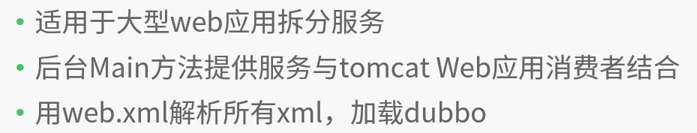


1. **Main函数的生产者和消费者**

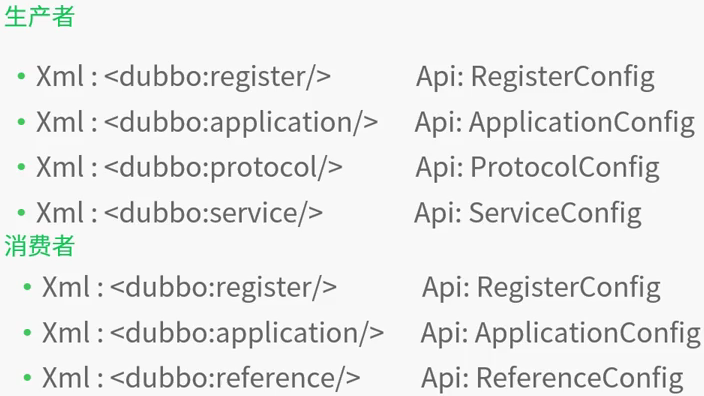


使用**Main函数**启动服务提供方，需要将dubbo-provider.xml文件放置在默认路径：META-INF/spring/目录下。

1. **Java与Tomcat生产者消费者**



1. **Xml配置与代码配置**



# 工程的实现步骤

## 分成三步骤

服务提供方（生产者）配置及启动

消费者配置及启动

生产者消费者的整合

官网介绍步骤：

### Maven dependency

### Define service interface

### Implement service provider

### Configure service provider

### Start service provider

### Configure service consumer

### Run service consumer

## 实现思路

### 创建服务，需要共享接口文件

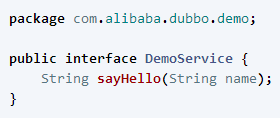
简历一个存放接口的Maven工程中，这个Maven工程中只需要定义对应的接口即可；该工程也可以导出一个jar包。

服务提供方工程与消费者工程都需要依赖这个Maven工程，**在pom.xml文件中添加依赖，或者直接添加接口Maven工程对应的jar包**。

也就是说，定义的接口interfaces需要在服务提供方和消费者共享。

官网介绍：**Define service interface**

Since both service provider and service consumer rely on the same interface, it is strongly recommended to put the interface definition below in one separated module which could be shared by both provider module and consumer module.

