

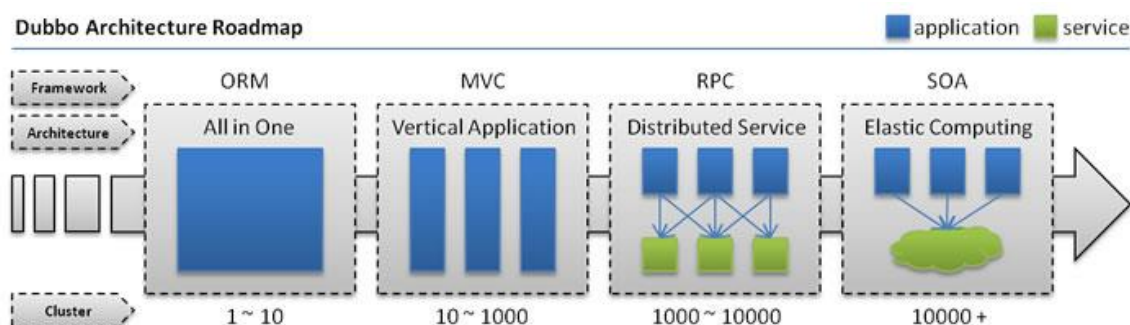
Dubbo 入门与监控中心安装

1、背景

网址: <http://dubbo.apache.org>

2018 年 2 月 15 日, 阿里巴巴的服务治理框架 dubbo 通过投票, 顺利成为 Apache 基金会孵化项目

随着互联网的发展, 网站应用的规模不断扩大, 常规的垂直应用架构已无法应对, 分布式服务架构以及流动计算架构势在必行, 亟需一个治理系统确保架构有条不紊的演进。



- **单一应用架构**
 - 当网站流量很小时, 只需一个应用, 将所有功能都部署在一起, 以减少部署节点和成本。
 - 此时, 用于简化增删改查工作量的 **数据访问框架(ORM)** 是关键。
- **垂直应用架构**
 - 当访问量逐渐增大, 单一应用增加机器带来的加速度越来越小, 将应用拆成互不相干的几个应用, 以提升效率。
 - 此时, 用于加速前端页面开发的 **Web 框架(MVC)** 是关键。
- **分布式服务架构**
 - 当垂直应用越来越多, 应用之间交互不可避免, 将核心业务抽取出来, 作为独立的服务, 逐渐形成稳定的服务中心, 使前端应用能更快速的响应多变的市场需求。
 - 此时, 用于提高业务复用及整合的 **分布式服务框架(RPC)** 是关键。
- **流动计算架构**
 - 当服务越来越多, 容量的评估, 小服务资源的浪费等问题逐渐显现, 此时需增加一个调度中心基于访问压力实时管理集群容量, 提高集群利用率。
 - 此时, 用于提高机器利用率的 **资源调度和治理中心(SOA)** 是关键。

2、RPC 基本概念

2.1、RPC 协议(Remote Procedure Call Protocol)

远程过程调用协议，它是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。RPC 协议假定某些传输协议的存在，如 TCP 或 UDP，为通信程序之间携带信息数据。在 OSI 网络通信模型中，RPC 跨越了传输层和应用层。RPC 使得开发包括网络分布式多程序在内的应用程序更加容易。

RPC 采用客户机/服务器模式。请求程序就是一个客户机，而服务提供程序就是一个服务器。首先，客户机调用进程发送一个有进程参数的调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，发送答复信息，然后等待下一个调用信息，最后，客户端调用进程接收答复信息，获得进程结果，然后调用执行继续进行。

2.2、RPC 框架

在单机时代一台电脑运行多个进程，进程之间无法通讯，显然这会浪费很多资源，因此后来出现 IPC(Inter-process communication：单机中运行的进程之间的相互通信)，这样就能允许进程之间进行通讯，比如在一台计算机中的 A 进程写了一个吃饭的方法，那在以前如果在 B 进程中也要有一个吃饭的方法，必须要在 B 进程中进行创建，但有了 RPC 后 B 只需要调用 A 进程的程序即可完成，再到后来网络时代的出现，大家电脑都连起来，这时可不可以调用其他电脑上的进程呢，当然可以，这样 RPC 框架就出现了。严格意义上来讲：Unix 的生态系统中 RPC 可以在同一台电脑上不同进程进行，也可以在不同电脑上进行；而在 windows 里面同一台电脑上不同进程间的通讯还可以采用 LPC(本地访问)。综上：RPC 或 LPC 是上层建筑，IPC 是底层基础。

RPC 框架有很多：比如 JAVA RMI、Thrift、Dubbo、grpc 等。

2.3、RPC 与 HTTP、TCP、UDP、Socket 的区别

TCP/UDP：都是传输协议，主要区别是 tcp 协议连接需要 3 次握手，断开需要四次挥手，是通过流来传输的，就是确定连接后，一直发送信息，传完后断开。udp 不需要进行连接，直接把信息封装成多个报文，直接发送。所以 udp 的速度更快，但是不保证数据的完整性。

Http：超文本传输协议是一种应用层协议，建立在 TCP 协议之上

Socket：是在应用程序层面上对 TCP/IP 协议的封装和应用。其实是一个调用接口，方便程序员使用 TCP/IP 协议栈而已。程序员通过 socket 来使用 tcp/ip 协议。但是 socket 并不是一定要使用 tcp/ip 协议，Socket 编程接口在设计的时候，就希望能适应其他的网络协议。

RPC 是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。所以 RPC 的实现可以通过不同的协议去实现比如可以使 http、RMI 等。

2.4、RPC 的运行流程

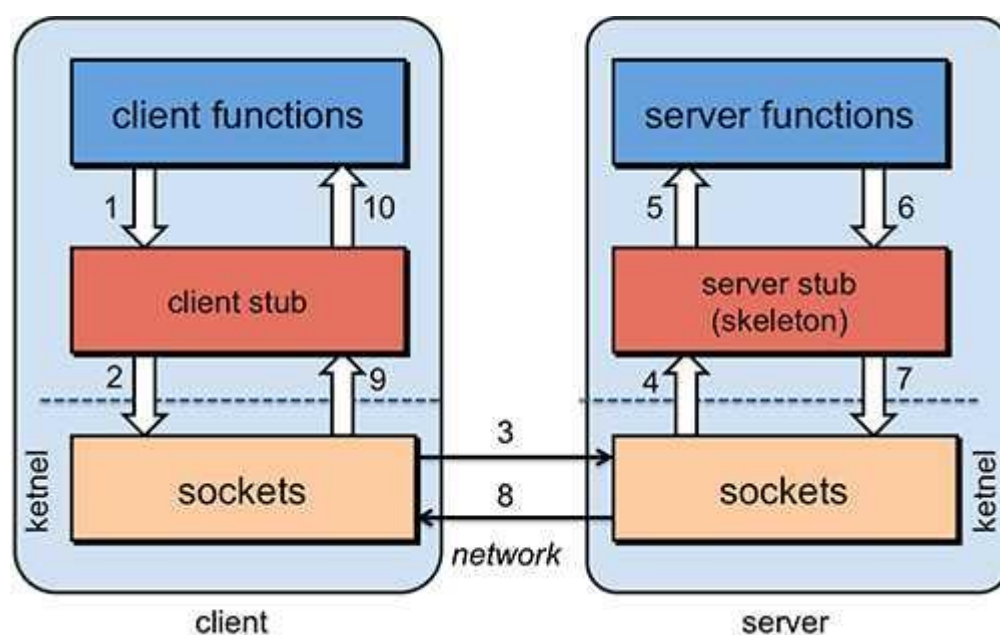
首先，要解决通讯的问题，主要是通过客户端和服务端之间建立 TCP 连接，远程过程调用的所有交换的数据都在这个连接里传输。连接可以是按需连接，调用结束后就断掉，也可以是长连接，多个远程过程调用共享同一个连接。

第二，要解决寻址的问题，也就是说，A 服务器上的应用怎么告诉底层的 RPC 框架，如何连接到 B 服务器（如主机或 IP 地址）以及特定的端口，方法的名称名称是什么，这样才能完成调用。比如基于 Web 服务协议栈的 RPC，就要提供一个 endpoint URI，或者是从 UDDI(一种目录服务，通过该目录服务进行服务注册与搜索)服务上查找。如果是 RMI 调用的话，还需要一个 RMI Registry 来注册服务的地址。

第三，当 A 服务器上的应用发起远程过程调用时，方法的参数需要通过底层的网络协议如 TCP 传递到 B 服务器，由于网络协议是基于二进制的，内存中的参数的值要序列化成二进制的形式，也就是序列化 (Serialize) 或编组 (marshal)，通过寻址和传输将序列化的二进制发送给 B 服务器。

