



北京动力节点教育科技有限公司

# 动力节点课程讲义

DONGLIJIEDIANKECHENGJIANGYI

[www.bjpowernode.com](http://www.bjpowernode.com)

## 第1章 远程调用

### 1.1 访问网络上的接口

#### 1.1.1 互联网服务

天气接口：

中国天气网地址：<http://www.weather.com.cn>

请求地址：<http://www.weather.com.cn/data/sk/101110101.html>

101010100=北京

101020100=上海

101210101=杭州

京东万象：<https://wx.jcloud.com/api>

### 1.2 公司内部提供的服务

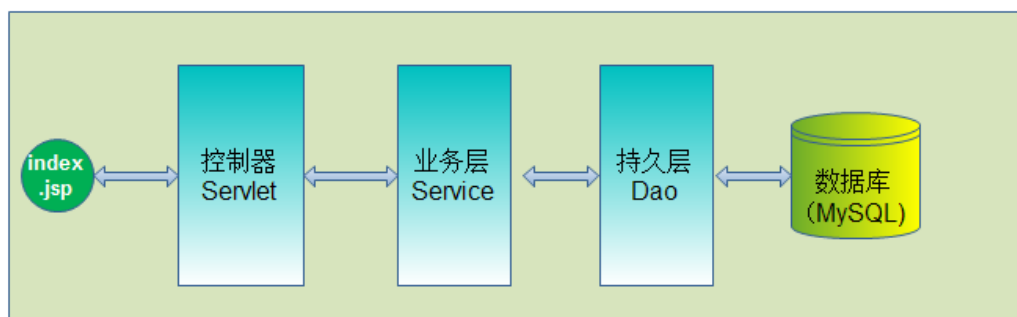
一家对外提供服务的公司，例如百度，腾讯，阿里，京东，58 同城等，公司内部有多个事业群，事业部门，每个事业部门内部又有若干个子部门，子部门里面有多个不同的小组负责各自的业务。提供对外的服务。

公司内部，外部提供的服务不仅多，而且细分，还有交叉的情况。

前面的例子是访问互联网上的服务，使用的是 **http** 请求网络资源。相对来说访问服务方式单一，处理服务的效率相对较低。公司内部服务之间可以使用多种不同的方式访问服务。

#### 1.2.1 使用单一应用访问天气服务

图一：



在一台机器上，运行单一应用，  
处理能力是有限的

图二:



### A、新建 web 项目 01-weathService

项目结构:

```
src
├── com.bjpowernode.action
│   └── QueryWeatherServlet.java
├── com.bjpowernode.beans
│   └── Weather.java
└── com.bjpowernode.service
    ├── GetWeathService.java
    └── GetWeathServiceImpl.java
```

### B、新建数据类 Weather

```
public class Weather {
    //城市
    private String city;
    //天气的描述
    private String desc;
    //湿度
    private String sd;
    //温度
    private int temp;
    //set ,get
}
```

重写 toString()

```
@Override  
public String toString() {  
    return "desc:"+desc+",city:"+city+",temp:"+temp+",sd:"+sd;  
}
```

### C、定义 Service 接口

```
public interface GetWeathService {  
  
    Weather queryWeatcher(String cityCode);  
}
```

### D、定义 Service 接口的实现类

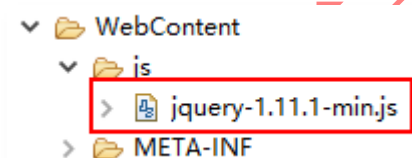
```
public class GetWeathServiceImpl implements GetWeathService {  
  
    @Override  
    public Weather queryWeatcher(String cityCode) {  
        Weather wea = new Weather();  
        if(cityCode.equals("101010100")){  
            System.out.println("101010100");  
            wea.setCity("北京");  
            wea.setDesc("晴好");  
            wea.setSd("20%");  
            wea.setTemp(22);  
        } else if(cityCode.equals("101020100")){  
            System.out.println("101020100");  
            wea.setCity("上海");  
            wea.setDesc("晴好");  
            wea.setSd("20%");  
            wea.setTemp(22);  
        }  
        return wea;  
    }  
}
```

## E、定义 Servlet，提供访问地址

```
@WebServlet("/qwsweather")
public class QueryWeatherServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public QueryWeatherServlet() {
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String cityCode = request.getParameter("cityCode");
        GetWeathService gs = new GetWeathServiceImpl();
        System.out.println("servlet:citycode:"+cityCode);
        Weather wea = gs.queryWeatcher(cityCode);
        System.out.println("wea:"+wea);
        response.setCharacterEncoding("utf-8");
        response.getWriter().write(wea.toString());
        response.getWriter().flush();
        response.getWriter().close();
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}
```

## F、定义访问添加服务的 jsp

首先加入 jQuery 库文件,放到项目的 js 目录下



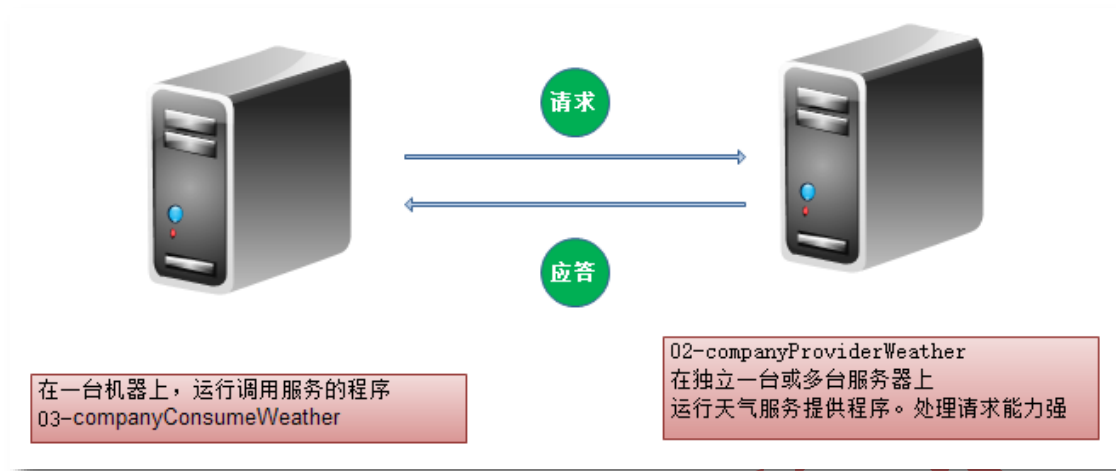
index.jsp

```
<script type="text/javascript" src="js/jquery-1.11.1-min.js"></script>
<script type="text/javascript">
    $(function(){
        $("button").click(function(){
            $.ajax({
                url:"${pageContext.request.contextPath}/qwsweather",
                data:{
                    cityCode:"101010100"
                },
                success:function(data){
                    alert(data);
                }
            })
        })
    })
</script>
<title>index.jsp</title>
</head>
<body>
    <button>显示天气</button>
</body>
```

## G、执行 web 应用



## 1.2.2 使用独立应用提供天气服务



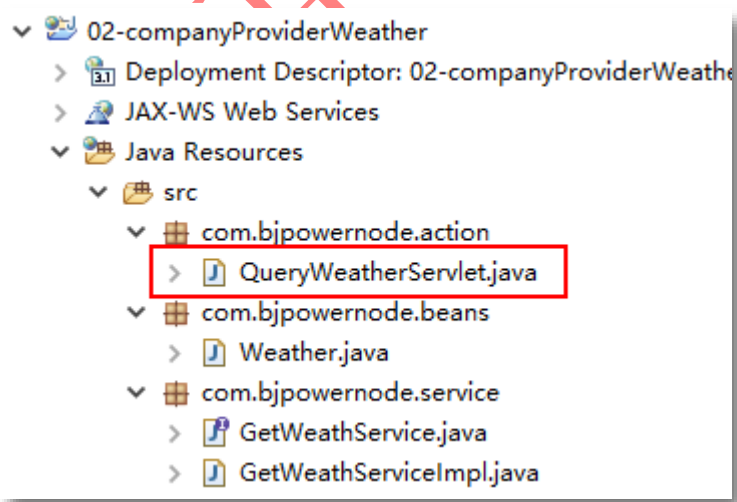
### (1) 独立的应用提供服务

在一台或多台物理机器上，运行的独立应用程序，供多个客户端访问天气服务。

A、把 01-weatherService 应用复制，名称 02-companyProviderWeather

B、去掉 js 文件夹，index.jsp 文件

C、使用 Servlet 提供服务



## (2) 在独立的应用中访问天气服务

在一台独立的计算上，通过应用访问天气服务。

A、把 01-weatherService 应用复制，名称 03-companyConsumeWeather

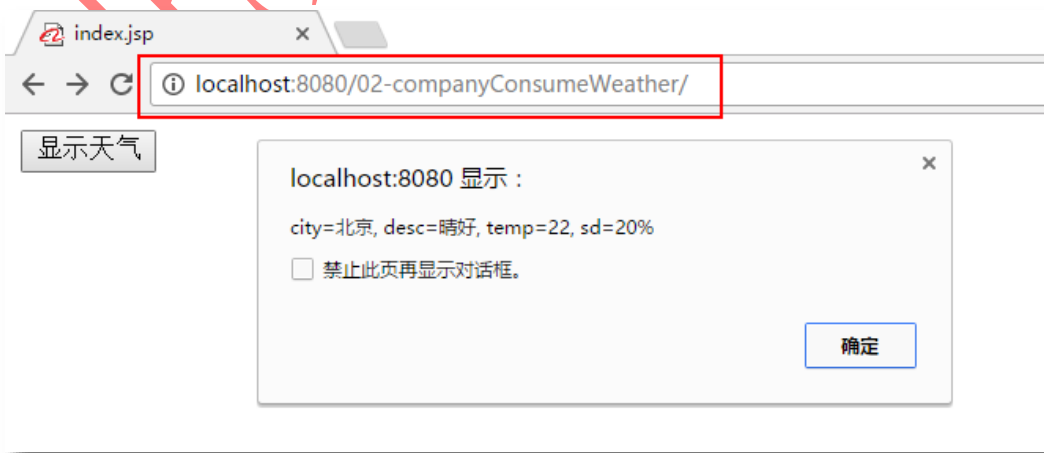
B、去掉 src 目录下的 java 代码

C、修改 index.jsp 中的访问服务 Servlet 的地址

```
<script type="text/javascript">
    $(function(){
        $("button").click(function(){
            $.ajax({
                url:"http://localhost:8080/02-companyConsumeWeather/qwsweather",
                data:{
                    cityCode:"101010100"
                },
                success:function(data){
                    alert(data);
                }
            })
        })
    })
</script>
```

D、运行应用

发布两个应用到 tomcat 服务器。03-companyConsumeWeather 应用访问 02-companyProviderWeather 提供的服务。两个应用是独立部署到不同的机器，使用两个或多个计算提供计算能力。





## 1.3 远程调用

### 1.3.1 远程调用涉及的概念

#### (1) 协议

协议指多方共同遵循的规范，在网络中的计算机进行数据交换依靠各种协议。例如 http，ftp 等。

一台计算机按规定好的格式发送数据，另一台计算的程序按指定的格式接收数据，两台计算使用互相理解的格式读写数据。达到数据交换的目的。

#### (2) RPC-远程过程调用协议

##### RPC 是什么？

PRC 是 Remote Procedure Call Protocol，称为：远程过程调用协议。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。该协议允许运行于一台计算机的程序调用另一台计算机的程序。程序员无需编为网络交互功能编码。

##### RPC 的作用？

主要功能是让构建分布式计算（应用）更容易，在提供强大的远程调用能力时不损失本地调用的语义简洁性。在一台计算的程序使用其他计算机上的功能就是使用自己的功能一样。RPC 技术提供了透明的访问其他服务的底层实现细节。使用分布式系统中的服务更加方便。