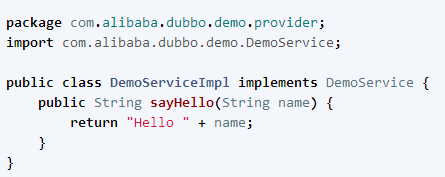


### 服务提供方Provider：

在Provider工程中，需要依赖**接口工程或jar包**，然后实现对应的接口。

在Consumer工程中，只需要依赖**接口工程或jar包即可，**不需要对其进行实现。

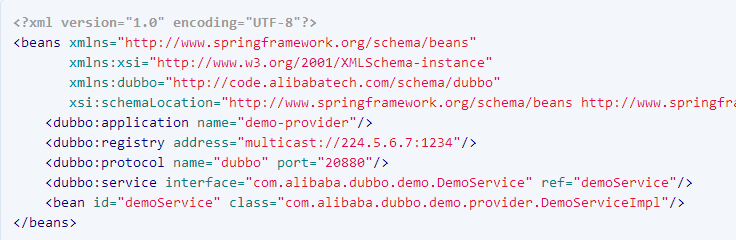
Implement service provider



# 生产者（又叫服务提供方）Provider

### 官网获取xml文件的sample

<http://dubbo.apache.org/>



<?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">

<dubbo:application name="zookeeper-provider"/>

<dubbo:provider registry="myRegistry" protocol="myProtocol"/>

**<dubbo:registry id="myRegistry" address="127.0.0.1:2181" protocol="zookeeper"/>**

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

<bean id="myPeople" class="xin.zhaohong.impl.PeopleImpl"/>

<dubbo:service interface="xin.zhaohong.interfaces.People" ref="myPeople"/>

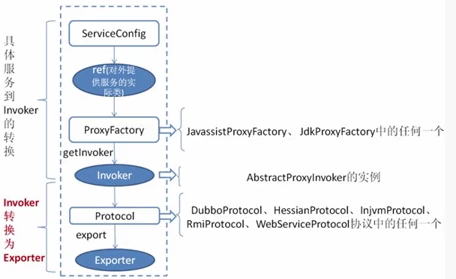
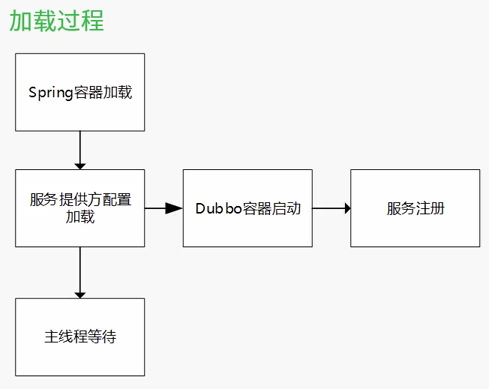
<bean id="myStudent" class="xin.zhaohong.impl.StudentImpl"/>

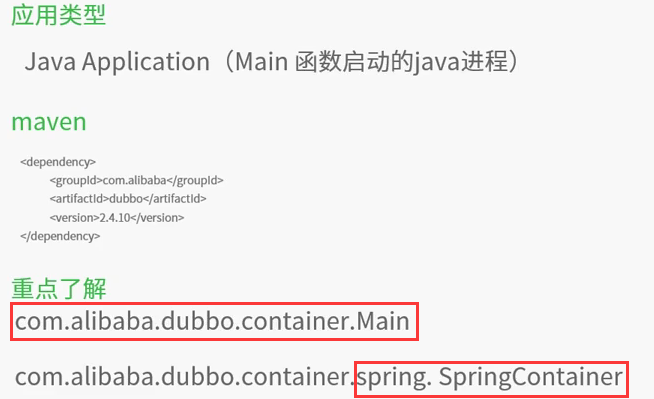
<dubbo:service interface="xin.zhaohong.interfaces.Student" ref="myStudent"/>

</beans>

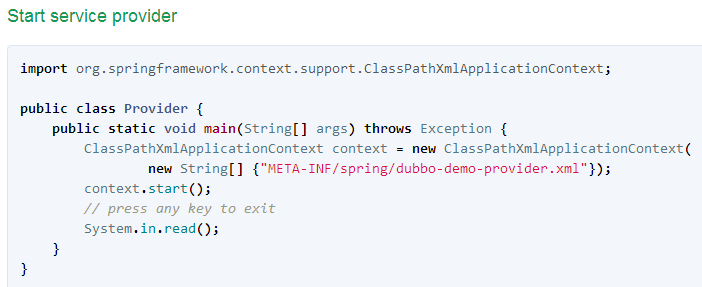
### 加载xml文件

本课时介绍**生产者**的作用，以及使用 **Xml 配置**加上注解来配置一个 **Main 函数启动的生产者以及启动检测**。



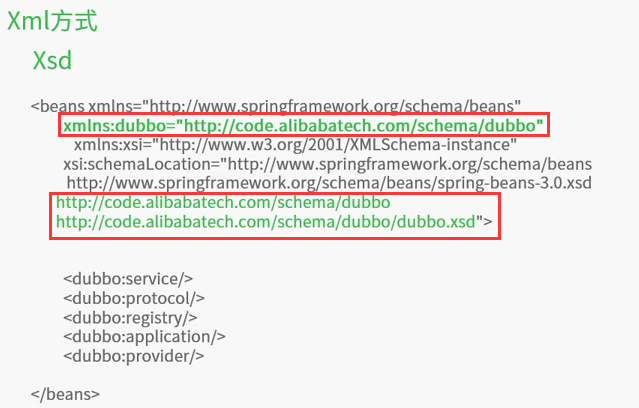






### 生产者的配置标签：

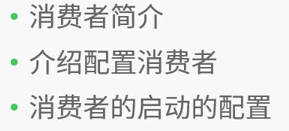
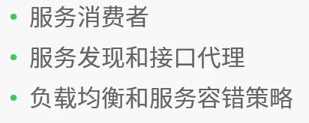
添加dubbo.xsd