第四，B 服务器收到请求后，需要对参数进行反序列化（序列化的逆操作），恢复为内存中的表达方式，然后找到对应的方法（寻址的一部分）进行本地调用，然后得到返回值。

第五，返回值还要发送回服务器 A 上的应用，也要经过序列化的方式发送，服务器 A 接到后，再反序列化，恢复为内存中的表达方式，交给 A 服务器上的应用



JAVAAE 里面的 stub 是为屏蔽客户调用远程主机上的对象，必须提供某种方式来模拟本地对象，这种本地对象称为存根(stub)，存根负责接收本地方法调用，并将它们委派给各自的具体实现对象

Skeleton：服务器的骨架

2.5、为什么需要 RPC

论复杂度，RPC 框架肯定是高于简单的 HTTP 接口的。但毋庸置疑，HTTP 接口由于受限于 HTTP 协议，需要带 HTTP 请求头，导致传输起来效率或者说安全性不如 RPC。

现在问题是，遇到怎样的瓶颈了才需要或者说更适合用 RPC（比如像阿里这么大的请求并发量，简单的 HTTP 肯定达不到预期），但问题是大家所在的公司，要有像阿里这么大的量是比较少的，甚至说 1/1000 的量可能都没有，那我们还需要使用 RPC 吗？

技术应该不是为了使用新技术而去使用，而应该是旧技术存在某些瓶颈，存在难以支撑或者扩展性越老越差等问题暴露出来之后，用新技术来进行解决。

那 RPC 最大的优点，或者说它相比简单的 HTTP 接口，它的优势、更适合它的业务场景是怎样呢？简单的 HTTP 又哪里不足，哪些场景明显不太适合呢？

http 接口是在接口不多、系统与系统交互较少的情况下，解决信息初期常使用的一种通信手段；优点就是简单、直接、开发方便。利用现成的 http 协议进行传输。但是如果是一个大型的网站，内部子系统较多、接口非常多的情况下，RPC 框架的好处就显示出来了，首先就是长链接，不必每次通信都要像 http 一样去 3 次握手什么的，减少了网络开销(这个问题在 http2.0 已经被解决不再算是问题了)；其次就是 **RPC 框架一般都有注册中心，有丰富的监控管理；发布、下线接口、动态扩展等，对调用方来说是无感知、统一化的操作。第三个来说就是安全性。最后就是流行的服务化架构、服务化治理，RPC 框架是一个强力的支撑。**

RPC 是一种概念，http 也是 RPC 实现的一种方式，用 http 交互其实就已经属于 RPC 了。

但是我们为什么要应用 RPC 层呢？

- a. 灵活部署
- b. 解耦

系统做大了，肯定是需要做微服务的。现在我们做电商就是这样，单独有一个订单系统，支付系统，商品系统，用户系统。都是分开部署，单独上线的。

RPC:远程过程调用。RPC 的核心并不在于使用什么协议。RPC 的目的是让你在本地调用远程的方法，而对你来说这个调用是透明的，你并不知道这个调用的方法是部署哪里。通过 RPC 能解耦服务，这才是使用 RPC 的真正目的。RPC 的原理主要用到了动态代理模式，至于 http 协议，只是传输协议而已。

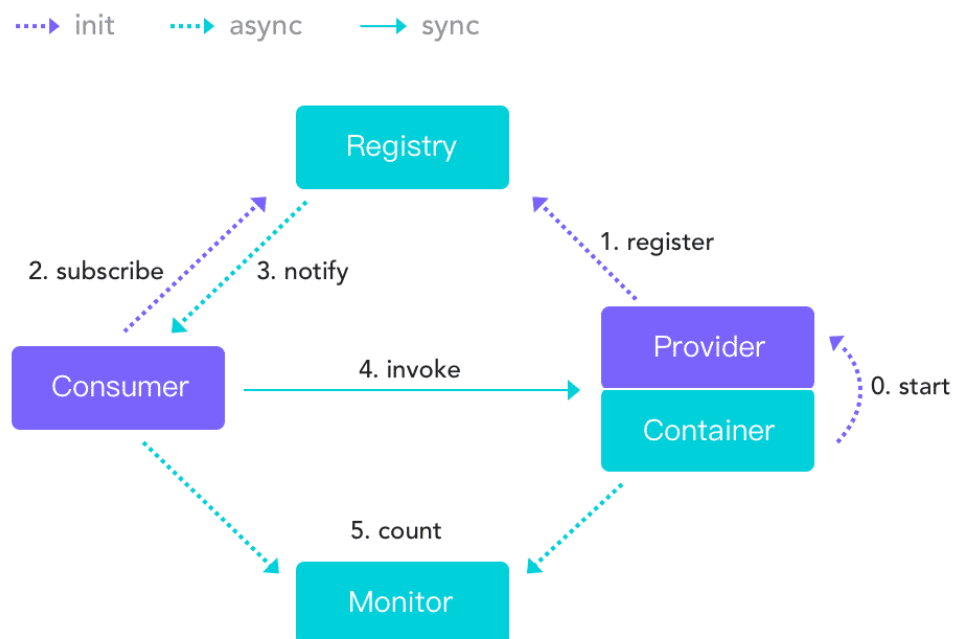
RPC 是一个软件结构概念，是构建分布式应用的理论基础。就好比为啥你家可以用到发电厂发出来的电？是因为电是可以传输的。至于用铜线还是用铁丝还是其他种类的导线，也就是用 http 还是用其他协议的问题了。这个要看什么场景，对性能要求怎么样。比如在 java 中的最基本的就是 RMI 技术，它是 java 原生的应用层分布式技术。我们可以肯定的是在传输性能方面，RMI 的性能是优于 HTTP 的。那为啥很少用到这个技术？那是因为用这个有很多局限性，首先它要保证传输的两端都要用 java 实现，且两边需要有相同的对象类型和代理接口，不需要容器，但是加大了编程的难度，在应用内部的各个子系统之间还是会看到他的身影，比如 EJB 就是基于 rmi 技术的。这就与目前的 bs 架构的软件大相径庭。用 http 必须要服务端位于 http 容器里面，这样减少了网络传输方面的开发，只需要关注业务开发即可。

3、Dubbo 架构

Dubbo 是由阿里巴巴开源的一个高性能、基于 Java 开源的远程调用框架。正如在许多 RPC 系统中一样，Dubbo 是基于定义服务的概念，指定可以通过参数和返回类型远程调用的方法。在服务器端，服务器实现这个接口，并运行一个 Dubbo 服务器来处理客户端调用。在客户端，客户机有一个存根，它提供与服务器相同的方法。

Dubbo 提供三个核心功能：基于接口的远程调用、容错和负载均衡，以及服务的自动注册与发现。Dubbo 框架广泛的在阿里巴巴内部使用，以及京东、当当、去哪儿、考拉等都在使用。

Dubbo Architecture



节点角色说明：

- **Provider**：暴露服务的服务提供方。
- **Consumer**：调用远程服务的服务消费方。
- **Registry**：服务注册与发现的注册中心。
- **Monitor**：统计服务的调用次调和调用时间的监控中心。
- **Container**：服务运行容器。

调用关系说明：

- 0. 服务容器负责启动，加载，运行服务提供者。
- 1. 服务提供者在启动时，向注册中心注册自己提供的服务。
- 2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

- 3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。
- 4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。
- 5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