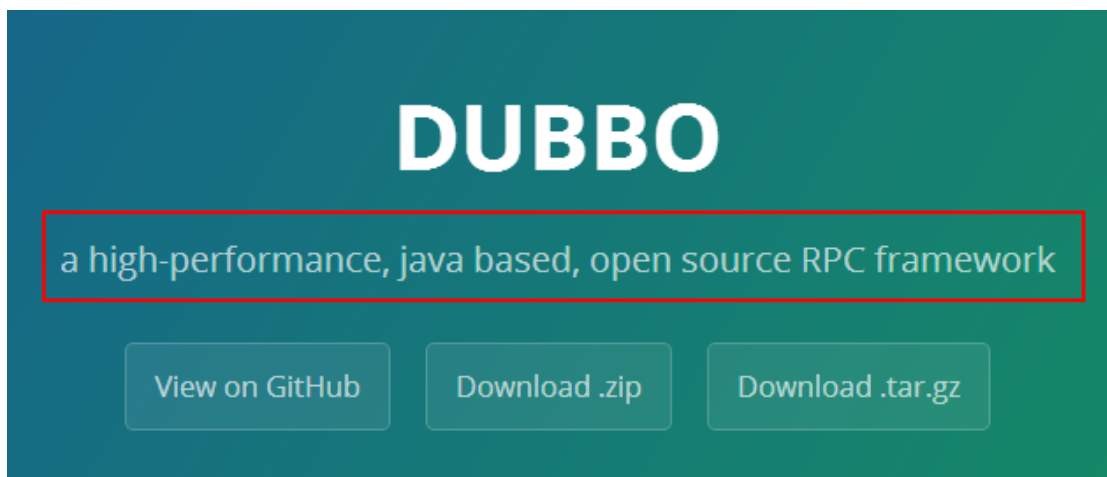
##### 分布式？

分布式指多台计算机位于网络系统中，多台计算给形成一个整体对外界提供服务。用户使用系统不知道是多台计算机，使用不同的操作系统，不同的应用程序提供服务。

## 第2章 Dubbo

### 2.1 Dubbo 介绍

官网:<http://dubbo.apache.org/>



红色框文字翻译后：一个高性能的，基于 java 的，开源 RPC 框架。

Dubbo 是一个框架

Dubbo 是一个分布式服务框架，致力于提供高性能和透明化的 **RPC** 远程服务调用方案、服务治理方案。

Dubbo 是阿里巴巴服务化治理方案的核心框架，每天为 2,000+ 个服务提供 3,000,000,000+ 次访问量支持，并被广泛应用于阿里巴巴集团的各成员站点：



已知的使用企业：



### 2.1.1 其他的 RPC 框架

- 新浪微博的 Motan (<https://github.com/weibocom/motan>)
  - Dubbox 是由当当对阿里的 Dubbo 的升级，可以被视为 Dubbo 的增强版 (<https://github.com/dangdangdotcom/dubbox>)
  - 国外的有 google grpc
- 前两个是基于 java 语言的和 spring 集成在一起的，grpc 是跨语言的，使用面更广。

### 2.1.2 Dubbo 能做什么

- 实现透明的远程方法调用，就像调用本地方法一样。可以忽略远程调用的实现细节。简单配置即可使用。
- 服务的自动注册和服务发现。通过注册中心，服务实现动态管理（增减服务方）。调用服务的消费者无需写死调用地址。

## 2.2 Dubbo 支持的协议

支持 8 种协议：dubbo , hessian , rmi , http, webservice , thrift , memcached , redis。  
dubbu 官方推荐使用 dubbo 协议。dubbo 协议默认端口 20880

### 2.2.1 Dubbo 协议

#### A、Dubbo 协议特点

Dubbo 协议采用单一长连接和异步通讯，适合于小数据量大并发的服务调用，以及服务消费

者机器数远大于服务提供者机器数的情况。

## B、网络通信

Dubbo 协议底层网络通信默认使用的是 netty，性能非常优秀，官方推荐使用

## C、不适合的地方

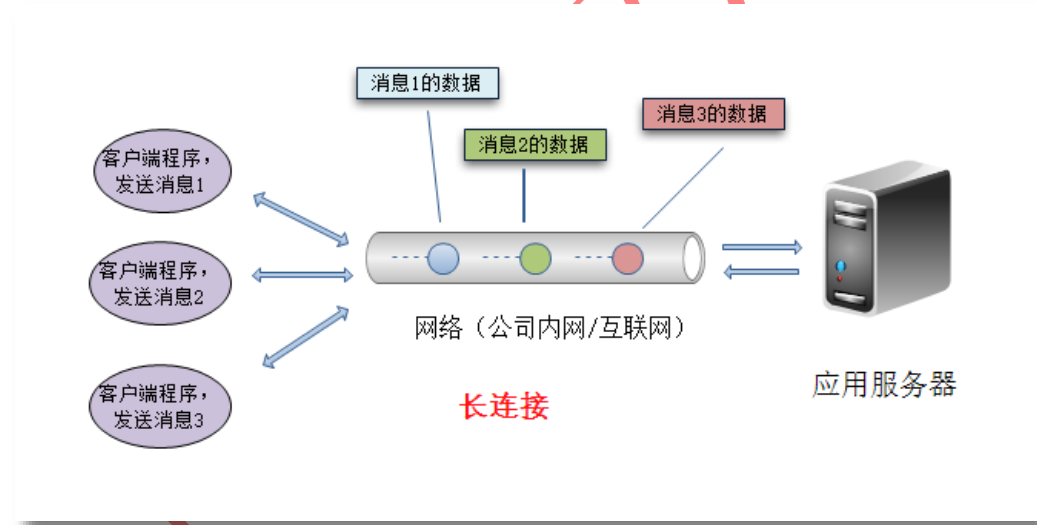
Dubbo 协议不适合传送大数据量的服务，比如传文件，传视频等，除非请求量很低

## D、使用 Dubbo 协议

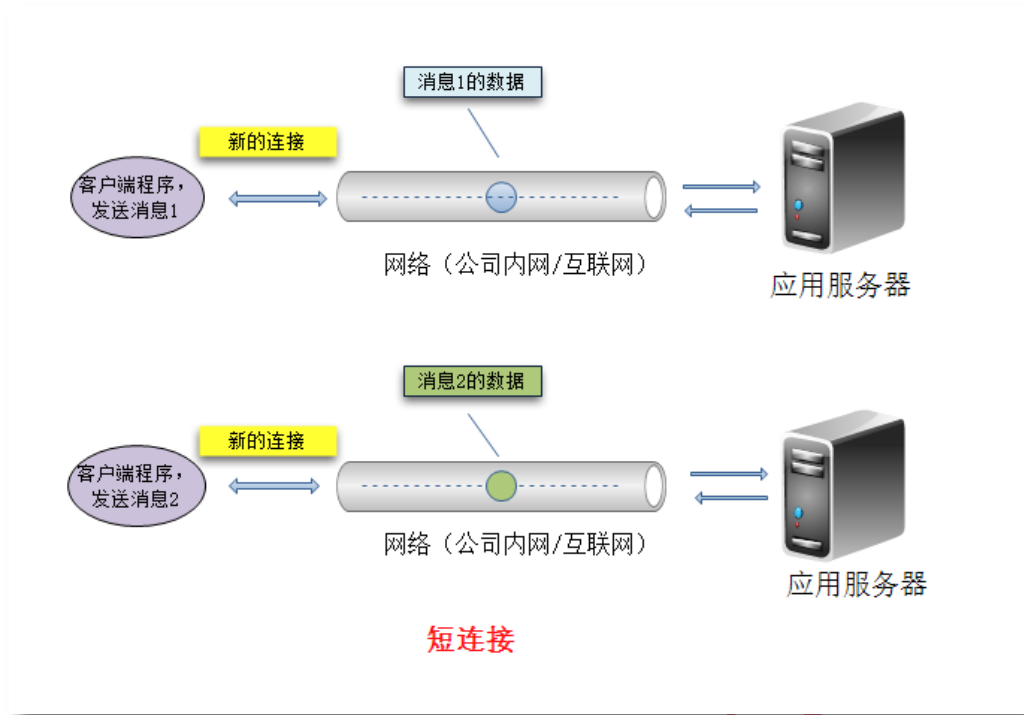
```
<dubbo:protocol name="dubbo" port="20880" />
```

### 2.2.2 长连接和短连接

Dubbo 协议使用的长连接：



短链接：

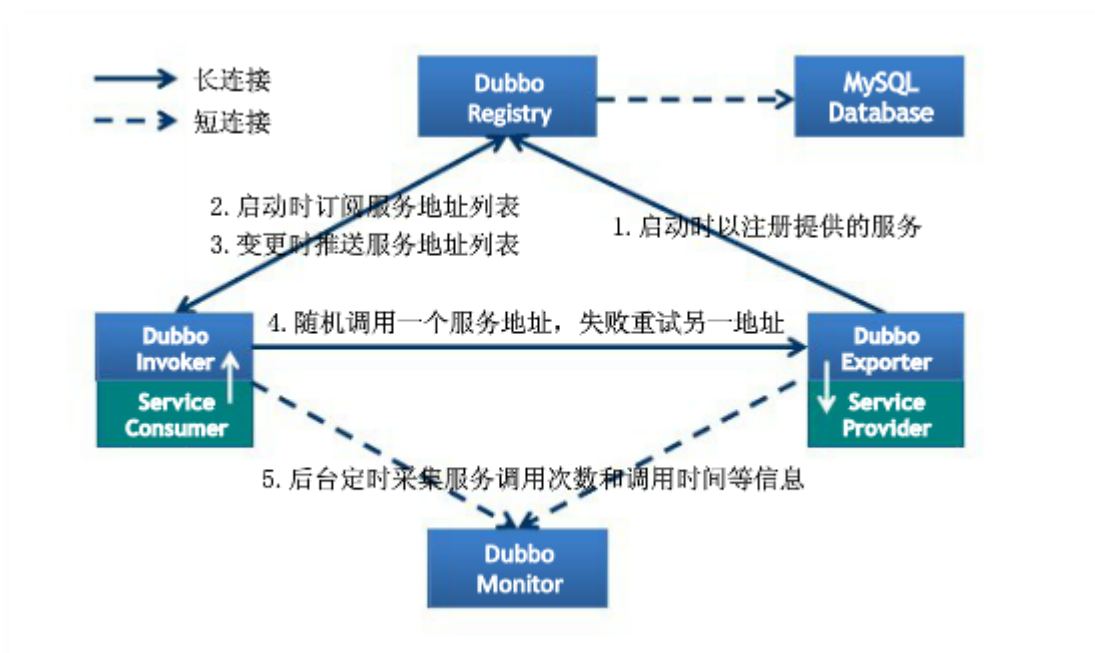


长连接和短连接接：

所谓长连接，指在一个连接上可以连续发送多个数据包，在连接保持期间，如果没有数据包发送，需要双方发检测包。短连接是指通讯双方有数据交互时，就建立一个连接，数据发送完成后，则断开此连接，即每次连接只完成一项业务的发送。

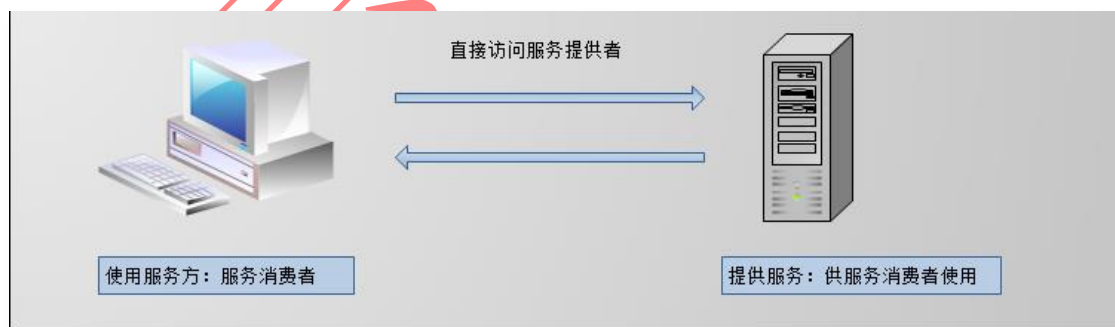
长连接多用于操作频繁，点对点的通讯，而且连接数不能太多情况。例如：数据库的连接用长连接。像 Web 网站的 http 服务一般都用短链接，因为长连接对于服务端来说会耗费一定的资源，而像 Web 网站频繁的用，使用短连接会更省一些资源，并发量大，但每个用户无需频繁操作情况下需用短连好。