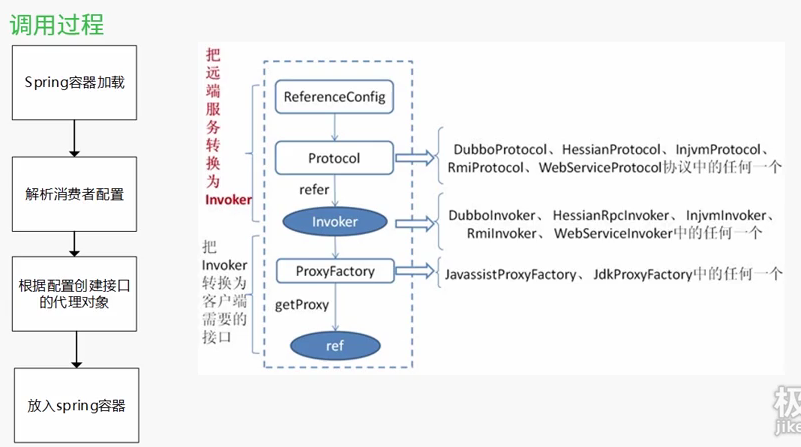


通过service配置，实现**接口服务**的配置。

# 消费者

## 消费者的调用过程



应用类型Java Application、Web、Xml配置

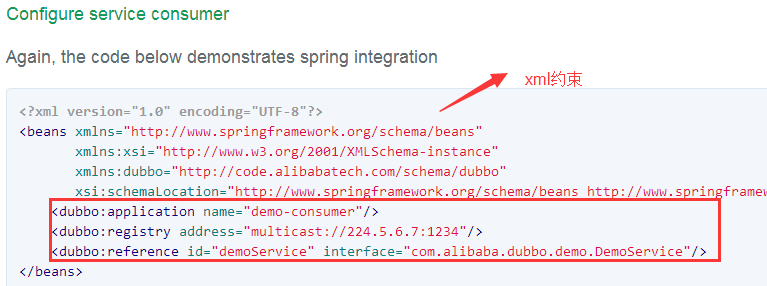
## Java Application、Web应用



## Xml方式配置

### 官网获取xml的sample

Dubbo的消费者Consumer的配置文件约束（sample）从官网获取：



<?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">

<dubbo:application name="zookeeper-consumer"/>

<dubbo:registry id="myRegistry" address="127.0.0.1:2181" protocol="zookeeper"/>

<dubbo:consumer registry="myRegistry"/>

**<dubbo:reference** id = "myPeople" interface = "xin.zhaohong.interfaces.People"/>

**<dubbo:reference** id = "myStudent" interface = "xin.zhaohong.interfaces.Student"/>

</beans>

### 标签介绍：



、

<dubbo:application name="zookeeper-consumer"/>

<dubbo:registry id="myRegistry" address="127.0.0.1:2181" protocol="zookeeper"/>

<dubbo:consumer registry="myRegistry"/>

**<dubbo:reference** id = "myPeople" interface = "xin.zhaohong.interfaces.People"/>

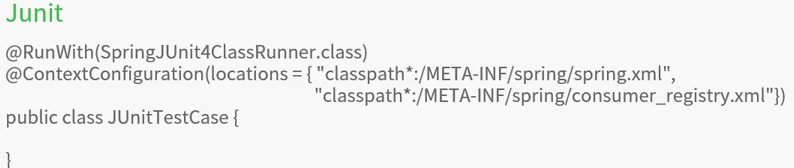
**<dubbo:reference** id = "myStudent" interface = "xin.zhaohong.interfaces.Student"/>

1. <dubbo:application ：用于计算依赖关系，每个应用命名尽量独立；
2. <dubbo:registry：配置 注册中心；
3. **<dubbo:consumer** ：抽象reference中公共的部分，作为默认配置，不必要。
4. **<dubbo:reference**：服务注册配置，指明提供服务的类。（关键）

### 加载xml文件



### Junit测试



**@RunWith(SpringJUnit4ClassRunner.class)**

**//applicationContext.xml必须写这个名字**

**@ContextConfiguration(locations= {"classpath\*:META-INF/applicationContext.xml",**

**"classpath\*:META-INF/spring-dubbo-consumer.xml"**

**})**

public class DubboTest {

**@Autowired**

**private People myPeople ;**

**@Autowired**

**private Student myStudent** ;

@Test

public void test() {

System.out.println("开始测试Dubbo");

System.out.println(myPeople.getAge());

System.out.println(myPeople.getName());

System.out.println("开始测试Dubbo");

System.out.println(myStudent.getAge());

System.out.println(myStudent.getName());