4、Dubbo 快速入门

官方文档: <http://dubbo.apache.org/zh-cn/docs/user/quick-start.html>

4.1、依赖

jdk1.6 以上和 maven3.0 以上，采用 maven 分模块构建 api 模块，provider 模块以及 consumer 模块

4.2、Dubbo 坐标

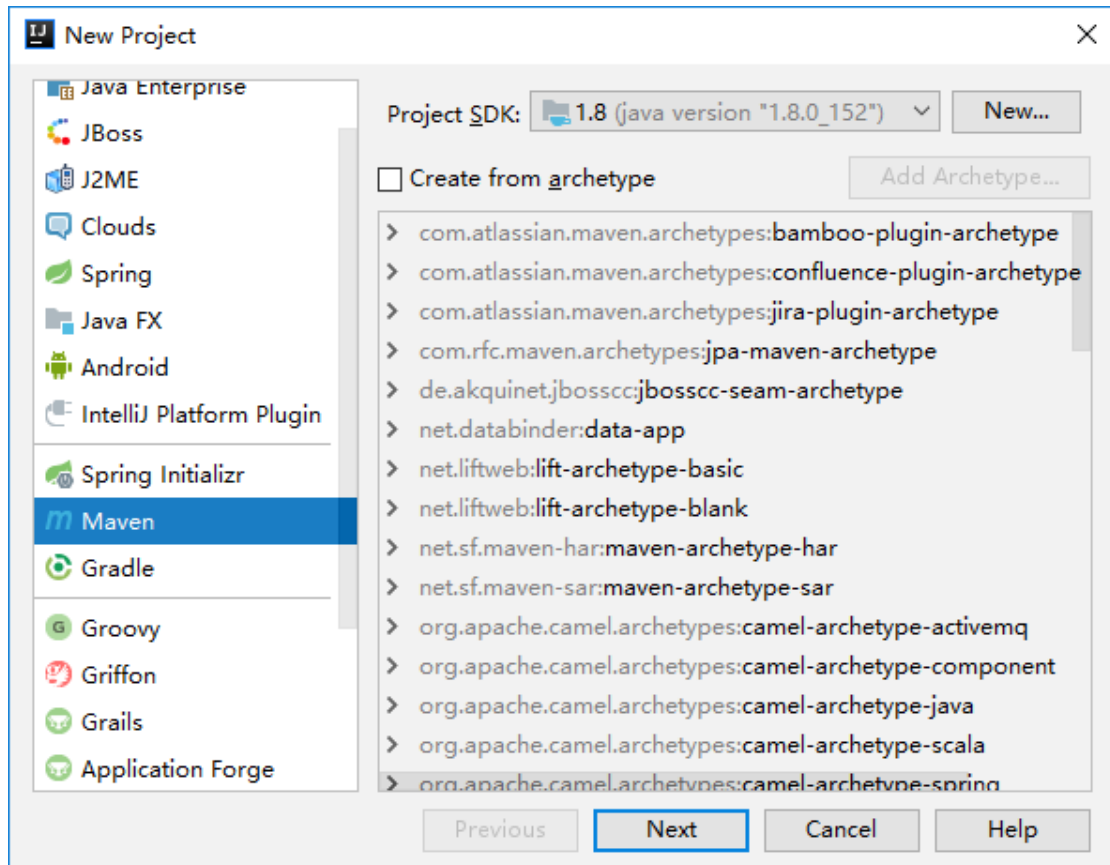
<https://mvnrepository.com/artifact/com.alibaba/dubbo>


dubbo-parent 的 pom.xml

```
<dependencies>
  <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>dubbo</artifactId>
    <version>2.5.6</version>
  </dependency>
</dependencies>
```

4.3、搭建 maven 多模块聚合项目

4.3.1、创建 dubbo-parent




 New Project ✕

GroupId: ☒ Inherit

ArtifactId:

Version: ☒ Inherit

 New Project ✕

Project name:

Project location:

pom.xml 依赖 dubbo


```

<?xml version="1.0" encoding="UTF-8"?>
<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.shsxt</groupId>
  <artifactId>dubbo-parent</artifactId>
  <packaging>pom</packaging>
  <version>1.0-SNAPSHOT</version>
  <modules>
    <module>dubbo-api</module>
    <module>dubbo-provider</module>
    <module>dubbo-consumer</module>
  </modules>

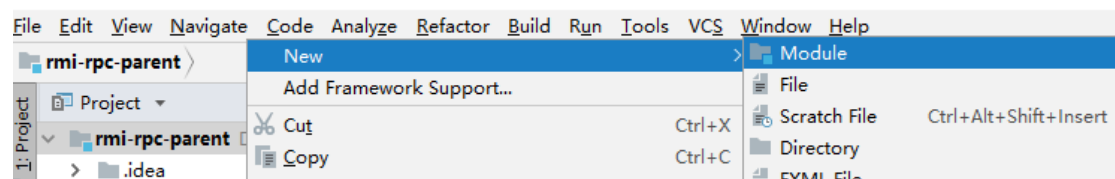
  <dependencies>
    <dependency>
      <groupId>com.alibaba</groupId>
      <artifactId>dubbo</artifactId>
      <version>2.5.6</version>
    </dependency>
  </dependencies>

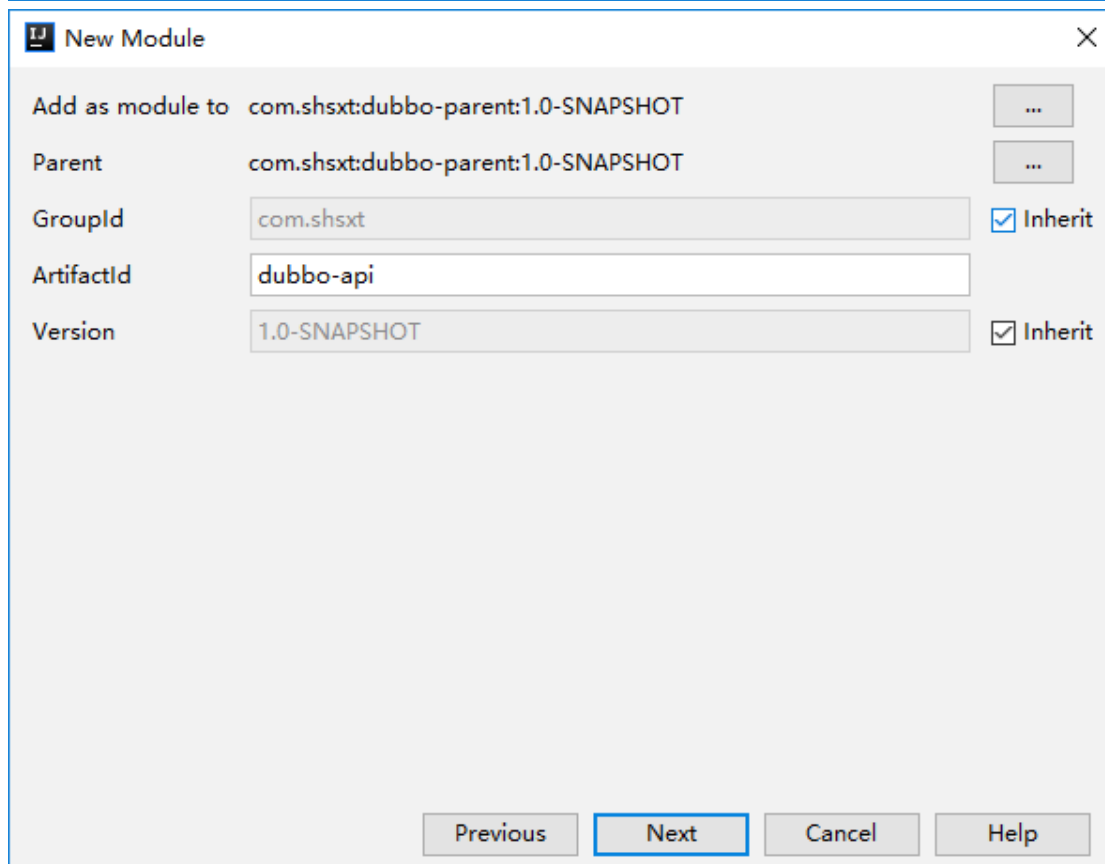
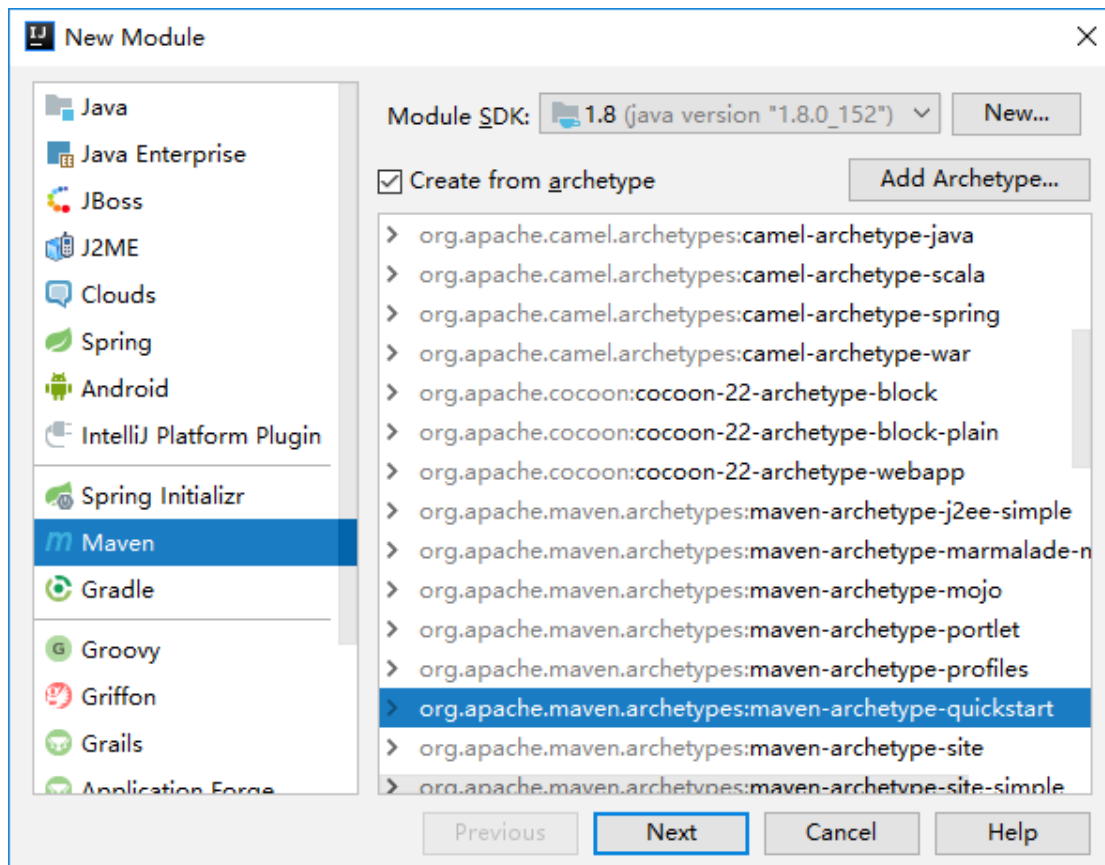
</project>

```

4.3.2、创建 dubbo-api

鼠标右键 dubbo-parent 项目 new -> Module





New Module [X]

Maven home directory: [v] [...]
(Version: 3.3.9)

User settings file: [...] ☒ Override

Local repository: [...] ☒ Override

Properties

groupId	com.shsxt	+ - ✎
artifactId	dubbo-api	
version	1.0-SNAPSHOT	
archetypeGroupId	org.apache.maven.archetypes	
archetypeArtifactId	maven-archetype-quickstart	
archetypeVersion	RELEASE	

New Module [X]

Module name:

Content root: [...]