## 2.3 Dubbo 的组件



## 2.4 使用 Dubbo 的第一个项目

点对点的直连项目:消费者直接访问服务提供者，没有注册中心。消费者必须指定服务提供者的访问地址（url）。



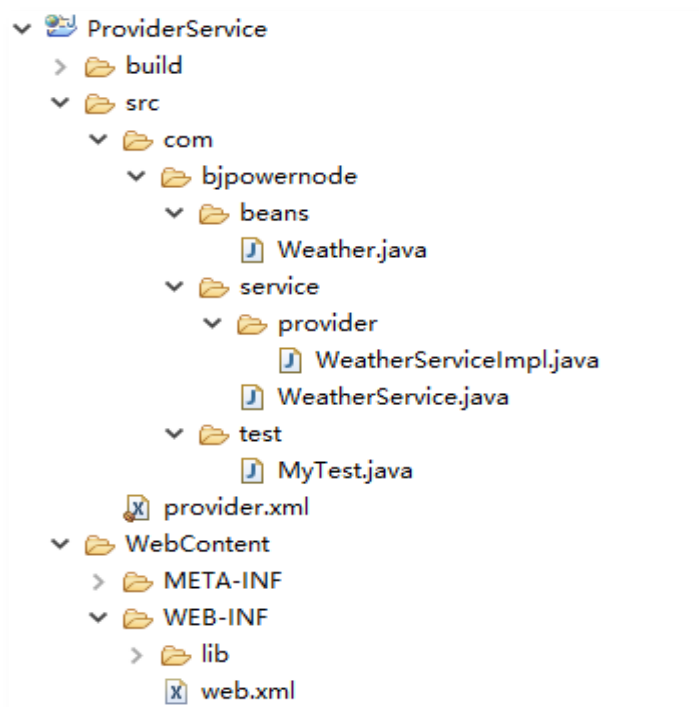
### 2.4.1 服务提供者（在 web 容器中使用）

服务提供者开发步骤：

- 定义服务接口（该接口需单独打包，在服务提供方和消费方共享）
- 在服务提供方实现接口（对服务消费方隐藏实现）
- 用 Spring 配置声明暴露服务

- 加载 Spring 配置（创建 bean）

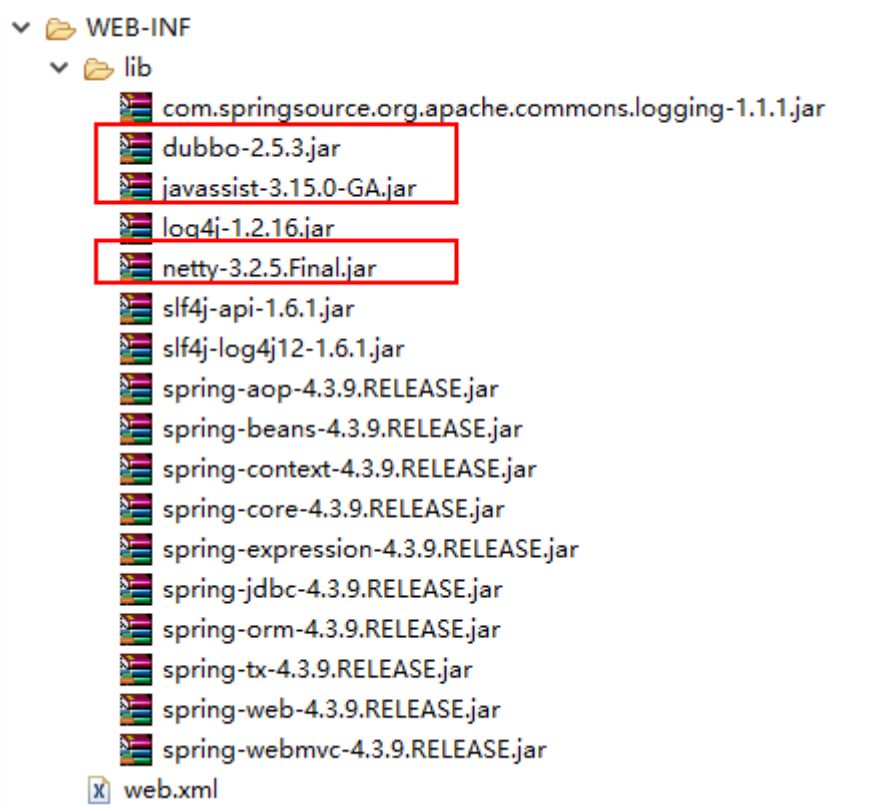
项目结构：



#### A、新建 web 项目，命名：04-dubboProviderService

注意：1) 选择 Tomcat 作为服务器。 2) 生成 web.xml 文件。

## B、导入 jar



dubbo.jar: Dubbo 框架的实现

javaassist-3.15.0-GA.jar: 字节码生成 jar

netty-3.2.5.Final.jar: 网络传输

spring-\*.jar: Dubbo 是基于 spring 的。配置 bean。

## C、定义表示天气信息的对象 Weather

```
import java.io.Serializable;

public class Weather implements Serializable {

    private static final long serialVersionUID = 1L;
    //城市
    private String city;
    //天气的描述
    private String desc;
    //湿度
    private String sd;
    //温度
    private int temp;
    //set ,get
}
```

必须实现序列化接口，网络传输中对象是转为二进制数据处理

重写的 toString()



```
@Override
public String toString() {
    return "天气信息: " + city + "的天气: " + desc + ", 湿度: " + sd + ", 温度: " + temp ;
}
```

不实现 Serializable 接口的错误提示:

```
java.lang.IllegalStateException: Serialized class com.bjpowernode.beans.Weather must implement java.io.Serializable
    at com.alibaba.com.caucho.hessian.io.SerializerFactory.getDefaultSerializer(SerializerFactory.java:261)
    at com.alibaba.com.caucho.hessian.io.SerializerFactory.getSerializer(SerializerFactory.java:233)
    at com.alibaba.com.caucho.hessian.io.Hessian2Output.writeObject(Hessian2Output.java:406)
```

#### D、定义服务的接口（面向接口编程）

```
package com.bjpowernode.service;
import com.bjpowernode.beans.Weather;
public interface WeatherService {
    //获取的天气信息
    Weather getWeather(String cityCode);
}
```

#### E、定义天气接口的实现类

包名: package com.bjpowernode.service.provider

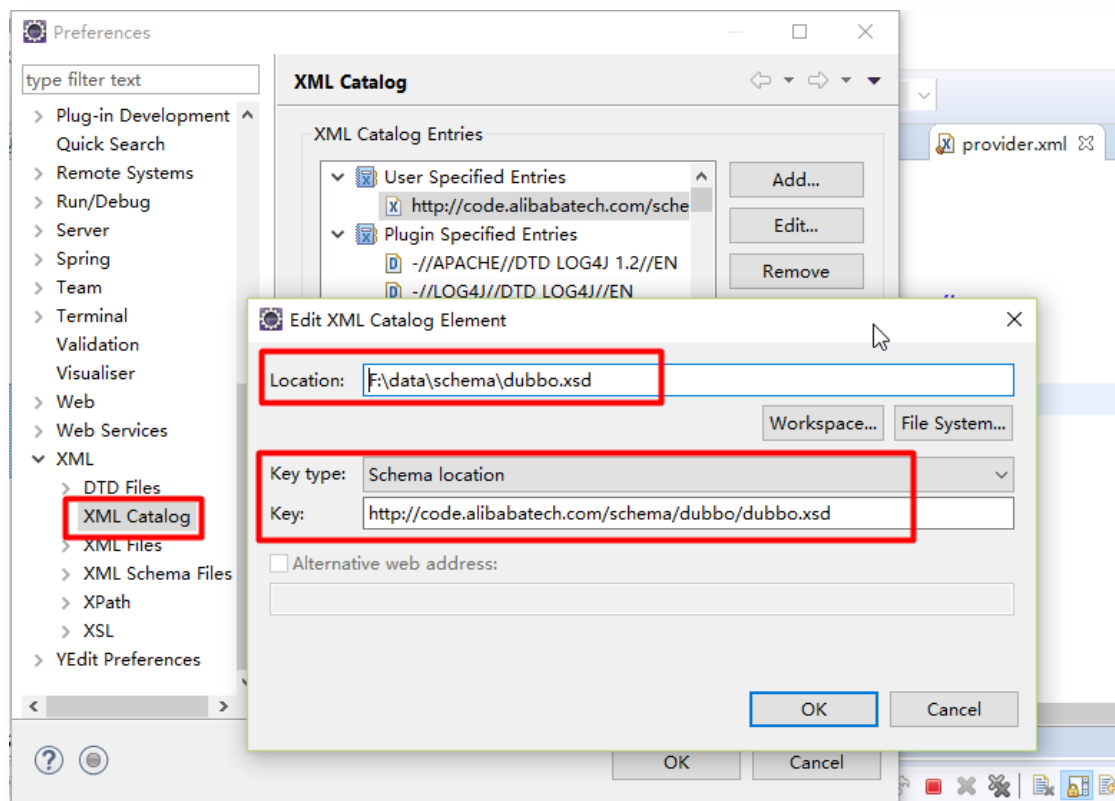
```
public class WeatherServiceImpl implements WeatherService {
    @Override
    public Weather getWeather(String cityCode) {
        Weather wea = new Weather();
        if( "101010100".equals(cityCode)){
            //北京天气
            wea.setCity("北京");
            wea.setDesc("天气晴朗, 空气新鲜, 温度适中, 适合外出");
            wea.setTemp(25);
            wea.setSd("25%");
        } else if("101010200".equals(cityCode)){
            //上海天气
            wea.setCity("上海");
            wea.setDesc("多云, 午后有小阵雨, 温度适中");
            wea.setTemp(29);
            wea.setSd("30%");
        } else {
            //全国天气
            wea.setCity("全国");
            wea.setDesc("北方晴朗, 南方阴雨");
            wea.setTemp(29);
            wea.setSd("30%");
        }
        return wea;
    }
}
```

## F、编写 Spring 配置文件，Spring 作为容器管理对象

第一步：加入：dubbo.xsd 约束文件

```
<?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://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">
```

第二步：本机配置代码提示：eclipse 菜单 Windows-→Preferences



第三步：声明服务定义

dubbo:application: 定义服务名称，一般使用项目名。

dubbo:service: 声明服务，暴露给消费者使用。

bean: 定义服务的实现类，提供服务的代码实现。spring 中的标签

```
<?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">

  <!-- 指定服务提供者的名称，唯一值 -->
  <dubbo:application name="WeatherServiceProvider"/>

  <!-- 暴露提供的服务
  interface: 服务接口的全限定类名
  ref: 实现服务接口的bean的id
  protocol: 访问服务使用的协议（访问的规则）
  registry: 注册中心信息，N/A: 不使用注册中心，采用点对点直连方式。
  -->
  <dubbo:service interface="com.bjpowernode.service.WeatherService"
    ref="weathserServiceImpl" protocol="dubbo" registry="N/A"/>

  <!-- 服务的具体实现 -->
  <bean id="weathserServiceImpl" class="com.bjpowernode.service.provider.WeatherServiceImpl" />
</beans>
```

## G、新建测试类：MyTest 测试配置文件，对象定义等是否正确。

```
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MyTest {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext ac = new ClassPathXmlApplicationContext("provider.xml");
        //容器启动，通知容器中所有的bean。
        ac.start();
    }
}
```

