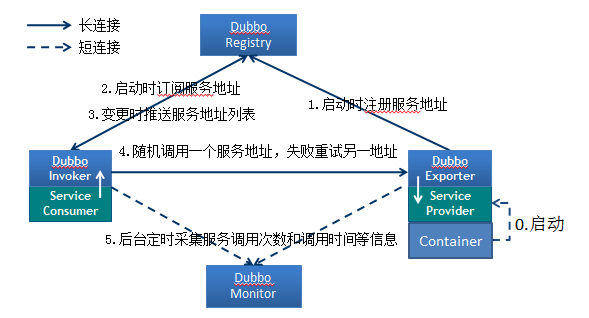
# 3.Dubbox框架

## 3.1 Dubbox简介

Dubbox 是一个分布式服务框架，其前身是阿里巴巴开源项目Dubbo ，被国内电商及互联网项目中使用，后期阿里巴巴停止了该项目的维护，当当网便在Dubbo基础上进行优化，并继续维护，为了与原有的Dubbo区分，故将其命名为**Dubbox**。

Dubbox 致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。简单的说，dubbox就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有dubbox这样的分布式服务框架的需求，并且本质上是个服务调用的东东，说白了就是个远程服务调用的分布式框架。



**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。

## 3.2注册中心Zookeeper

### 3.2.1 Zookeeper 介绍

官方推荐使用 zookeeper 注册中心。注册中心负责服务地址的注册与查找，相当于目录服务，服务提供者和消费者只在启动时与注册中心交互，注册中心不转发请求，压力较小。

Zookeeper 是 Apacahe 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Dubbox 服务的注册中心，工业强度较高，可用于生产环境。

### 3.2.2 Zookeeper 在Linux系统的安装

安装步骤：

第一步：安装 jdk（此步省略，我给大家提供的镜像已经安装好JDK）

第二步：把 zookeeper 的压缩包（资源\配套软件\dubbox\zookeeper-3.4.6.tar.gz）上传到 linux 系统。

Alt+P 进入SFTP ，输入put d:\zookeeper-3.4.6.tar.gz 上传

第三步：解压缩压缩包

|  |
| --- |
| tar -zxvf zookeeper-3.4.6.tar.gz |

第四步：进入 zookeeper-3.4.6 目录，创建 data 文件夹。

|  |
| --- |
| mkdir data |

第五步：进入conf目录 ，把 zoo\_sample.cfg 改名为 zoo.cfg

|  |
| --- |
| cd conf  mv zoo\_sample.cfg zoo.cfg |

第六步：打开zoo.cfg , 修改 data 属性：dataDir=/root/zookeeper-3.4.6/data

### 3.2.3 Zookeeper 服务启动

进入bin目录，启动服务输入命令

|  |
| --- |
| ./zkServer.sh start |

输出以下内容表示启动成功

## 3.3 Dubbox本地 JAR包部署与安装（了解）

Dubbox的jar包并没有部署到Maven的中央仓库中，大家在Maven的中央仓库中可以查找到Dubbo的最终版本是2.5.3 , 阿里巴巴解散了Dubbo团队后由当当网继续维护此项目，并改名为 Dubbox ,坐标不变，版本变更了，但是并没有提交到中央仓库。

我们现在需要手动将Dubbox的jar包安装到我的本地仓库中。

先将dubbo-2.8.4.jar包放到d:\setup, 然后输入命令

|  |
| --- |
| mvn install:install-file -Dfile=d:\setup\dubbo-2.8.4.jar -DgroupId=com.alibaba -DartifactId=dubbo -Dversion=2.8.4 -Dpackaging=jar |

## 3.4配置离线约束

地址：

<http://code.alibabatech.com/schema/dubbo/dubbo.xsd>

## 3.5入门小Demo

### 3.5.1服务提供者开发

开发步骤：

**（1）创建Maven工程（WAR）dubboxdemo-service ，在pom.xml中引入依赖**

|  |
| --- |
| <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>cn.itcast.dubboxdemo</groupId>  <artifactId>dubboxdemo-service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <properties>  <spring.version>4.2.4.RELEASE</spring.version>  </properties>  <dependencies>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jms</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>${spring.version}</version>  </dependency>  <!-- dubbo相关 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <version>2.8.4</version>  </dependency>  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  <version>3.4.6</version>  </dependency>  <dependency>  <groupId>com.github.sgroschupf</groupId>  <artifactId>zkclient</artifactId>  <version>0.1</version>  </dependency>  <dependency>  <groupId>javassist</groupId>  <artifactId>javassist</artifactId>  <version>3.11.0.GA</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>2.3.2</version>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <!-- 指定端口 -->  <port>8081</port>  <!-- 请求路径 -->  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

**（2）在工程的webapps下创建WEB-INF文件夹，创建web.xml**

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  version=*"2.5"*>  <!-- 加载spring容器 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext\*.xml</param-value>  </context-param>  <listener> <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  </web-app> |

**（3）创建业务接口**

创建包cn.itcast.dubbodemo.service，用于存放业务接口，创建接口

|  |
| --- |
| **package** cn.itcast.dubbodemo.service;  /\*\*  \* 业务接口  \* **@author** Administrator  \*  \*/  **public** **interface** UserService {  **public** String getName();  } |