Module file location: [...]

pom.xml

```

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

<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">
  <parent>
    <artifactId>dubbo-parent</artifactId>
    <groupId>com.shsxt</groupId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>

  <artifactId>dubbo-api</artifactId>

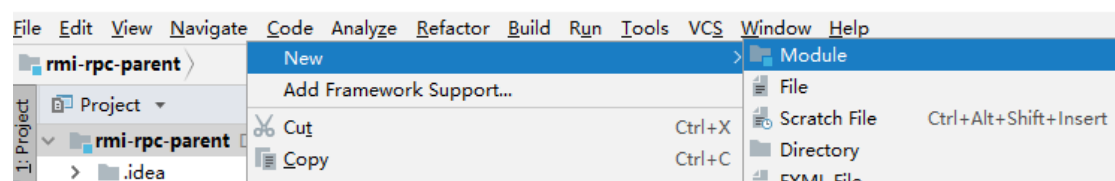
  <name>dubbo-api</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

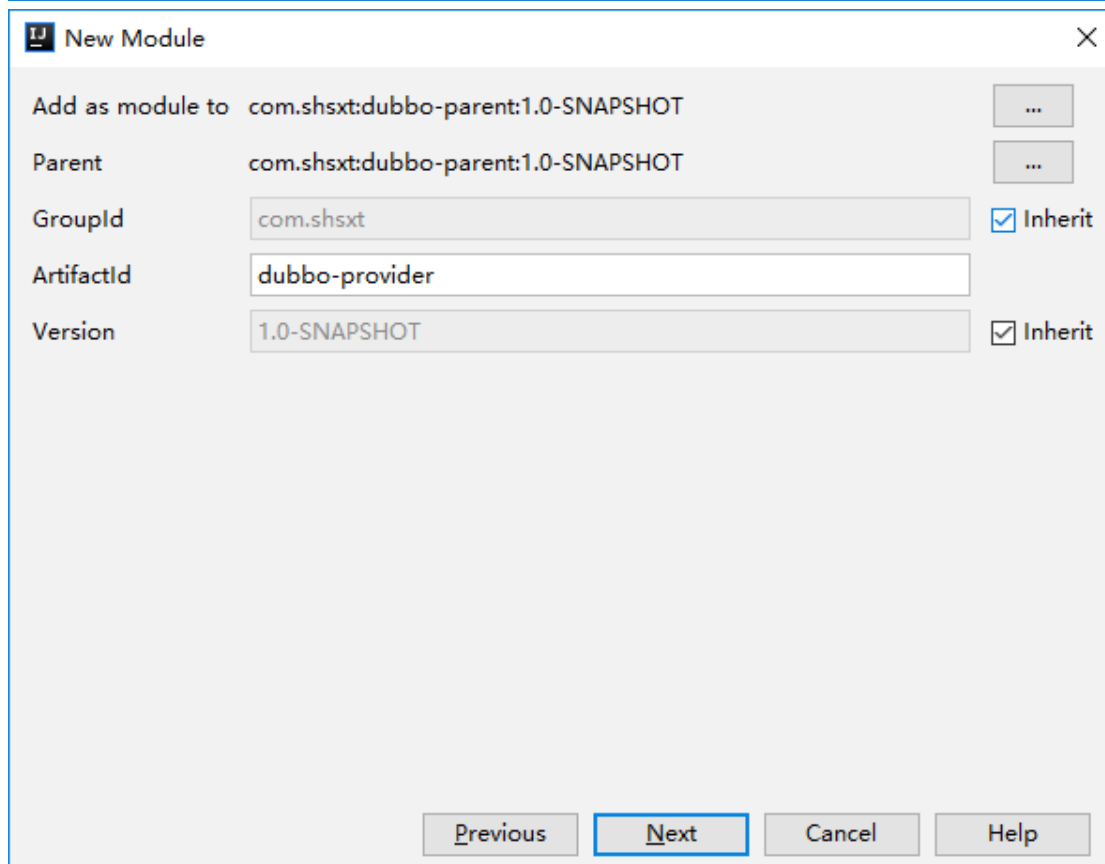
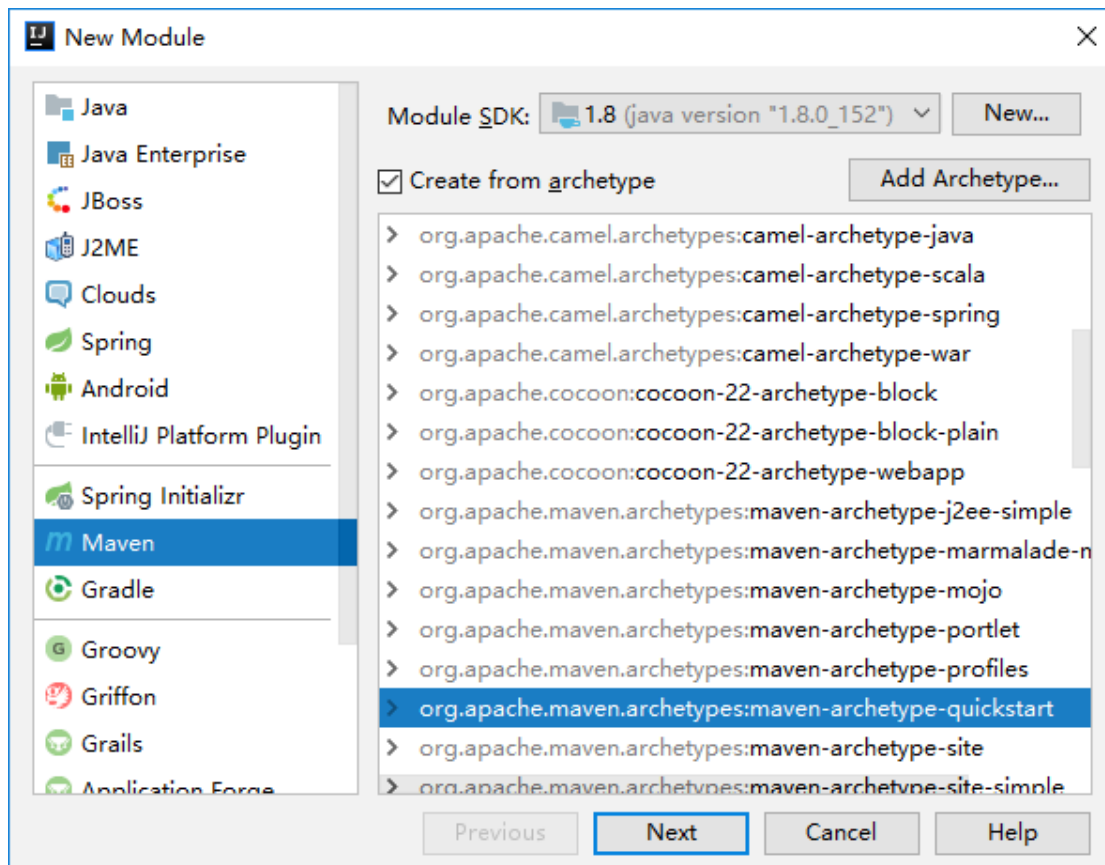
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
</project>

```

4.3.3、创建 dubbo-provider

鼠标右键 dubbo-parent 项目 new -> Module





New Module [X]

Maven home directory: D:/maven/apache-maven-3.3.9 [v] [...] (Version: 3.3.9)

User settings file: D:\maven\apache-maven-3.3.9\conf\settings.xml [...] ☒ Override

Local repository: D:\maven\repository [...] ☒ Override

Properties

groupId	com.shsxt	+
artifactId	dubbo-provider	-
version	1.0-SNAPSHOT	
archetypeGroupId	org.apache.maven.archetypes	
archetypeArtifactId	maven-archetype-quickstart	
archetypeVersion	RELEASE	

[Previous] [Next] [Cancel] [Help]

New Module [X]

Module name: dubbo-provider

Content root: D:\IdeaProjects\Dylan\dubbo-parent\dubbo-provider [...]