## H、修改 web.xml 文件，web 应用中使用 Spring

```
<!-- spring配置文件 -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:provider.xml</param-value>
</context-param>

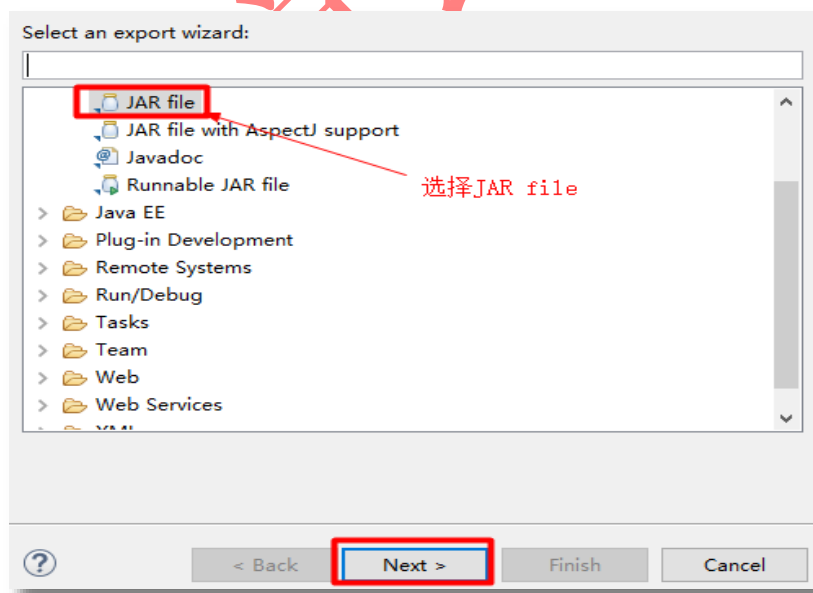
<!-- 创建Spring容器对象 -->
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

服务提供者的功能实现完成。

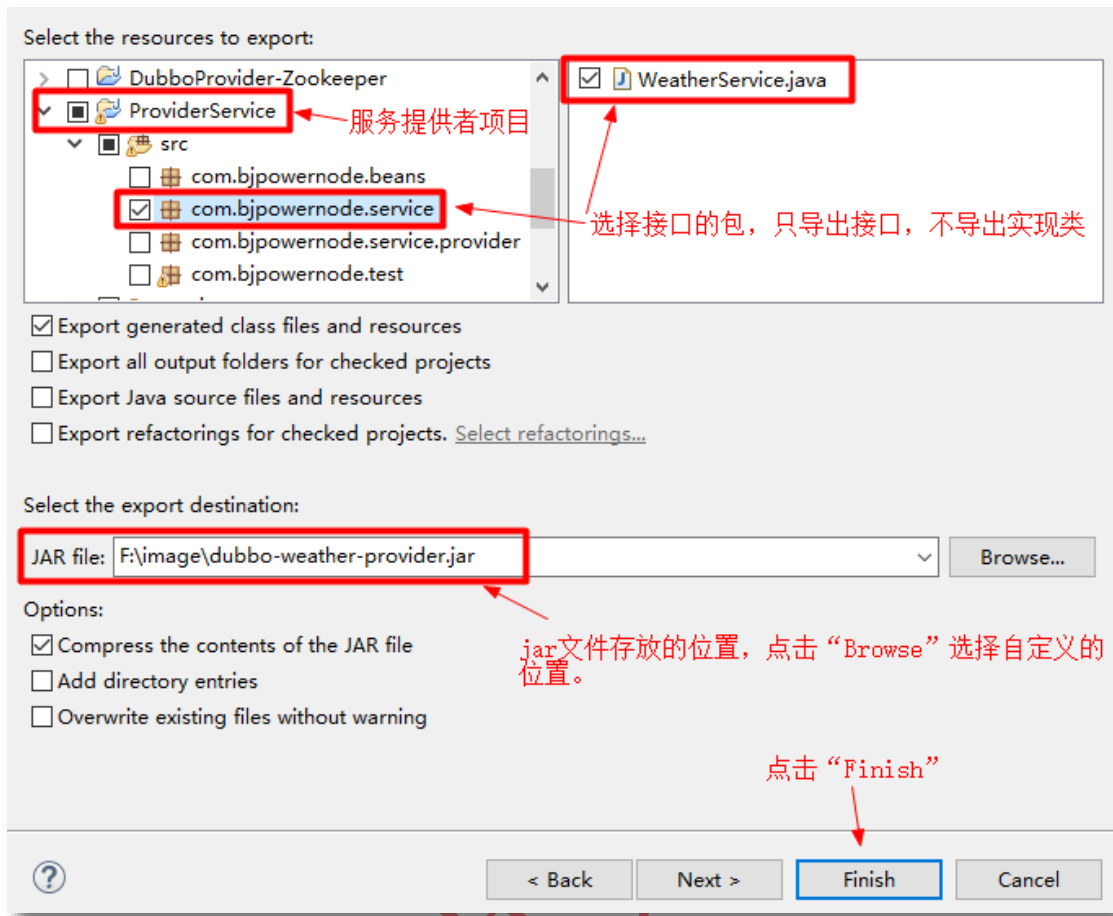
## I、导出服务接口

服务接口要给消费者使用，需要把接口定义打成 jar。

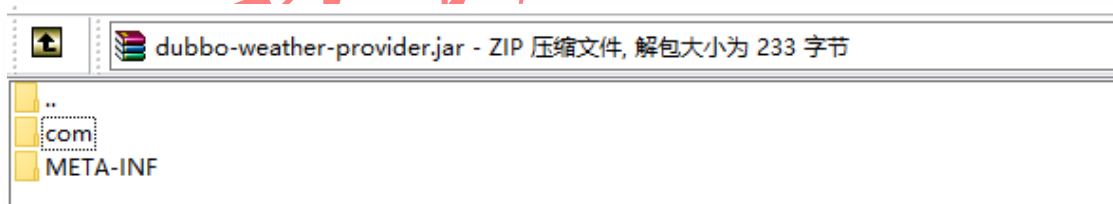
第一步：eclipse 菜单 File --- Export



第二步：导出接口



第三步：查看导出的 jar 文件  
使用 winrar 等压缩软件查看。



## 2.4.2 服务消费者

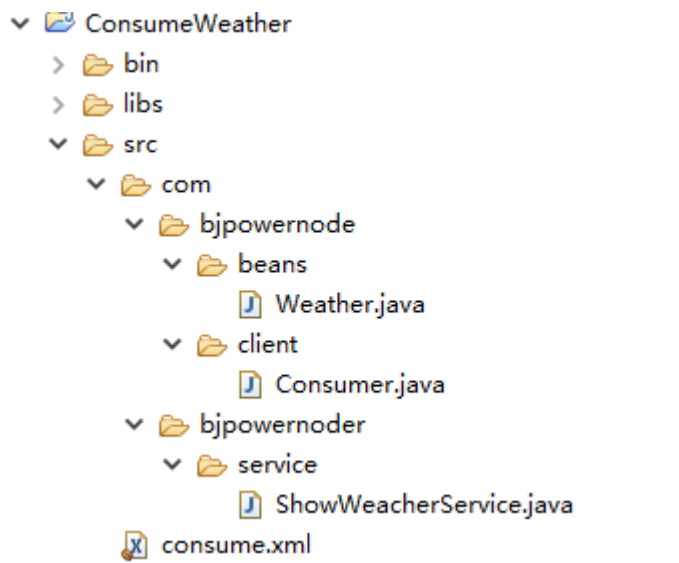
### (1) 直接使用远程服务提供者对象

服务消费者项目可以是 j2se, j2ee 等项目类型，使用服务。

服务消费者开发步骤：

- 通过 Spring 配置引用远程服务
- 加载 Spring 配置，并调用远程服务：(也可以使用 IoC 注入)

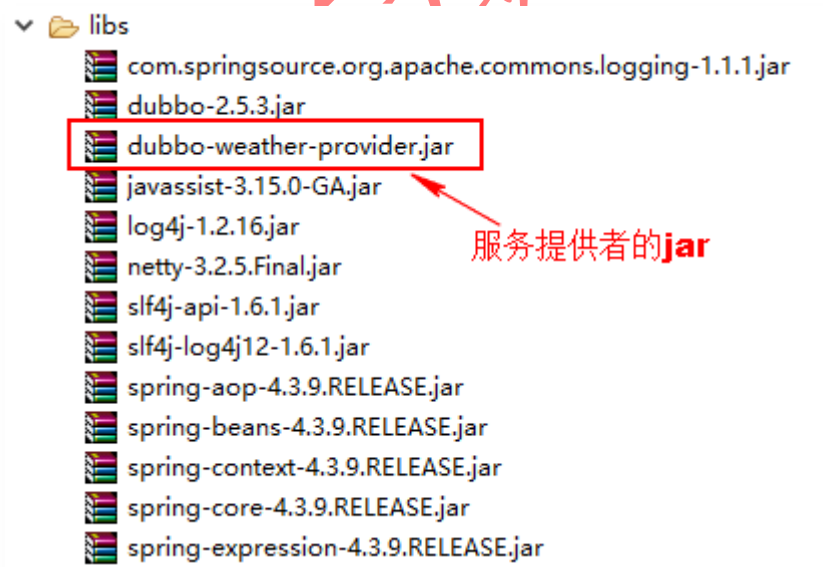
项目结构:



A、新建 java project ,命名 04-dubboConsume

B、导入 jar

工程根目录下，新建 libs，存放 jar。这个项目是非 web 应用，不需要 spring-web.jar, spring-webmvc.jar。必须导入服务提供者的 jar。上面步骤导出的 dubbo-weather-provider.jar



C、新建天气信息类 Weacher,结构同服务提供者项目中的 Weacher

可以拷贝服务提供者类中的 Weather, 或者服务提供者导出接口的同时，把 Weather 类和接

口一同导出到 dubbo-weather-provider.jar

```
public class Weather {
    //城市
    private String city;
    //天气的描述
    private String desc;
    //湿度
    private String sd;
    //温度
    private int temp;
    // set , get
}
```

重写的 toString()

```
@Override
public String toString() {
    return "天气信息: " + city + "的天气: " + desc + ", 湿度: " + sd + ", 温度: " + temp ;
}
```

#### D、新建配置文件 dubbo-consume.xml

配置文件需要加入 spring 和 dubbo 的约束文件，同服务提供者的 provider.xml 的操作方式。

```
<?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">

    <!-- 声明消费者的名称，唯一值 -->
    <dubbo:application name="ConsumeWeather"/>

    <!-- 声明使用的服务：生成服务的远程代理，就像使用本地服务一样。
        id:对象的名称，唯一表示这个对象
        interface:服务的接口（面向接口编程）
        protocol:使用的协议
        url:没有使用注册中心时，需要指定访问服务方的：访问协议和ip、端口
    -->
    <dubbo:reference id="weatherService" interface="com.bjpowernode.service.WeatherService"
        url="dubbo://localhost:20880"/>

</beans>
```

## E、定义测试类，访问远程服务

```
public class Consume {

    public static void main(String[] args) {
        ApplicationContext ac = new ClassPathXmlApplicationContext("dubbo-consume.xml");
        WeatherService weatherService = (WeatherService) ac.getBean("weatherService");
        Weather wea = weatherService.getWeather("101020100"); //上海
        System.out.println("wea:"+wea);
    }

}
```

## F、测试访问远程服务

首先启动，服务提供者。服务提供者 05-ProviderService 是 web 应用发布到 tomcat 上运行  
然后再运行 Consume 程序。

信息: [DUBBO] Refer dubbo service com.bjpowernode.service.WeatherService from url dubbo://localhost  
消费输出: 天气信息: 北京的天气: 天气晴朗, 空气新鲜, 温度适中, 适合外出, 湿度: 25%, 温度: 25

加入:

```
System.out.println("远程服务对象:"+weatherService.getClass().getName());
```

输出:

远程服务对象:com.alibaba.dubbo.common.bytecode.proxy0

## (2) 注入方式使用远程服务对象。

### A、新建使用远程服务对象的 Service

```
com.bjpowernode.service
> ShowWeacherService.java
```



```
public class ShowWeacherService {

    private WeatherService remote_weather_service;
    //设值注入
    public void setRemote_weather_service(WeatherService remote_weather_service) {
        this.remote_weather_service = remote_weather_service;
    }
    //获取的天气信息
    public String getWeatherInfo(String cityCode){
        Weather wea = remote_weather_service.getWeather(cityCode);
        return wea.toString();
    }
}
```

## B、修改 dubbo-consume.xml 声明 ShowWeatherService 对象，注入服务提供者对象

```
<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">

    <!-- 声明消费者的名称，唯一值 -->
    <dubbo:application name="ConsumeWeather"/>

    <!-- 声明使用的服务：生成服务的远程代理，就像使用本地服务一样。
        id:对象的名称，唯一表示这个对象
        interface:服务的接口（面向接口编程）
        protocol:使用的协议
        url:没有使用注册中心时，需要指定访问服务方的：访问协议和ip、端口
    -->
    <dubbo:reference id="weatherService" interface="com.bjpowernode.service.WeatherService"
        url="dubbo://localhost:20880"/>