}

# 出现的错误异常及解决办法

## Caused by: java.lang.ClassNotFoundException: org.apache.curator.RetryPolicy

必须添加org.apache.curato依赖，否则报错



**<dependency>**

**<groupId>org.apache.curator</groupId>**

**<artifactId>curator-framework</artifactId>**

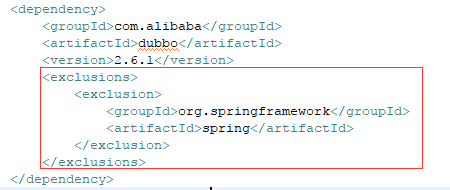
**<version>4.0.0</version>**

**</dependency>**

## java.lang.NoSuchMethodError: org.springframework.core.annotation.AnnotatedElementUtils.getAnnotationAttributes

这种一般都是spring版本不兼容，或存在冲突问题。

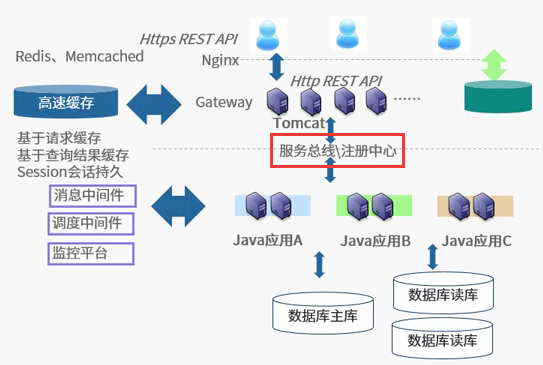
dubbo中的spring版本比较低，排除dubbo中的spring版本，以免带来冲突。

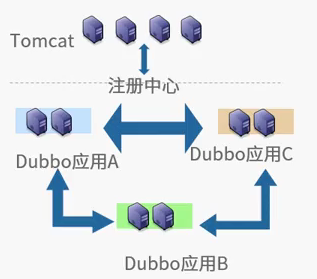


**另外，spring版本使用不要太高，选择4.x.x的还是比较稳定的。**

# Dubbo的混合型应用及配置

## 分布式服务示意图：





## 混合型应用

混合型应用：指的是一个应用不仅是服务提供方（Provider）也是服务调用方（Consumer）。

在配置xml文件时，一般是对consumer和provider分开配置，两个配置文件。采用Java Application即Dubbo自带的Main函数启动，xml配置文件默认存放在META-INF/spring/目录下。



## 混合型应用的注意事项

**循环依赖**：最值得注意的，A服务依赖B服务，而B服务又依赖A服务，有可能循环依赖。

**超时时间**：一般都是**服务调用链**，A调用B，而B调用C，但是超时时间多数是统一配置的，所以需要底层服务高效些，以免致使**上游服务**调用时间过长。

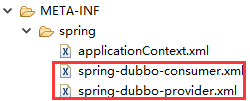
**多线程**：

**网络环境**：



## Dubbo-混合型的xml文件配置问题

### consumer与provider的xml配置文件分开



### xml配置内容注意：

一个工程只能具有一个**<dubbo:application**；因此在这两个配置文件中配置一个（任意一个文件）就可以，否则会报异常**java.lang.IllegalStateException: Duplicate application configs**。

**<dubbo:registry、<dubbo:protocol** 是一样的，在一个xml文件中配置即可（任意一个就可以），必要情况下，可以设置id来唯一标识registry与protocol。

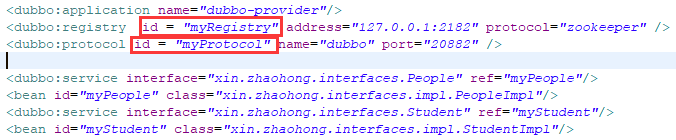
区别：

消费者中重点配置需要调用的服务：**<dubbo:reference**；

服务提供方重点配置提供的服务：**一个<bean对应一个<dubbo:service**。

### 示例：

spring-dubbo-provider.xml

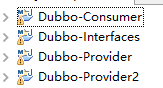


spring-dubbo-consumer.xml



## 混合型示例

四个工程：



Dubbo-Interfaces只负责定义接口；

Dubbo-Provider2中只提供服务；

Dubbo-Provider中既提供服务也向Dubbo-Provider2调用服务。

Dubbo-Consumer为服务调用。

大体流程：Dubbo-Consumer向Dubbo-Provider调用服务，Dubbo-Provider又向Dubbo-Provider2调用服务。