Module file location: D:\IdeaProjects\Dylan\dubbo-parent\dubbo-provider [...]

[Previous] [Finish] [Cancel] [Help]

pom.xml 依赖 dubbo-api

```

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

<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">
    <parent>
        <artifactId>dubbo-parent</artifactId>
        <groupId>com.shsxt</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>dubbo-provider</artifactId>

    <name>dubbo-provider</name>
    <!-- FIXME change it to the project's website -->
    <url>http://www.example.com</url>

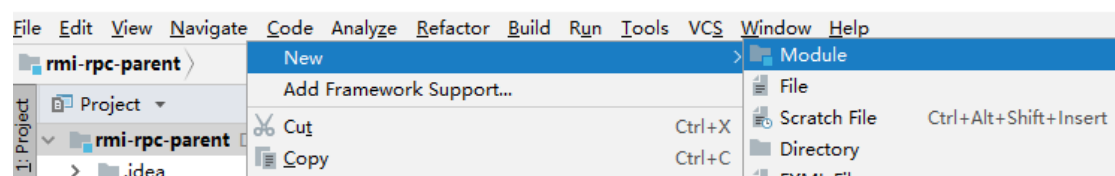
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

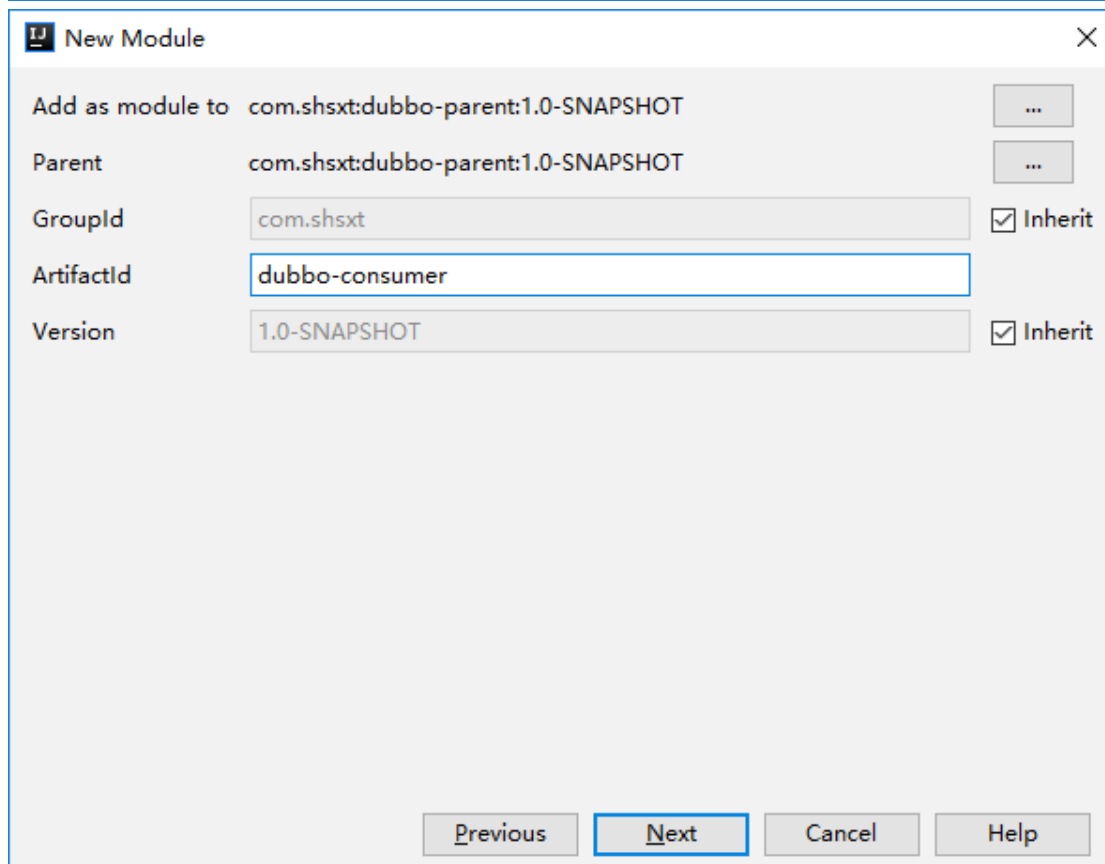
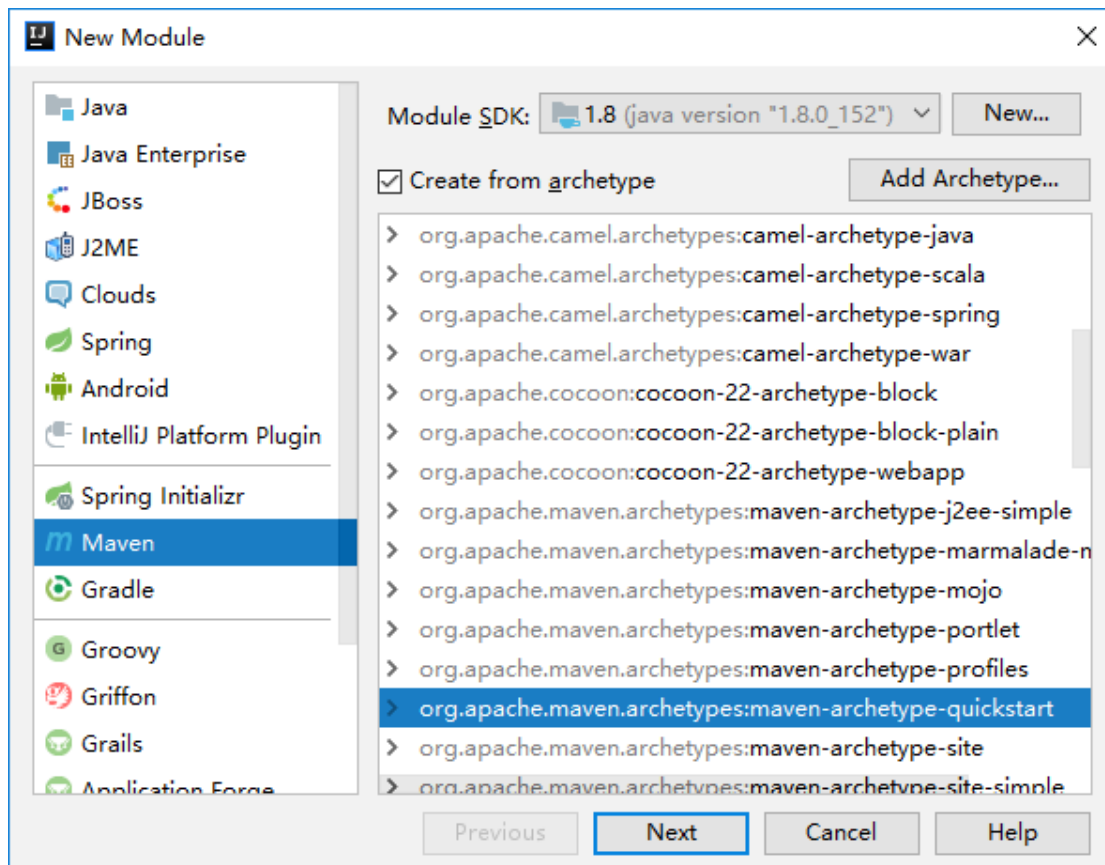
    <dependencies>
        <dependency>
            <groupId>com.shsxt</groupId>
            <artifactId>dubbo-api</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
</project>

```

4.3.4、创建 dubbo-consumer

鼠标右键 dubbo-parent 项目 new -> Module





New Module [X]

Maven home directory: D:/maven/apache-maven-3.3.9 [v] [...] (Version: 3.3.9)

User settings file: D:\maven\apache-maven-3.3.9\conf\settings.xml [...] ☒ Override

Local repository: D:\maven\repository [...] ☒ Override

Properties

groupId	com.shsxt	+
artifactId	dubbo-consumer	-
version	1.0-SNAPSHOT	
archetypeGroupId	org.apache.maven.archetypes	
archetypeArtifactId	maven-archetype-quickstart	
archetypeVersion	RELEASE	

[Previous] [Next] [Cancel] [Help]

New Module [X]

Module name: dubbo-consumer

Content root: D:\IdeaProjects\Dylan\dubbo-parent\dubbo-consumer [...]

Module file location: D:\IdeaProjects\Dylan\dubbo-parent\dubbo-consumer [...]