    <bean id="showService" class="com.bjpowernode.service.ShowWeacherService">
        <property name="remote_weather_service" ref="weatherService" />
    </bean>
</beans>
```

## C、测试类

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

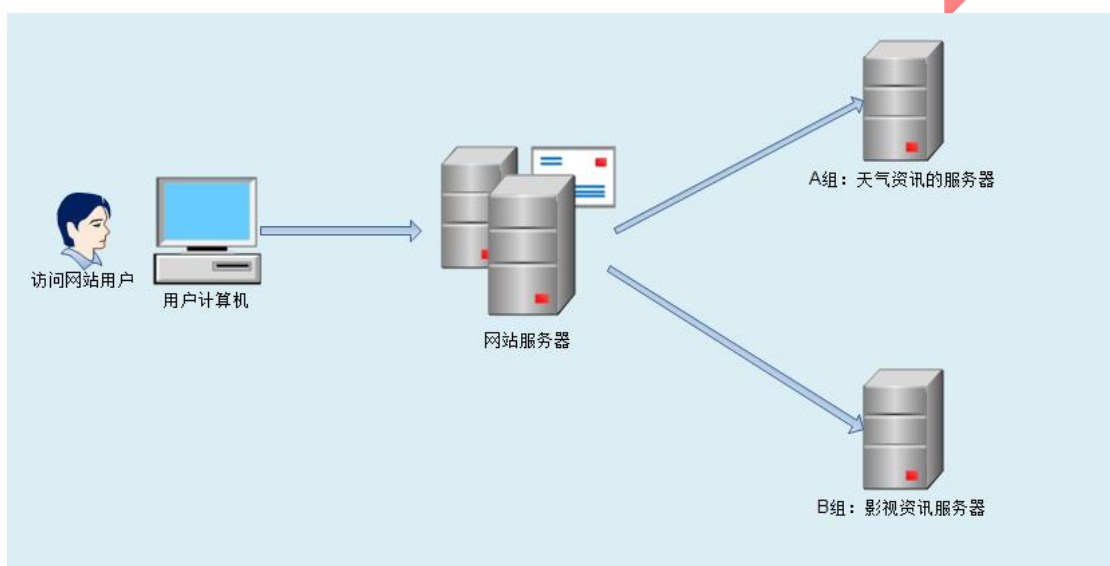
    ApplicationContext ac = new ClassPathXmlApplicationContext("consume.xml");
    LoggerFactory.setLevel(Level.OFF);
    ShowWeacherService showService =(ShowWeacherService) ac.getBean("showService");
    String weatherInfo= showService.getWeatherInfo("101010200"); //上海
    System.out.println("消费输出: "+ weatherInfo);

}
```

## 2.5 使用接口作为独立项目

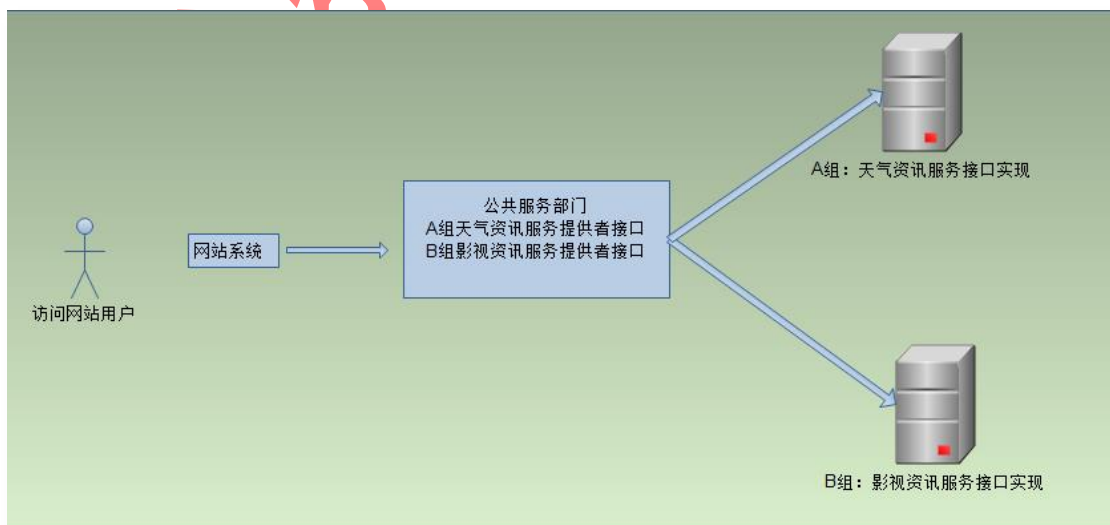
公司或者企业有很多的服务，这些服务有不同的部门，不同的人员管理，维护。例如公司做生活服务类业务的，提供类似 58 同城的业务。其中公共服务部门负责提供天气资讯和影视资讯两个内容。A 小组两个人负责天气资讯；B 小组三个人负责影视资讯。现在需要在公司的网站同时提供两种资讯。你作为网站开发人员需要使用 A 和 B 两个小组不同服务内容。使用 A 组，B 组的两个服务提供者接口。

图一：



公司使用 Dubbo 管理服务，A 组，B 组分别各自服务的接口的 jar 包。比如 A-Weather.jar，B-Movie.jar。网站的开发人员需要同时维护两个 jar。任何一个有改动，都需要做调整代码。

图二：



现在只要使用公共服务部门提供的一个服务接口 jar 包就可以了。

## 2.5.1 服务提供者接口定义

### A、新建 Java Project 项目:05-ProviderInteface

Java 项目，只定义接口，无需导入 jar。

### B、新建天气信息的数据类 Weatcher

```
import java.io.Serializable;

public class Weather implements Serializable {

    private static final long serialVersionUID = 1L;
    //城市
    private String city;
    //天气的描述
    private String desc;
    //湿度
    private String sd;
    //温度
    private int temp;
    //set ,get
}
```

必须实现序列化接口，网路传输中对象是转为二进制数据处理

重写的 toString()

```
@Override
public String toString() {
    return "天气信息: " + city + "的天气: " + desc + ", 湿度: " + sd + ", 温度: " + temp ;
}
```

### C、新建天气服务接口

```
package com.bjpowernode.service;
import com.bjpowernode.beans.Weather;
public interface WeatherService {
    //获取的天气信息
    Weather getWeather(String cityCode);
}
```

#### D、新建影视信息的数据类 Movie

```
import java.io.Serializable;

public class Movie implements Serializable {
    private static final long serialVersionUID = 1L;
    //电影名称
    private String name;
    //主演
    private String actor;
    //片长
    private int minute;
    // set , get
}
```

实现序列化接口

重新 toString()

```
public String toString() {
    return "电影: " + name + ",主演: " + actor + ", 片长: " + minute ;
}
```

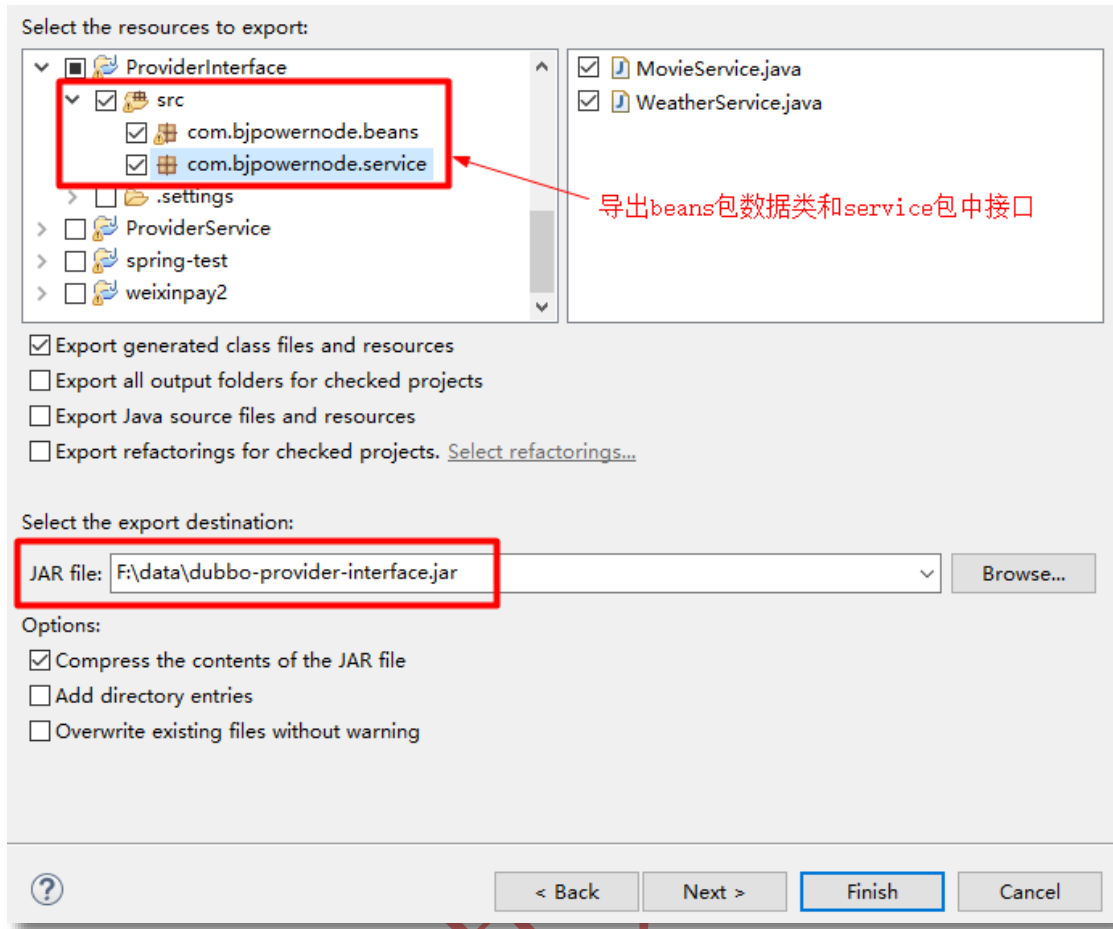
#### E、新建影视服务接口

```
import com.bjpowernode.beans.Movie;

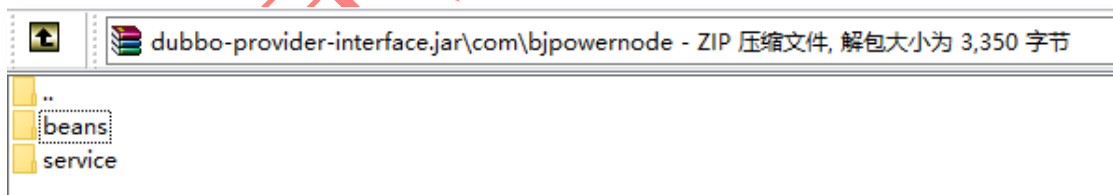
public interface MovieService {
    //受欢迎的电影列表
    public List<Movie> getMovies();
}
```

#### F、导出包含所有接口的 jar

导出 jar，选择 eclipse 菜单 File ----> Export



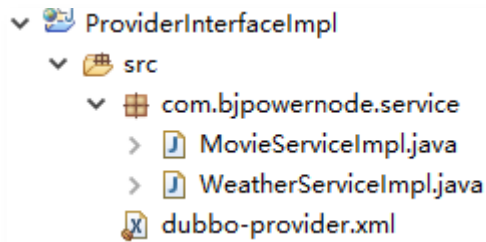
## G、查看导出的 jar 内容



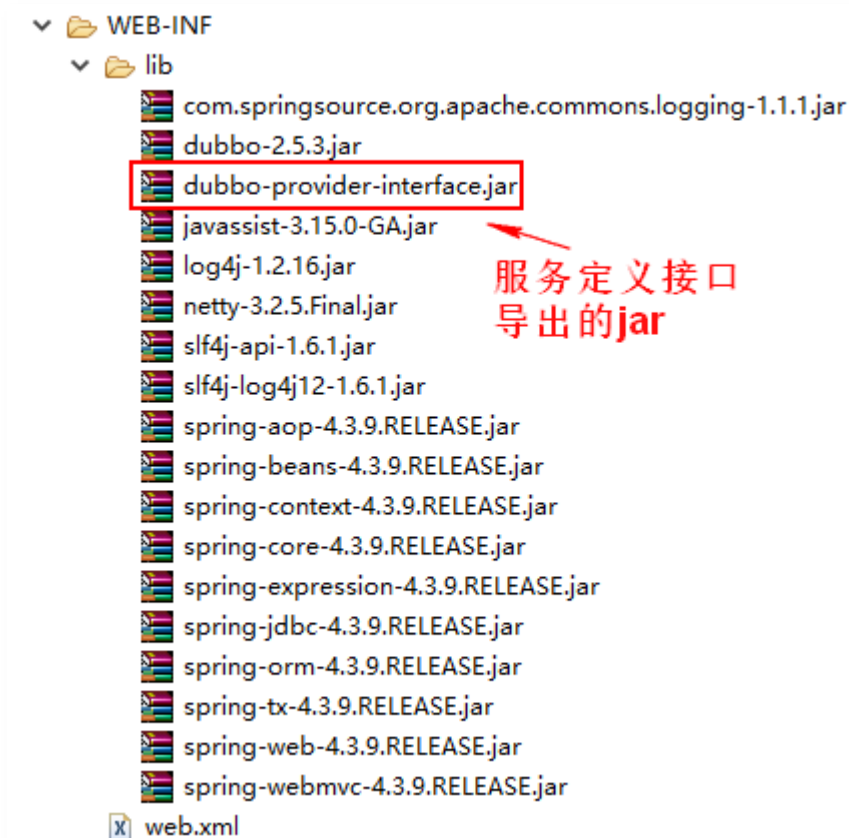
## 2.5.2 服务提供者接口实现

### A、新建 web 工程 05-ProviderInterfaceImpl, 生成 web.xml

项目结构:



## B、导入 dubbo，spring，服务接口的 jar



### C、新建实现 WeatherService 接口的实现类

```
public class WeatherServiceImpl implements WeatherService {  
    @Override  
    public Weather getWeather(String cityCode) {  
        Weather wea = new Weather();  
        if( "101010100".equals(cityCode)){  
            //北京天气  
            wea.setCity("北京");  
            wea.setDesc("天气晴朗, 空气新鲜, 温度适中, 适合外出");  
            wea.setTemp(25);  
            wea.setSd("25%");  
        } else if("101010200".equals(cityCode)){  
            //上海天气  
            wea.setCity("上海");  
            wea.setDesc("多云, 午后有小阵雨, 温度适中");  
            wea.setTemp(29);  
            wea.setSd("30%");  
        } else {  
            //全国天气  
            wea.setCity("全国");  
            wea.setDesc("北方晴朗, 南方阴雨");  
            wea.setTemp(29);  
            wea.setSd("30%");  
        }  
        return wea;  
    }  
}
```

### D、新建实现 MovieService 接口的实现类

```
public class MovieServiceImpl implements MovieService {  
    @Override  
    public List<Movie> getMovies() {  
  
        List<Movie> movies = new ArrayList<>();  
        Movie m1 = new Movie();  
        m1.setName("功夫");  
        m1.setActor("周星驰");  
        m1.setMinute(150);  
  
        Movie m2 = new Movie();  
        m2.setName("攻壳特工队");  
        m2.setActor("斯嘉丽");  
        m2.setMinute(120);  
  
        movies.add(m1);  
        movies.add(m2);  
  
        return movies;  
    }  
}
```

## E、新建 spring 配置文件 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">
  <dubbo:application name="DubboProviderService"/>
  <!-- 暴露提供的服务
       interface: 服务接口的全限定类名
       ref: 实现服务接口的bean的id
       protocol: 访问服务使用的协议（访问的规则）
       registry: 注册中心信息，N/A: 不使用注册中心，采用点对点直连方式。
  -->
  <dubbo:service interface="com.bjpowernode.service.MovieService" ref="movieServiceImpl"
                 protocol="dubbo" registry="N/A"/>
  <dubbo:service interface="com.bjpowernode.service.WeatherService" ref="weatherServiceImpl"
                 protocol="dubbo" registry="N/A" />
  <!-- 服务的具体实现 -->
  <!-- 天气服务 -->
  <bean id="weatherServiceImpl" class="com.bjpowernode.service.WeatherServiceImpl" />
  <!-- 电影服务 -->
  <bean id="movieServiceImpl" class="com.bjpowernode.service.MovieServiceImpl" />
</beans>
```

## F、修改 web.xml，增加 ContextLoaderListener 监听器

```
<!-- spring配置文件 -->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:dubbo-provider.xml</param-value>
</context-param>

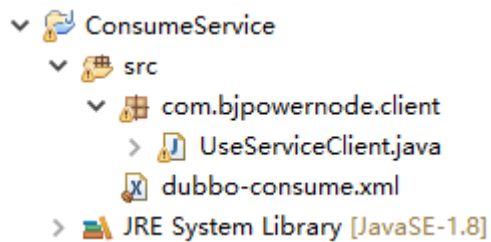
<!--创建spring容器的监听器-->
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

## 2.5.3 服务消费者

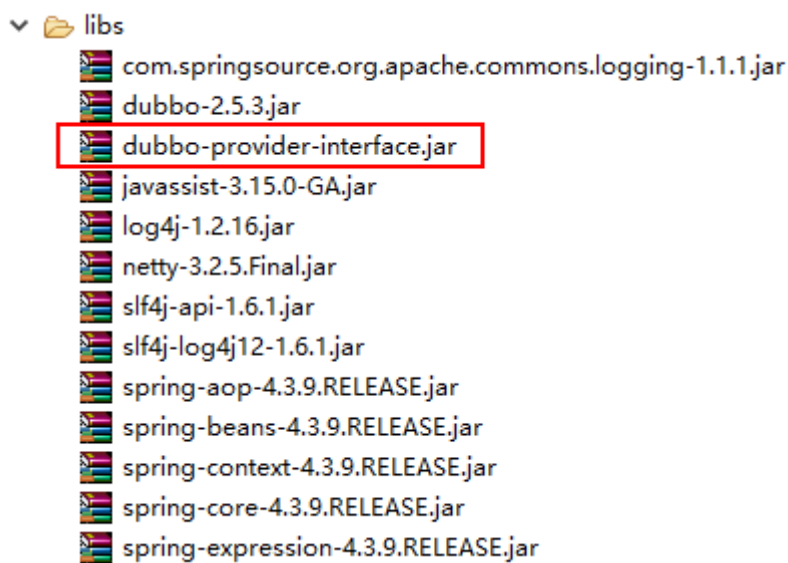
### A、新建 Java Project: 05-ConsumeService

项目结构:





## B、导入 dubbo, spring, 服务提供者接口 jar



### C、新建 spring 配置文件 dubbo-consume.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">

    <dubbo:application name="DubboConsumeClient"/>

    <!-- 声明使用的服务 :生成服务的远程代理, 就像使用本地服务一样。
        id:对象的名称, 唯一表示这个对象
        interface:服务的接口 (面向接口编程)
        url:没有使用注册中心时, 需要指定访问服务方的: 访问协议和ip、端口
    -->
    <dubbo:reference id="remoteWeatherService"
                   interface="com.bjpowernode.service.WeatherService"
                   url="dubbo://localhost:20880" />

    <dubbo:reference id="removeMovieService"
                   interface="com.bjpowernode.service.MovieService"
                   url="dubbo://localhost:20880" />

</beans>
```

### D、定义访问服务的测试类

```
public class UseServiceClient {
    public static void main(String[] args) {
        String resource="dubbo-consume.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(resource);

        //天气服务
        WeatherService ws = (WeatherService) ac.getBean("remoteWeatherService");
        Weather wea = ws.getWeather("101010100"); //北京
        System.out.println("访问远程天气服务: "+wea);

        //电影资讯
        MovieService ms = (MovieService) ac.getBean("removeMovieService");
        List<Movie> movies = ms.getMovies();
        for(Movie m : movies){
            System.out.println("访问远程电影服务: "+m);
        }
    }
}
```

服务提供者实现项目 06-ProviderInterfaceImpl 是 web 项目, 先发布到 tomcat, 启动应用。在执行 UseServiceClient.查看访问结果。

Successed connect to server /192.168.1.100:20880 from NettyClient 192.168.1.100 using dubbo version 2.5.3.

## 2.6 Dubbo 常用标签

Dubbo 中常用标签。分为三个类别：公用标签，服务提供者标签，服务消费者标签

### 2.6.1 公用标签

<dubbo:application/> 和 <dubbo:registry/>

#### A、配置应用信息

<dubbo:application name="服务的名称"/>

#### B、配置注册中心

<dubbo:registry address="ip:port" protocol="协议"/>

### 2.6.2 服务提供者标签

配置暴露的服务

<dubbo:service interface="服务接口名" ref="服务实现对象 bean">

### 2.6.3 服务消费者

配置服务消费者引用远程服务

<dubbo:reference id="服务引用 bean 的 id" interface="服务接口名"/>

## 第3章 注册中心

### 3.1 注册中心 Zookeeper

#### 3.1.1 为什么使用注册中心

对于服务提供方，它需要发布服务，而且由于应用系统的复杂性，服务的数量、类型也不断膨胀；对于服务消费方，它最关心如何获取到它所需要的服务，而面对复杂的应用系统，需要管理大量的服务调用。

而且，对于服务提供方和服务消费方来说，他们还有可能兼具这两种角色，即需要提供服务，有需要消费服务。通过将服务统一管理起来，可以有效地优化内部应用对服务发布/使用的流程和管理。服务注册中心可以通过特定协议来完成服务对外的统一。Dubbo 提供的注册中心有如下几种类型可供选：

Multicast 注册中心：组播方式

Redis 注册中心：使用 Redis 作为注册中心

Simple 注册中心：就是一个 dubbo 服务。作为注册中心。提供查找服务的功能。

Zookeeper 注册中心：使用 Zookeeper 作为注册中心

推荐使用 Zookeeper 注册中心。

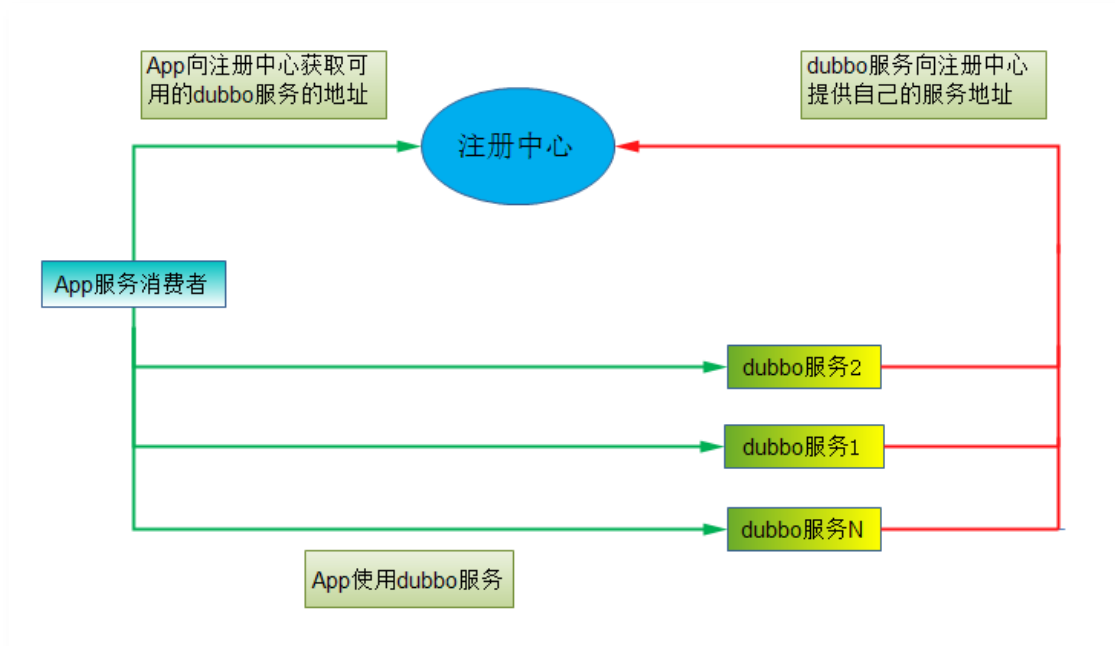
Zookeeper 是 Apache Hadoop 的子项目，是一个树型的目录服务，支持变更推送，适合作为 Dubbo 服务的注册中心，工业强度较高，可用于生产环境，并推荐使用。

Zookeeper 就像是 windows 的资源管理器，包含了所有的文件。都可以在这里找到。服务提供者和服务消费者，都在注册中心登记。服务消费者使用的访问地址不用写死。而且注册中心使用“心跳”机制更新服务提供者的状态。不能使用的服务提供者会自动从注册中心删除。服务提供者不会调用不能使用的服务。

#### 3.1.2 注册中心是什么？

Zookeeper 是一个高性能的，分布式的，开放源码的分布式应用程序协调服务。简称 zk。Zookeeper 是翻译管理是动物管理员。可以理解为 windows 中的资源管理器或者注册表。他是一个树形结构。这种树形结构和标准文件系统相似。ZooKeeper 树中的每个节点被称为 Znode。和文件系统的目录树一样，ZooKeeper 树中的每个节点可以拥有子节点。每个节点表示一个唯一服务资源。Zookeeper 运行需要 java 环境。

图一：



### 3.1.3 注册中心 Zookeeper 怎么用

#### A、Zookeeper 下载

官网下载地址: <http://zookeeper.apache.org/>

进入官网地址, 首页找到下载地址, 现在稳定版本是 3.4.10

#### What is ZooKeeper?

ZooKeeper is a centralized service for maintaining configuration information group services. All of these kinds of services are used in some form or and there is a lot of work that goes into fixing the bugs and race conditions th kinds of services, applications initially usually skimp on them ,which make t when done correctly, different implementations of these services lead to m

Learn more about ZooKeeper on the [ZooKeeper Wiki](#).

#### Getting Started

Start by installing ZooKeeper on a single machine or a very small cluster.

1. **Learn about** ZooKeeper by reading the documentation.
2. **Download** ZooKeeper from the release page.

选择镜像

## Apache ZooKeeper™ Releases

The Apache ZooKeeper system for distributed coordination is a high-performance service for

- [Download](#)
- [Release Notes](#)
- [News](#)

### Download

Releases may be downloaded from Apache mirrors: [Download](#)

On the mirror, all recent releases are available, but are not guaranteed to be stable. For sta

点击“Download”



We suggest the following mirror site for your download:

<http://mirror.bit.edu.cn/apache/zookeeper/>

Other mirror sites are suggested below. Please use the backup mirrors only to download working.

### HTTP

<http://apache.fayea.com/zookeeper/>

<http://mirror.bit.edu.cn/apache/zookeeper/>

<http://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/>

下载 3.4.10 版本

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">bookkeeper/</a>	2017-06-20 19:31	-	
<a href="#">current/</a>	2017-06-20 19:33	-	
<a href="#">stable/</a>	2017-06-20 19:33	-	
<a href="#">zookeeper-3.3.6/</a>	2017-06-20 19:33	-	
<a href="#">zookeeper-3.4.10/</a>	2017-06-20 19:33	-	
<a href="#">zookeeper-3.4.8/</a>	2017-06-20 19:32	-	
<a href="#">zookeeper-3.4.8/</a>	2017-06-20 19:33	-	
<a href="#">zookeeper-3.4.9/</a>	2017-07-30 18:17	-	
<a href="#">zookeeper-3.5.0-alpha/</a>	2017-06-20 19:33	-	
<a href="#">zookeeper-3.5.1-alpha/</a>	2017-06-20 19:33	-	
<a href="#">zookeeper-3.5.2-alpha/</a>	2017-06-20 19:32	-	
<a href="#">zookeeper-3.5.3-beta/</a>	2017-06-20 19:33	-	
<a href="#">HEADER.html</a>	2017-06-20 19:31	420	

## B、Windows 平台 Zookeeper 安装，配置

下载的文件 zookeeper-3.4.10.tar. 解压后到目录就可以了，例如 d:/servers/ zookeeper-3.4.10  
修改 zookeeper-3.4.10/conf/ 目录下配置文件

名称	修改日期
configuration	2017/3/23 18:14
log4j.properties	2017/3/23 18:14
zoo_sample.cfg	2017/3/23 18:14

复制 zoo-sample.cfg 改名为 zoo.cfg

文件内容：