**（4）创建业务实现类**

创建包cn.itcast.dubbodemo.service.impl ，用于存放业务实现类。创建业务实现类：

|  |
| --- |
| **package** cn.itcast.dubbodemo.service.impl;  import com.alibaba.dubbo.config.annotation.Service;  **import** cn.itcast.dubbodemo.service.UserService;  @Service  **public** **class** UserServiceImpl **implements** UserService {  **public** String getName() {  **return** "itcast";  }  } |

注意：Service注解与原来不同，需要引入com.alibaba包下的

**（5）编写配置文件**

在src/main/resources下创建applicationContext-service.xml ,内容如下：

|  |
| --- |
| <?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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"*>    <dubbo:application name=*"dubboxdemo-service"*/>  <dubbo:registry address=*"zookeeper://192.168.25.132:2181"*/>  <dubbo:annotation package="cn.itcast.dubboxdemo.service" />  </beans> |

注意：dubbo:annotation用于扫描@Service注解。

**（6）测试运行**

tomcat7:run

### 3.5.2服务消费者开发

开发步骤：

**（1）创建Maven工程（WAR）dubboxdemo-web** ，在pom.xml引入依赖 ，同“dubboxdemo-service”工程。区别就是把tomcat插件的运行端口改为8082 。

**（2）在webapps目录下创建WEB-INF 目录**，并创建web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  version=*"2.5"*>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharacterEncodingFilter</filter-name> <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <servlet>  <servlet-name>springmvc</servlet-name> <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- 指定加载的配置文件 ，通过参数contextConfigLocation加载-->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext-web.xml</param-value>  </init-param>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>\*.do</url-pattern>  </servlet-mapping>  </web-app> |

**（3）拷贝业务接口**

将“dubboxdemo-service”工程的cn.itcast.dubboxdemo.service 包以及下面的接口拷贝至此工程。

（4）编写Controller

|  |
| --- |
| **package** cn.itcast.dubboxdemo.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.ResponseBody;  **import** cn.itcast.dubbodemo.service.UserService;  @Controller  @RequestMapping("/user")  **public** **class** UserController {  @Reference  **private** UserService userService;  @RequestMapping("/showName")  @ResponseBody  **public** String showName(){  **return** userService.getName();  }  } |

**（5）编写spring配置文件**

在src/main/resources下创建applicationContext-web.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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd*  *http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd*  *http://www.springframework.org/schema/context* [*http://www.springframework.org/schema/context/spring-context.xsd"*>](http://www.springframework.org/schema/context/spring-context.xsd%22%3e)  <mvc:annotation-driven >  <mvc:message-converters register-defaults="false">  <bean class="org.springframework.http.converter.StringHttpMessageConverter">  <constructor-arg value="UTF-8" />  </bean>  </mvc:message-converters>  </mvc:annotation-driven>  <!-- 引用dubbo 服务 -->  <dubbo:application name=*"dubboxdemo-web"* />  <dubbo:registry address=*"zookeeper://192.168.25.132:2181"*/>  <dubbo:annotation package=*"cn.itcast.dubboxdemo.controller"* />  </beans> |

**（6）测试运行**

tomcat7:run

在浏览器输入http://localhost:8082/user/showName.do，查看浏览器输出结果

## 3.6管理中心的部署

我们在开发时，需要知道注册中心都注册了哪些服务，以便我们开发和测试。我们可以通过部署一个管理中心来实现。其实管理中心就是一个web应用，部署到tomcat即可。

### 3.6.1管理端安装

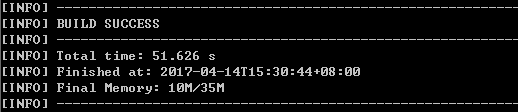
（1）编译源码，得到war包

给大家下发的资源中有个dubbox-master.zip ，这个是dubbox的源码，我们可以使用maven命令编译源码得到“管理端”的war包

将此压缩包解压，在命令符下进入dubbo-admin目录 ，输入maven命令

|  |
| --- |
| mvn package -Dmaven.skip.test=true |

如果你看到如下信息，就说明成功了



（2）进入target文件夹，你会看到一个dubbo-admin-2.8.4.war ， 在linux服务器上安装tomcat, 将此war包上传到linux服务器的tomcat的webapps下。为了访问方便，你可以把版本号去掉。 启动tomcat后自动解压。

（3）如果你部署在zookeeper同一台主机并且端口是默认的2181，则无需修改任何配置。如果不是在一台主机上或端口被修改，需要修改WEB-INF下的dubbo.properties ，修改如下配置：

|  |
| --- |
| dubbo.registry.address=zookeeper://127.0.0.1:2181 |

修改后重新启动tomcat

### 3.6.2管理端使用

（1）打开浏览器，输入http://192.168.25.132:8080/dubbo-admin/ ,登录用户名和密码均为root 进入首页。 (192.168.25.132:)是我部署的linux主机地址。



（2）启动服务提供者工程，即可在服务治理-提供者查看到该服务。



点击其中一条数据后可以查看详情。



（3）启动服务消费者工程，运行页面，观察“消费者”列表