[Previous] [Finish] [Cancel] [Help]

pom.xml 依赖 dubbo-api

```

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

<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">
  <parent>
    <artifactId>dubbo-parent</artifactId>
    <groupId>com.shsxt</groupId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>

  <artifactId>dubbo-consumer</artifactId>

  <name>dubbo-consumer</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>com.shsxt</groupId>
      <artifactId>dubbo-api</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>
</project>

```

4.4、定义服务接口

由于服务的生产者和消费者都会依赖这个接口，因此强烈建议把这个接口定义在一个独立的模块，然后由生产者模块和消费者模块各自依赖即可。

dubbo-api 里 UserServiceI.java

```

package com.shsxt.service;

import com.shsxt.pojo.User;

```

```
/**
 * 服务接口
 */
public interface UserServiceI {

    User selectUserById(Integer userId);

}
```

dubbo-api 里 User.java

```
package com.shsxt.pojo;

import java.io.Serializable;

public class User implements Serializable {

    private static final long serialVersionUID = 4179691914534970793L;
    private Integer id;
    private String name;
    private String pwd;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPwd() {
        return pwd;
    }

    public void setPwd(String pwd) {
        this.pwd = pwd;
    }

}
```

```

@Override
public String toString() {
    return "User{" +
        "id=" + id +
        ", name='" + name + '\'' +
        ", pwd='" + pwd + '\'' +
        '}';
}
}

```

4.5、配置服务提供方

4.5.1、在 provider 模块中实现服务接口

dubbo-provider 里 UserServiceImpl.java

```

package com.shsxt.service;

import com.shsxt.pojo.User;

/**
 * 服务实现
 */
public class UserServiceImpl implements UserServiceI {

    @Override
    public User selectUserById(Integer userId) {
        User user = new User();
        user.setId(userId);
        user.setName("admin");
        user.setPwd("123456");
        System.out.println("userId:" + userId);
        return user;
    }

}

```

4.5.2、配置服务提供方

dubbo-provider 里 applicationContext-dubbo-provider.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://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">

    <!-- 提供方应用信息，用于计算依赖关系 -->
    <dubbo:application name="dubbo-provider"/>

    <!-- 使用multicast 广播注册中心暴露服务地址 -->
    <dubbo:registry address="multicast://224.5.6.7:1234"/>

    <!-- 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.shsxt.service.UserServiceI" ref="userService"/>

    <!-- 和本地 bean 一样实现服务 -->
    <bean id="userService" class="com.shsxt.service.UserServiceImpl"/>

</beans>

```

4.5.3、启动服务

dubbo-provider 里 Publish.java

```

package com.shsxt;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.io.IOException;

/**
 * 启动服务
 */
public class Publish {

    public static void main(String[] args) throws IOException {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext-dubbo-provider.xml");
        context.start();
        System.out.println("服务注册成功!");
    }
}

```

```
        System.in.read(); // 按任意键退出
    }
}
```

4.6、配置服务消费方

dubbo-consumer 里 applicationContext-dubbo-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://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">

    <!-- 消费方应用名，用于计算依赖关系，不是匹配条件，不要与提供方一样 -->
    <dubbo:application name="dubbo-consumer"/>

    <!-- 使用 multicast 广播注册中心暴露发现服务地址 -->
    <dubbo:registry address="multicast://224.5.6.7:1234"/>

    <!-- 生成远程服务代理，可以和本地 bean 一样使用 userService -->
    <dubbo:reference id="userService" interface="com.shsxt.service.UserServiceI"/>

</beans>
```

4.7、消费服务

这里我们使用 main 方法进行测试

dubbo-consumer 里 ConsumerTest.java

```
package com.shsxt.test;

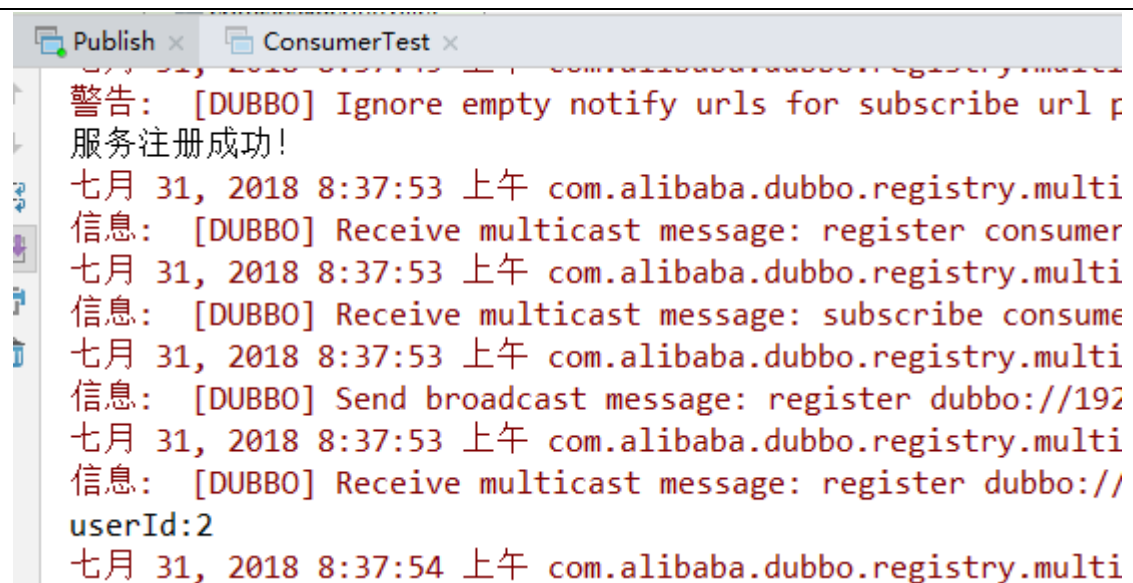
import com.shsxt.test.pojo.User;
import com.shsxt.test.service.UserServiceI;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * 测试服务消费
 */
public class ConsumerTest {
```

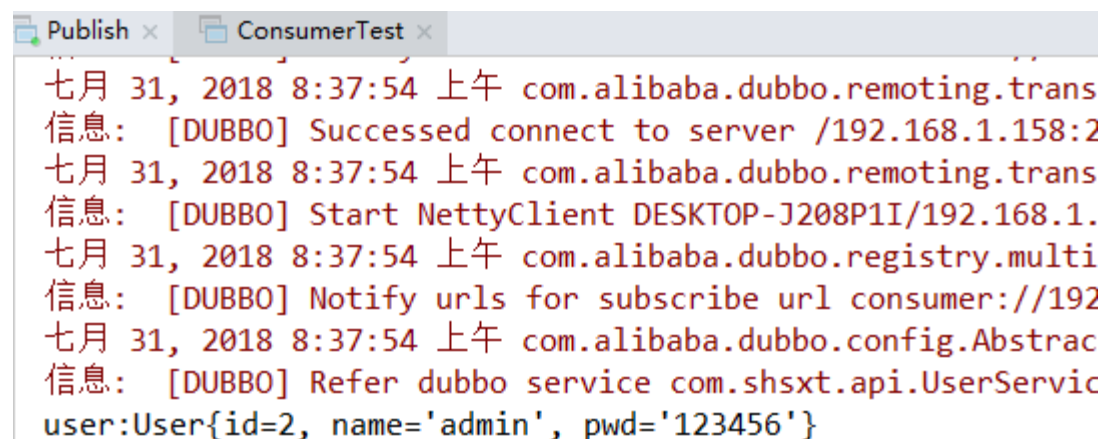
```

public static void main(String[] args) {
    ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext-dubbo-consumer.xml");
    context.start();
    // 获取远程服务代理
    UserServiceI userService = (UserServiceI) context.getBean("userService");
    // 执行远程方法
    User user = userService.selectUserById(2);
    // 显示调用结果
    System.out.println("user:" + user);
}
}

```



警告: [DUBBO] Ignore empty notify urls for subscribe url p
 服务注册成功!
 七月 31, 2018 8:37:53 上午 com.alibaba.dubbo.registry.multi
 信息: [DUBBO] Receive multicast message: register consumer
 七月 31, 2018 8:37:53 上午 com.alibaba.dubbo.registry.multi
 信息: [DUBBO] Receive multicast message: subscribe consume
 七月 31, 2018 8:37:53 上午 com.alibaba.dubbo.registry.multi
 信息: [DUBBO] Send broadcast message: register dubbo://192
 七月 31, 2018 8:37:53 上午 com.alibaba.dubbo.registry.multi
 信息: [DUBBO] Receive multicast message: register dubbo://
 userId:2
 七月 31, 2018 8:37:54 上午 com.alibaba.dubbo.registry.multi



七月 31, 2018 8:37:54 上午 com.alibaba.dubbo.remoting.trans
 信息: [DUBBO] Succeeded connect to server /192.168.1.158:2
 七月 31, 2018 8:37:54 上午 com.alibaba.dubbo.remoting.trans
 信息: [DUBBO] Start NettyClient DESKTOP-J208P1I/192.168.1.
 七月 31, 2018 8:37:54 上午 com.alibaba.dubbo.registry.multi
 信息: [DUBBO] Notify urls for subscribe url consumer://192
 七月 31, 2018 8:37:54 上午 com.alibaba.dubbo.config.Abstrac
 信息: [DUBBO] Refer dubbo service com.shsxt.api.UserService
 user:User{id=2, name='admin', pwd='123456'}