```

1  # The number of milliseconds of each tick
2  tickTime=2000
3  # The number of ticks that the initial
4  # synchronization phase can take
5  initLimit=10
6  # The number of ticks that can pass between
7  # sending a request and getting an acknowledgement
8  syncLimit=5
9  # the directory where the snapshot is stored.
10 # do not use /tmp for storage, /tmp here is just
11 # example sake.
12 dataDir=F:/data/zookeeper
13 # the port at which the clients will connect
14 clientPort=2181
15 # the maximum number of client connections.

```

tickTime: 心跳的时间, 单位毫秒. Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔, 也就是每个 tickTime 时间就会发送一个心跳。表明存活状态。

dataDir: 数据目录, 可以是任意目录。存储 zookeeper 的快照文件、pid 文件, 默认为 /tmp/zookeeper, 建议在 zookeeper 安装目录下创建 data 目录, 将 dataDir 配置改为 /usr/local/zookeeper-3.4.10/data

clientPort: 客户端连接 zookeeper 的端口, 即 zookeeper 对外的服务端口, 默认为 2181

## C、Linux 平台 Zookeeper 安装、配置

Zookeeper 的运行需要 jdk。使用前 Linux 系统要安装好 jdk。

①: 上传 zookeeper-3.4.10.tar.gz 并解压

解压文件 zookeeper-3.4.10.tar.gz

执行命令: tar -zxvf zookeeper-3.4.10.tar.gz -C /usr/local/

```
[root@localhost ~]# ll zoo*  
-rw-r--r--. 1 root root 35042811 Sep  2 11:08 zookeeper-3.4.10.tar.gz  
[root@localhost ~]# tar -zxvf zookeeper-3.4.10.tar.gz -C /usr/local/
```

查看已解压:

```
[root@localhost local]# cd /usr/local/  
[root@localhost local]# ls -l  
total 8  
drwxr-xr-x. 2 root root      6 Nov  5 2016 lib64  
drwxr-xr-x. 2 root root      6 Nov  5 2016 libexec  
drwxr-xr-x. 10 mysql mysql 141 Aug 16 14:57 mysql5.7.18  
drwxrwxr-x. 6 root root    4096 Aug 27 16:09 redis-3.2.9  
drwxr-xr-x. 2 root root      6 Nov  5 2016 sbin  
drwxr-xr-x. 5 root root     49 Aug  5 23:11 share  
drwxr-xr-x. 2 root root      6 Nov  5 2016 src  
drwxr-xr-x. 10 mytest mytest 4096 Mar 23 19:28 zookeeper-3.4.10
```

②: 配置文件

在 zookeeper 的 conf 目录下, 将 zoo\_sample.cfg 改名为 zoo.cfg, cp zoo\_sample.cfg zoo.cfg  
zookeeper 启动时会读取该文件作为默认配置文件。

进入 zookeeper 目录下的 conf

```
[root@localhost conf]# pwd  
/usr/local/zookeeper-3.4.10/conf  
[root@localhost conf]# ll  
total 12  
-rw-rw-r--. 1 mytest mytest 535 Mar 23 18:14 configuration.xml  
-rw-rw-r--. 1 mytest mytest 2161 Mar 23 18:14 log4j.properties  
-rw-rw-r--. 1 mytest mytest 922 Mar 23 18:14 zoo_sample.cfg  
[root@localhost conf]#
```



拷贝样例文件 zoo-sample.cfg 为 zoo.cfg

```
[root@localhost conf]# ll
total 12
-rw-rw-r--. 1 mytest mytest 535 Mar 23 18:14 configuration.xml
-rw-rw-r--. 1 mytest mytest 2161 Mar 23 18:14 log4j.properties
-rw-rw-r--. 1 mytest mytest 922 Mar 23 18:14 zoo_sample.cfg
[root@localhost conf]# cp zoo_sample.cfg zoo.cfg
[root@localhost conf]# ll
total 16
-rw-rw-r--. 1 mytest mytest 535 Mar 23 18:14 configuration.xml
-rw-rw-r--. 1 mytest mytest 2161 Mar 23 18:14 log4j.properties
-rw-r--r--. 1 root root 922 Sep 2 11:21 zoo.cfg
-rw-rw-r--. 1 mytest mytest 922 Mar 23 18:14 zoo_sample.cfg
```

### ③：启动 Zookeeper

启动（切换到安装目录的 bin 目录下）：./zkServer.sh start

```
[root@localhost bin]# ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.4.10/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@localhost bin]# ps -ef | grep zoo
root      3569      1 17 11:22 pts/0    00:00:01 /usr/local/jdk1.8.0_
,CONSOLE -cp /usr/local/zookeeper-3.4.10/bin/../build/classes:/usr/loc
```

### ④：关闭 Zookeeper

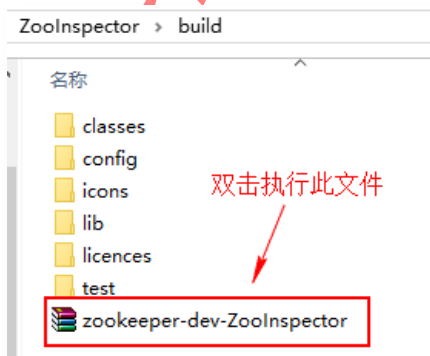
关闭（切换到安装目录的 bin 目录下）：./zkServer.sh stop

```
[root@localhost bin]# ./zkServer.sh stop
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.4.10/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
```

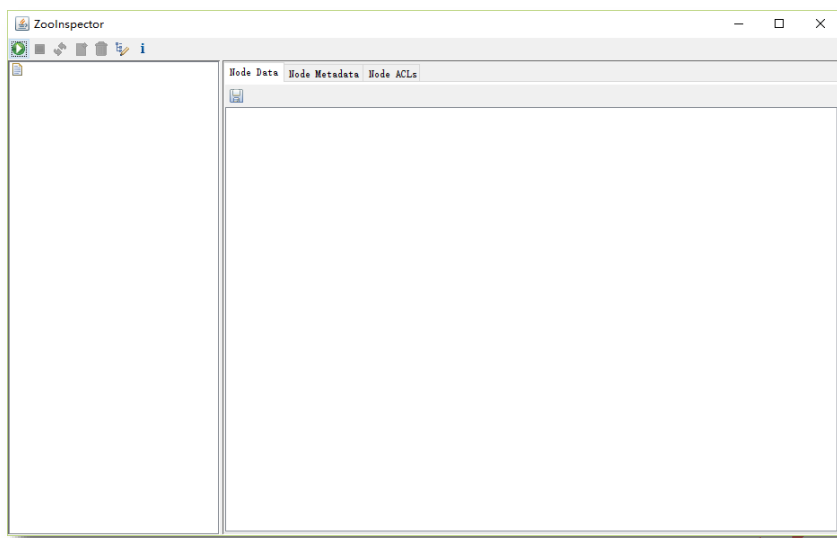
## D、Zookeeper 的客户端图形工具

Zookeeper 图形界面的客户端：ZooInspector.

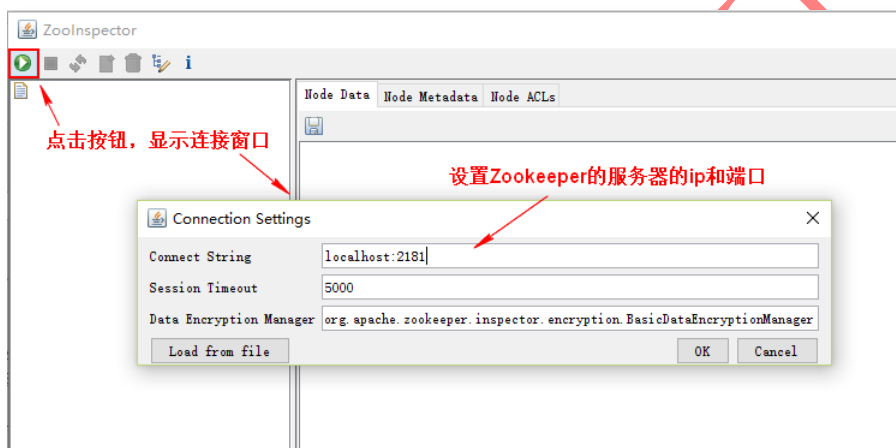
使用方式：Windows 上解压 ZooInspector 文件



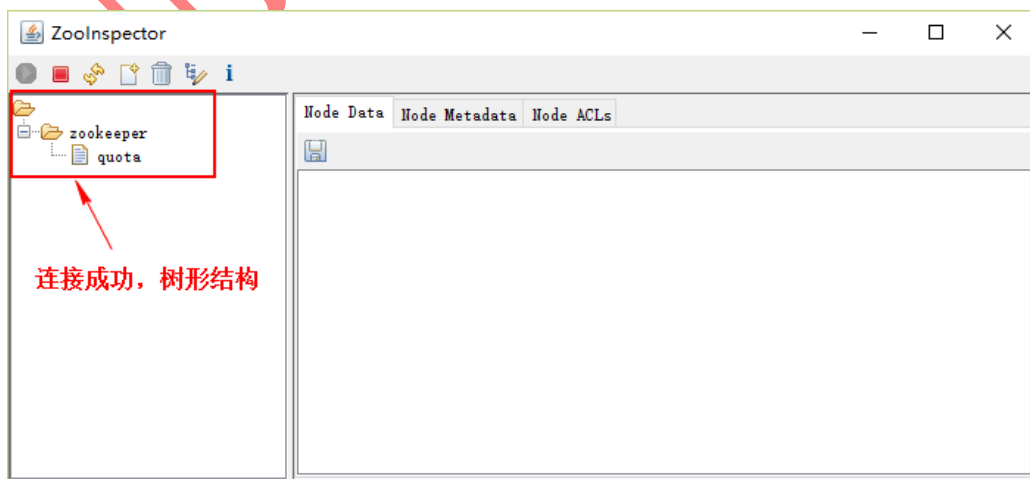
主界面:



配置连接:



连接成功:



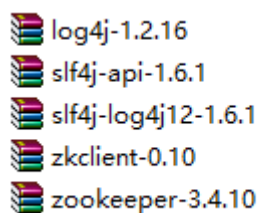
## 3.2 使用 Zookeeper 注册中心

### 3.2.1 服务提供者:

A、拷贝 05-ProviderInterfaceImpl 项目，命名为 06-ProviderInterfaceImplZk

B、导入 Zookeeper 相关 jar

在原来的 jar 基础上加入



log4j-1.2.16  
slf4j-api-1.6.1  
slf4j-log4j12-1.6.1  
zkclient-0.10  
zookeeper-3.4.10

C、修改 spring 配置文件

修改 dubbo-provider.xml

- 1) 加入注册中心<dubbo:registry address="zookeeper://localhost:2181" />
- 2) 去掉<dubbo:service>中的 protocol="dubbo" registry="N/A"

修改后文件内容:

```
<dubbo:application name="DubboProviderServiceZk"/>
<!-- 指定注册中心 -->
<dubbo:registry address="zookeeper://localhost:2181" />
<!-- 暴露提供的服务
    interface: 服务接口的全限定类名
    ref: 实现服务接口的bean的id
-->
<dubbo:service interface="com.bjpowernode.service.MovieService" ref="movieServiceImpl"/>
<dubbo:service interface="com.bjpowernode.service.WeatherService" ref="weatherServiceImpl"/>

<!-- 服务的具体实现 -->
<!-- 天气服务 -->
<bean id="weatherServiceImpl" class="com.bjpowernode.service.WeatherServiceImpl" />
<!-- 电影服务 -->
<bean id="movieServiceImpl" class="com.bjpowernode.service.MovieServiceImpl" />
</beans>
```

### 3.2.2 服务消费者

A、拷贝 05-ConsumeService 项目，重新命名为 06-ConsumeServiceZk

B、导入 Zookeeper 相关 jar

```
log4j-1.2.16
slf4j-api-1.6.1
slf4j-log4j12-1.6.1
zkclient-0.10
zookeeper-3.4.10
```

C、修改 spring 配置文件

修改 dubbo-consume.xml

- 1) 加入注册中心: `<dubbo:registry address="zookeeper://localhost:2181" />`
- 2) 去掉<dubbo:reference>中 `url="dubbo://127.0.0.1:20880"`

修改后配置如下:

```
<dubbo:application name="DubboConsumeClientZk"/>
<!-- 注册中心 -->
<dubbo:registry address="zookeeper://localhost:2181" />
<!-- 声明使用的服务 :生成服务的远程代理,就像使用本地服务一样。
    id:对象的名称,唯一表示这个对象
    interface:服务的接口(面向接口编程)
-->
<dubbo:reference id="remoteWeatherService"
    interface="com.bjpowernode.service.WeatherService" />

<dubbo:reference id="removeMovieService"
    interface="com.bjpowernode.service.MovieService"/>
</beans>
```

### 3.2.3 运行程序

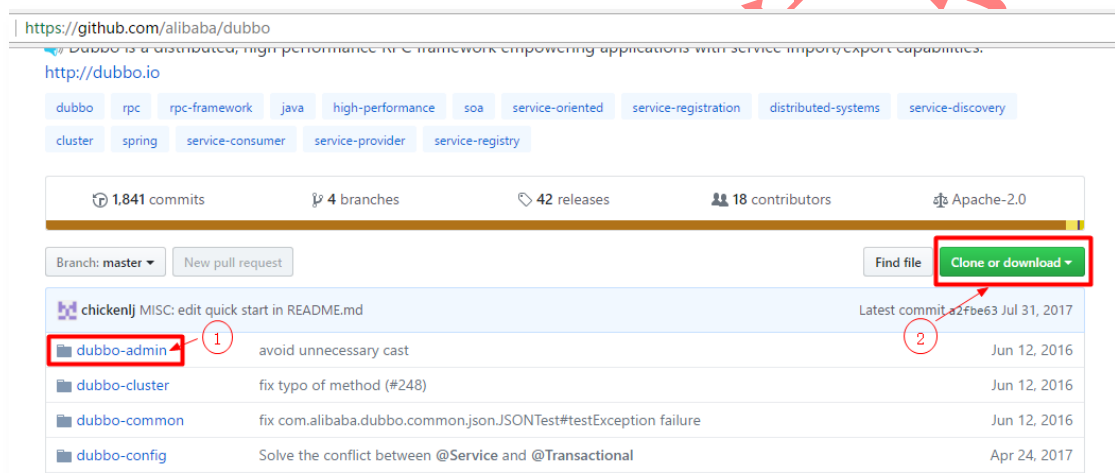
先运行 Zookeeper 注册中心，再启动服务提供者，最后运行服务消费者。一定要运行 Zookeeper。因为服务提供者要向 Zookeeper 注册服务。服务消费者需要在 Zookeeper 查找使用的服务。

## 第4章 监控中心

dubbo 的使用，其实只需要有注册中心，消费者，提供者这三个就可以使用了，但是并不能看到有哪些消费者和提供者，为了更好的调试，发现问题，解决问题，因此引入 dubbo-admin。通过 dubbo-admin 可以对消费者和提供者进行管理。可以在 dubbo 应用部署做动态的调整，服务的管理。

### 4.1 发布配置中心

#### A、下载监控中心，<https://github.com/alibaba/dubbo>

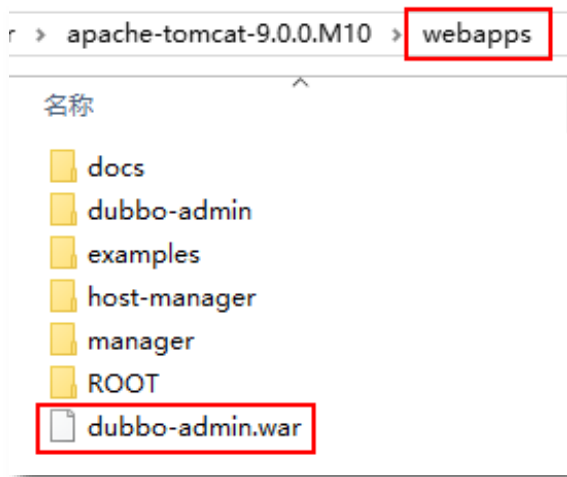


这里下载的是源代码，需要手工编译才能使用。可以在学习 maven 后才自己编译。

网络上编译好的，稳定的版本 dubbo-admin.war（和 rar 类似是一种压缩文件格式）

#### B、下载好的 dubbo-admin.war 部署到 tomcat 服务器的发布目录

把 dubbo-admin.war 文件拷贝到 tomcat 的 webapps



### C、修改配置 dubbo-properties 文件

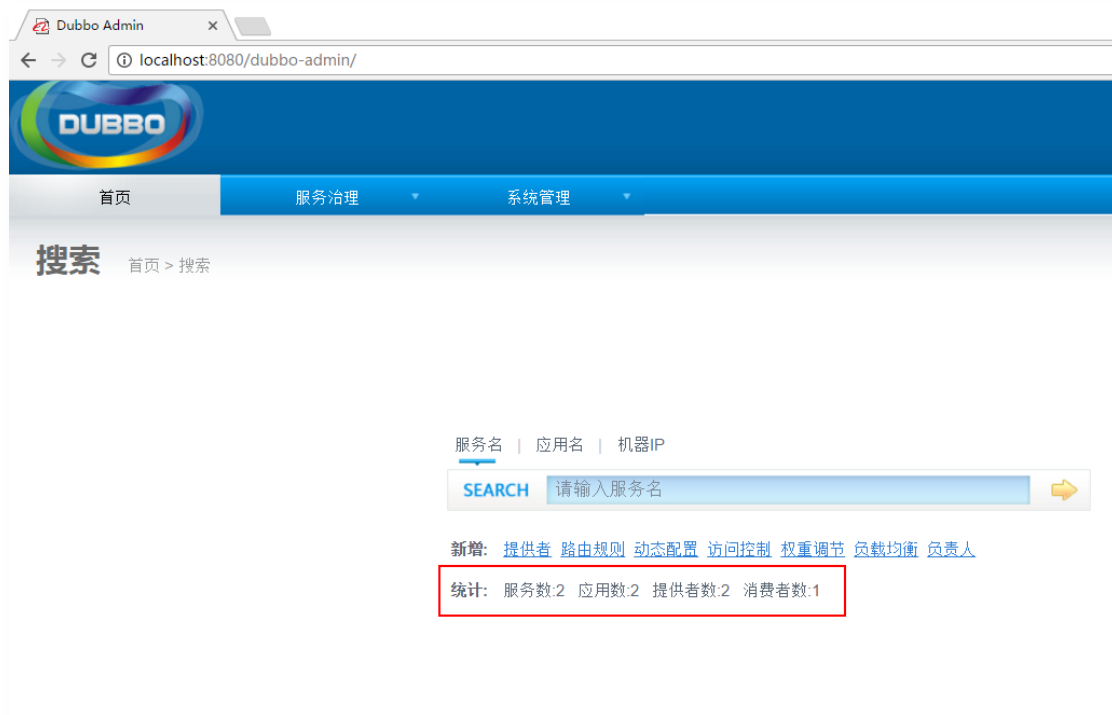
dubbo-admin 应用的 WEB-INF/dubbo-properties 文件，内容如下：

```
dubbo.registry.address=zookeeper://127.0.0.1:2181  
dubbo.admin.root.password=root  
dubbo.admin.guest.password=guest
```

注册中心地址  
控制台的登录用户root,密码root

### D、运行 dubbo-admin 应用

- 1) 先启动注册中心
- 2) 发布 dubbo 服务提供者和消费者到 tomcat，或者独立 jar 运行。
- 3) 启动 tomcat, 运行控制台 web 应用程序。
- 4) 在浏览器地址栏输入 `http://localhost:8080/dubbo-admin` 。访问监控中心-控制台。



## 4.2 发布 dubbo 项目

(1) 发布 web 项目 06-ProviderInterfaceImplZk 到 tomcat

(2) 服务消费项目改造

A、拷贝 06-ConsumeServiceZk,改为 07-ConsumeServiceZkAdmin

B、修改 main 方法代码

```
public static void main(String[] args) throws IOException {
    ApplicationContext ac = new ClassPathXmlApplicationContext("dubbo-consume.xml");
    //启动容器
    ((ClassPathXmlApplicationContext)ac).start();
    //让应用持续运行
    System.in.read();
}
```

## C、新建 InvokeService 类

```
public class InvokeService {

    private WeatherService ws;

    //设置注入
    public void setWs(WeatherService ws) {
        this.ws = ws;
    }

    //获取天气服务
    public String getWeatherMessage(){
        Weather wea = ws.getWeather("101010100"); //北京
        return wea.toString();
    }

}
```

## D、修改 dubbo-consume.xml, 注册 InvokeService

```
<dubbo:application name="DubboConsumeClientZk" />
<!-- 注册中心 -->
<dubbo:registry address="zookeeper://localhost:2181" />
<!-- 声明使用的服务 :生成服务的远程代理，就像使用本地服务一样。
    id:对象的名称，唯一表示这个对象
    interface:服务的接口（面向接口编程）
    check:false启动时不检查服务提供者的存在
-->
<dubbo:reference id="remoteWeatherService"
    interface="com.bjpowernode.service.WeatherService" check="false"/>

<dubbo:reference id="removeMovieService"
    interface="com.bjpowernode.service.MovieService" check="false" />

<!-- 注册使用服务的bean -->
<bean id="invokeService" class="com.bjpowernode.client.InvokeService">
    <property name="ws" ref="remoteWeatherService" />
</bean>
```

## (3) 运行 dubbo 项目

先启动 zookeeper, 再启动 tomcat, 浏览器输入 <http://localhost:8080/dubbo-admin/>  
首页:



搜索

首页 > 搜索

服务名 | 应用名 | 机器IP

SEARCH

请输入服务名



新增: [提供者](#) [路由规则](#) [动态配置](#) [访问控制](#) [权重调节](#) [负载均衡](#) [负责人](#)

统计: 服务数:2 应用数:2 提供者数:2 消费者数:1

服务治理---服务:

<input type="checkbox"/>	服务名: <input type="text"/>	状态: 所有
<input type="checkbox"/>	com.bjpowernode.service.MovieService	没有消费者
<input type="checkbox"/>	com.bjpowernode.service.WeatherService	正常

注意: 在 eclipse 中启动 tomcat, 有可能会有超时的情况。双击 tomcat 进行配置选项, 修改启动时间。增加启动超时的时间。

Timeouts

Specify the time limit to complete server operations.

Start (in seconds):

450

Stop (in seconds):

15