后期我们在项目中注入 UserServiceI 和调用本地方法一样调用远程接口即可。
 dubbo-consumer 里 UserController.java

```

package com.shsxt.controller;

import com.shsxt.service.UserServiceI;

```

```
import com.shsxt.pojo.User;

public class UserController {

    private UserServiceI userService;

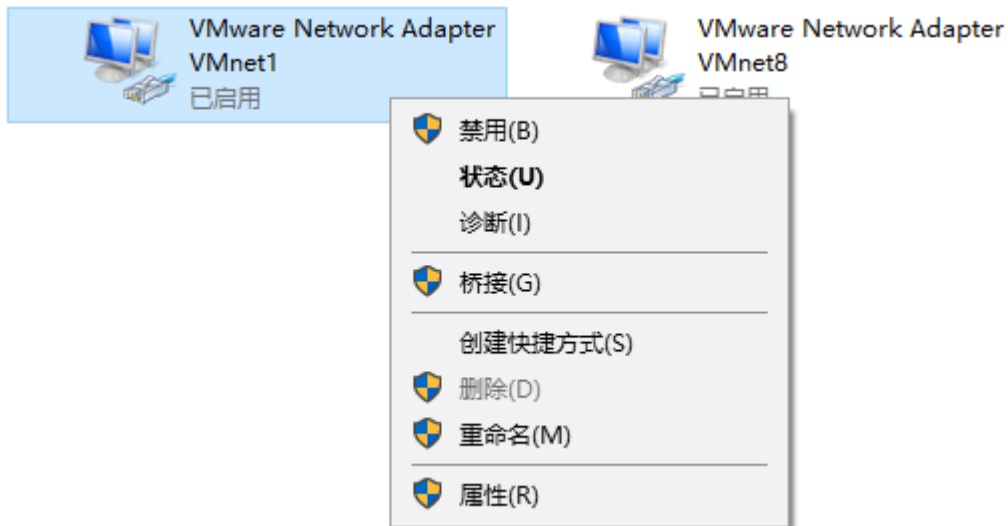
    public User selectUserById(Integer userId) {
        return userService.selectUserById(userId);
    }

}
```

如果报错，服务消费失败：

禁用虚拟机网络适配器，测试成功以后重新启动即可。

VMnet1 和 VMnet8 都需要禁用，测试成功以后再重启即可。



5、Dubbo 常用标签

dubbo:application 指定应用程序名称
 dubbo:registry 指定连接注册中心信息(配置注册中心)
 dubbo:protocol 服务提供方注册服务采用的协议
 dubbo:service 对外暴露服务配置
 dubbo:reference 配置订阅的服务(接口的代理)
 其他标签及属性用到了去 api 查询即可。

6、Dubbo 中 RPC 协议配置

Dubbo 框架采用 RPC 协议来对外暴露自己所提供的服务。

Dubbo 协议

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

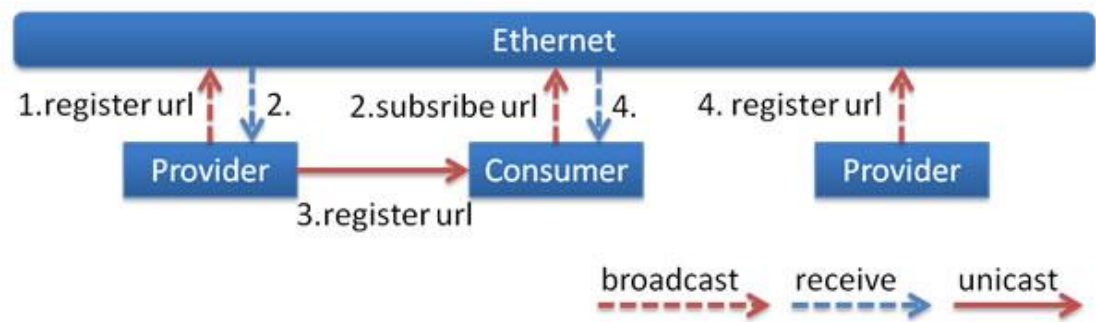
注册中心

注册中心的作用：就是更高效的管理系统的服务：比如服务接口的发布、自动剔除无效的服务、自动恢复服务等

Dubbo 中支持四种注册中心：multicast、zookeeper(推荐)、redis、simple

Multicast 注册中心

Multicast 注册中心不需要启动任何中心节点，只要广播地址一样，就可以互相发现。



- 1. 提供方启动时广播自己的地址
- 2. 消费方启动时广播订阅请求
- 3. 提供方收到订阅请求时，单播自己的地址给订阅者，如果设置了 unicast=false，则广播给订阅者
- 4. 消费方收到提供方地址时，连接该地址进行 RPC 调用。
- 5. 组播受网络结构限制，只适合小规模应用或开发阶段使用。组播地址段: 224.0.0.0 - 239.255.255.255

配置

```
<dubbo:registry address="multicast://224.5.6.7:1234" />
<dubbo:registry protocol="multicast" address="224.5.6.7:1234" />
```

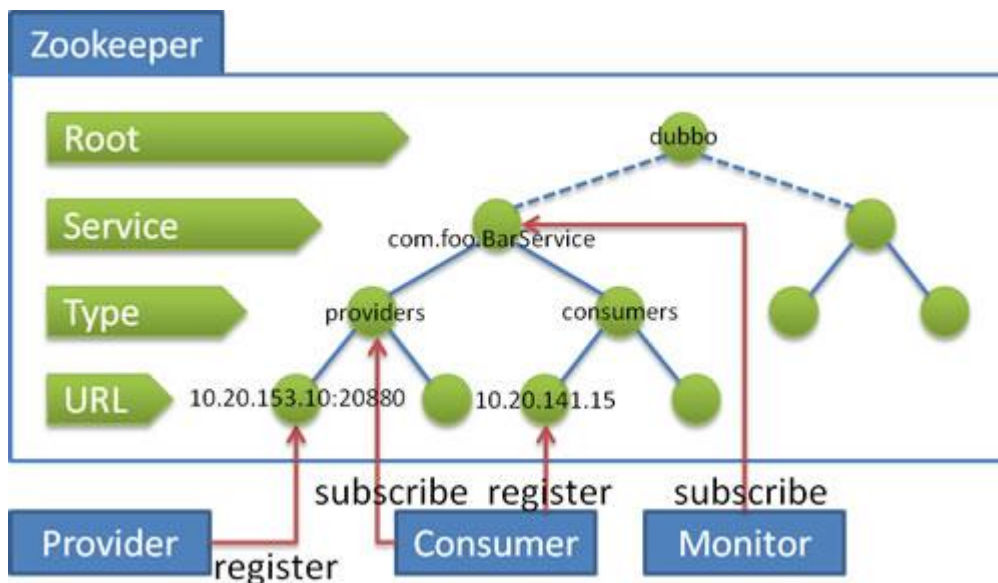
为了减少广播量，Dubbo 缺省使用单播发送提供者地址信息给消费者，如果一个机器上同时启了多个消费者进程，消费者需声明 unicast=false，否则只会有一个消费者能收到消息：

```
<dubbo:registry address="multicast://224.5.6.7:1234?unicast=false" />
<dubbo:registry protocol="multicast" address="224.5.6.7:1234">
```

```
<dubbo:parameter key="unicast" value="false" />
</dubbo:registry>
```

zookeeper 注册中心

Zookeeper 是 Apache Hadoop 的子项目，是一个树型的目录服务，支持变更推送，适合作为 Dubbo 服务的注册中心，工业强度较高，可用于生产环境，并推荐使用。



流程说明：

- 服务提供者启动时：向 `/dubbo/com.foo.BarService/providers` 目录下写入自己的 URL 地址
- 服务消费者启动时：订阅 `/dubbo/com.foo.BarService/providers` 目录下的提供者 URL 地址。并向 `/dubbo/com.foo.BarService/consumers` 目录下写入自己的 URL 地址
- 监控中心启动时：订阅 `/dubbo/com.foo.BarService` 目录下的所有提供者和消费者 URL 地址。

支持以下功能：

- 当提供者出现断电等异常停机时，注册中心能自动删除提供者信息
- 当注册中心重启时，能自动恢复注册数据，以及订阅请求
- 当会话过期时，能自动恢复注册数据，以及订阅请求
- 当设置 `<dubbo:registry check="false" />` 时，记录失败注册和订阅请求，后台定时重试
- 可通过 `<dubbo:registry username="admin" password="1234" />` 设置 zookeeper 登录信息
- 可通过 `<dubbo:registry group="dubbo" />` 设置 zookeeper 的根节点，不设置将使用无根树
- 支持 * 号通配符 `<dubbo:reference group="*" version="*" />`，可订阅服务的所有分组和所有版本的提供者

使用

在 provider 和 consumer 中增加 zookeeper 客户端 jar 包依赖:

```
<dependency>
  <groupId>org.apache.zookeeper</groupId>
  <artifactId>zookeeper</artifactId>
  <version>3.3.3</version>
</dependency>
```

或直接[下载](#)。

Dubbo 支持 zkclient 和 curator 两种 Zookeeper 客户端实现:

使用 zkclient 客户端

缺省配置:

```
<dubbo:registry ... client="zkclient" />
```

或:

```
dubbo.registry.client=zkclient
```

或:

```
zookeeper://10.20.153.10:2181?client=zkclient
```

需依赖或直接[下载](#):

```
<dependency>
  <groupId>com.github.sgroschupf</groupId>
  <artifactId>zkclient</artifactId>
  <version>0.1</version>
</dependency>
```

使用 curator 客户端

从 2.3.0 版本开始支持可选 curator 实现。[Curator](#) 是 Netflix 开源的一个 Zookeeper 客户端实现。

如果需要改为 curator 实现, 请配置:

```
<dubbo:registry ... client="curator" />
```

或:

```
dubbo.registry.client=curator
```

或:

```
zookeeper://10.20.153.10:2181?client=curator
```

需依赖或直接[下载](#):

```
<dependency>
  <groupId>com.netflix.curator</groupId>
  <artifactId>curator-framework</artifactId>
  <version>1.1.10</version>
</dependency>
```

Zookeeper 单机配置:

```
<dubbo:registry address="zookeeper://10.20.153.10:2181" />
```

或:

```
<dubbo:registry protocol="zookeeper" address="10.20.153.10:2181" />
```

Zookeeper 集群配置:

```
<dubbo:registry
address="zookeeper://10.20.153.10:2181?backup=10.20.153.11:2181,10.20.153.12:2181" />
```

或:

```
<dubbo:registry protocol="zookeeper"
address="10.20.153.10:2181,10.20.153.11:2181,10.20.153.12:2181" />
```

同一 Zookeeper，分成多组注册中心:

```
<dubbo:registry id="chinaRegistry" protocol="zookeeper"
address="10.20.153.10:2181" group="china" />
<dubbo:registry id="intlRegistry" protocol="zookeeper"
address="10.20.153.10:2181" group="intl" />
```

windows 连接 linux zookeeper 集群注册中心会引起连接超时的问题，课程中我们使用 windows 环境开发。

7、监控中心安装

7.1、安装环境

jdk1.8、zookeeper3.4.13、contos7.3-1611

7.2、上传监控中心安装包至服务器

```
[root@localhost ~]# ll
总用量 256812
-rw-----. 1 root root      1293 12月 20 18:14 anaconda-ks.cfg
-rw-r--r--. 1 root root    9653382 1月 19 14:11 apache-tomcat-8.5.37.tar.gz
-rw-r--r--. 1 root root   19092672 2月  1 18:16 dubbo-monitor-simple-2.5.3-assembly.tar.gz
-rw-r--r--. 1 root root  194042837 1月 18 16:10 jdk-8u202-linux-x64.tar.gz
drwxr-xr-x. 9 1001 1001      186 1月 19 14:35 nginx-1.14.2
-rw-r--r--. 1 root root    1015384 1月 19 14:33 nginx-1.14.2.tar.gz
drwxrwxr-x. 6 root root      4096 12月 12 20:25 redis-5.0.3
-rw-r--r--. 1 root root    1959445 2月  1 11:26 redis-5.0.3.tar.gz
-rw-r--r--. 1 root root    37191810 2月  1 17:05 zookeeper-3.4.13.tar.gz
```

7.3、创建安装目录，并解压至安装目录

创建安装目录

```
mkdir -p /usr/local/dubbo
```

解压

```
tar zxvf dubbo-monitor-simple-2.5.3-assembly.tar.gz -C /usr/local/dubbo/
```

7.4、修改配置文件

指定注册地址-重点

配置 hosts 文件

```
vi /etc/hosts
```

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.177.111 zk1
192.168.177.112 zk2
192.168.177.113 zk3
```

```
cd /usr/local/dubbo/dubbo-monitor-simple-2.5.3/conf
```

```
vim dubbo.properties
```

```

1 ##
2 # Copyright 1999-2011 Alibaba Group.
3 #
4 # Licensed under the Apache License, Version 2.0 (the "License");
5 # you may not use this file except in compliance with the License.
6 # You may obtain a copy of the License at
7 #
8 #     http://www.apache.org/licenses/LICENSE-2.0
9 #
10 # Unless required by applicable law or agreed to in writing, software
11 # distributed under the License is distributed on an "AS IS" BASIS,
12 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 # See the License for the specific language governing permissions and
14 # limitations under the License.
15 ##
16 dubbo.container=log4j,spring,registry,jetty
17 dubbo.application.name=simple-monitor
18 dubbo.application.owner=
19 #dubbo.registry.address=multicast://224.5.6.7:1234
20 dubbo.registry.address=zookeeper://zk1:2181?backup=zk2:2182,zk3:2183
21 #dubbo.registry.address=redis://127.0.0.1:6379
22 #dubbo.registry.address=dubbo://127.0.0.1:9090
23 dubbo.protocol.port=7070
24 dubbo.jetty.port=8080
25 dubbo.jetty.directory=${user.home}/monitor
26 dubbo.charts.directory=${dubbo.jetty.directory}/charts
27 dubbo.statistics.directory=${user.home}/monitor/statistics
28 dubbo.log4j.file=logs/dubbo-monitor-simple.log
29 dubbo.log4j.level=WARN

```

7.5、启动

启动监控中心前必须先启动 zookeeper 注册中心

```

cd /usr/local/dubbo/dubbo-monitor-simple-2.5.3/bin
./start.sh

```

```

[root@localhost bin]# pwd
/usr/local/dubbo/dubbo-monitor-simple-2.5.3/bin
[root@localhost bin]# ll
总用量 24
-rwxr-xr-x. 1 root root 2144 10月 23 2012 dump.sh
-rwxr-xr-x. 1 root root 49 10月 23 2012 restart.sh
-rwxr-xr-x. 1 root root 413 10月 23 2012 server.sh
-rwxr-xr-x. 1 root root 815 10月 23 2012 start.bat
-rwxr-xr-x. 1 root root 2963 10月 23 2012 start.sh
-rwxr-xr-x. 1 root root 836 10月 23 2012 stop.sh
[root@localhost bin]# ./start.sh
Starting the simple-monitor .....OK!
PID: 3078
STDOUT: logs/stdout.log

```

7.6、修改防火墙，访问

编辑防火墙规则

```
vim /etc/sysconfig/iptables
```

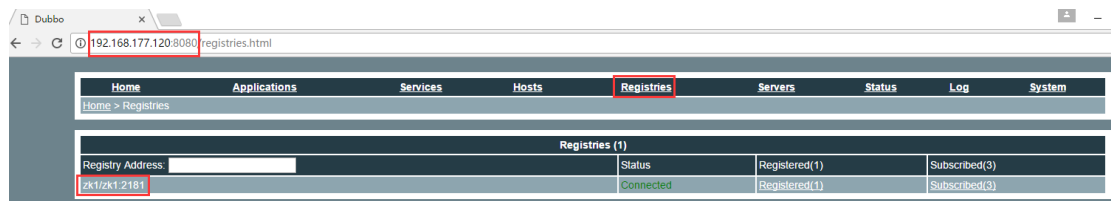
Dubbo 的默认端口为：8080，添加 dubbo 的 8080 端口至防火墙

```
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
```

启动防火墙

```
systemctl start/restart iptables.service 或者 service iptables start 或者 service iptables restart
```

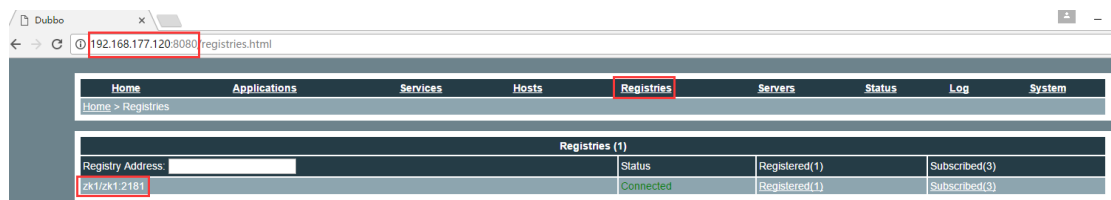
访问



或者关闭防火墙（CentOS7 中默认使用 firewall 作为防火墙）

```
systemctl stop firewalld.service | systemctl stop iptables.service
```

访问



windows 连接 linux zookeeper 集群注册中心会引起连接超时的问题，课程中我们使用 windows 环境开发。