# 前提

SOA架构：面向服务的架构

面向服务架构，它可以根据需求通过网络对松散耦合的粗粒度应用组件进行分布式部署、组合和使用。服务层是SOA的基础，可以直接被应用调用，从而有效控制系统中与软件代理交互的人为依赖性。

有些表现层和服务层是不同的工程。所以要实现两个系统之间进行通信。

简单说就是把表现层和服务层独立分开不同的服务器（tomcat）进行服务

如何实现远程通信？

1、Webservice：效率不高基于soap协议。

2、使用restful形式的服务：http+json。很多项目中应用。如果服务太多，服务之间调用关系混乱，需要治疗服务。

3、使用dubbo。使用rpc协议进行远程调用，直接使用socket通信。传输效率高，并且可以统计出系统之间的调用关系、调用次数。

# dubbo

## 部署步骤

第一步 maven工程的创建

新建maven工程：两个tomcat独立运行服务端和表现层

cs-parent ‘根工程用于用于维护java版本’：可有可无

创建工程服务端

cs-manager(聚合工程) ：将它们聚合到wwl-manager，简化操作，如install只要操作wwl-manager即可，其它都一样

|--cs-manager-dao

|--cs-manager-interface （必须有:表现层部署dubbo和需继承此工程，因为service方式是war包不是jar包继承不了的）

|--cs-manager-pojo

|--cs-manager-service（打包方式改为war）

创建工程表现层

cs-manager-web（打包方式改为war）

第二步 dubbo和spring框架整合

服务层：1主要添加dubbo的jar包，2在springmvc.xml发布服务

1、在cs-manager-service工程中添加dubbo依赖的jar包。共三个：dubbo、zookeeper、zkclient

<!-- dubbo相关 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>2.5.3</version>

<!--exclusions 排除一些不需要同时下载的依赖jar -->

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring</artifactId>

</exclusion>

<exclusion>

<groupId>org.jboss.netty</groupId>

<artifactId>netty</artifactId><!-- 注册中心自带有,需排除此jar包 -->

</exclusion>

</exclusions>

</dependency>

<!-- 注册中心 配合dubbo使用 -->

<dependency>

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

<artifactId>zookeeper</artifactId>

<version>3.4.7</version>

</dependency>

<dependency>

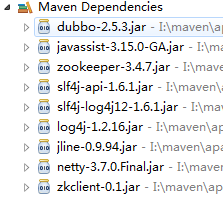
<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

<version>0.1</version>

</dependency>

最后生成的jar包如下：



2、添加tomcat插件配置。

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<configuration>

<path>/</path>这样配置：lacalHost:8081就是首页，不用工程名

<port>8081</port> 服务层是8080

</configuration>

</plugin>

</plugins>

</build>

3、在spring的配置文件中添加dubbo的约束，然后使用dubbo:service发布服务。

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context" xmlns:p="http://www.springframework.org/schema/p"

xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:dubbo="http://code.alibabatech.com/schema/dubbo" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.2.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-4.2.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.2.xsd

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

http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.2.xsd">

<context:component-scan base-package="wwl.service"/>

<!-- 使用dubbo发布服务 -->

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name="bills-service" /> <!-- 名字随便取 一般都是父工程名 -->

<!-- Zookeepe注册中心服务器的ip地址和端口,默认端口2181，可以配置多个 -->

<dubbo:registry protocol="zookeeper" address="106.12.106.21:2181,192.168.25.154:2182,192.168.25.154:2183" />

<!-- 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 service interface全路径名是接口类；ref属性值：必须是实现类以类名的小写字母开头-->

<!--timeout属性响应时间默认1000毫秒，接收方1秒没反应就断开，一般用于debug调试，dubbo默认尝试3次 -->

<dubbo:service interface="wwl.service.BillsService" ref="billsServiceImpl" timeout="600000"/>

</beans>

表现层工程 共四步： 添加dubbo的jar包、springmvc添加引用、添加tomcat插件

1、在cs-manager-web工程中添加dubbo依赖的jar包。和服务层一样(看服务层)

2、cs-manager-web添加对cs-manager-Interface的依赖。

3、在springmvc.xml的配置文件中添加服务的引用。

<?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-4.2.xsd

http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.2.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-4.2.xsd">

<context:component-scan base-package="cn.e3mall.controller" />

<mvc:annotation-driven />

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/jsp/" />

<property name="suffix" value=".jsp" />

</bean>

<!-- 引用dubbo服务 -->

<dubbo:application name="bills-web"/> <!--名字随便取 一般都是工程名 -->

<!--Zookeepe注册中心服务器的ip地址和端口，可以配置多个 -->

<dubbo:registry protocol="zookeeper" address="106.12.106.21:2181,192.168.25.154:2182,192.168.25.154:2183"/>

<!--全路径名：是接口类，表现层继承接口工程，id属性值：web工程的 controller类调用interface成员属性名-->

<dubbo:reference interface="wwl.service.BillsService" id="billsService" />

</beans>

4、在cs-manager-web工程中添加tomcat插件配置。

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<configuration>

<path>/</path>这样配置：lacalHost:8081就是首页，不用工程名

<port>8081</port> 服务层是8080

</configuration>

</plugin>

</plugins>

</build>

第三步 注册中心Zookeepe服务器进行开启

Linux系统：详情看Zookeepe配置

开启命令：[root@localhost bin]# ./zkServer.sh start

关闭：[root@localhost bin]# ./zkServer.sh stop

查看状态：[root@localhost bin]# ./zkServer.sh status

windows系统直接：解压后的bin目录下zkServer.cmd双击开启即可

共3步就能实现表现层和服务层独立一个tomcat开启：clean tomcat7:run

## dubbo

### 什么是dubbo

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 就是mvc三层架构

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 RPC

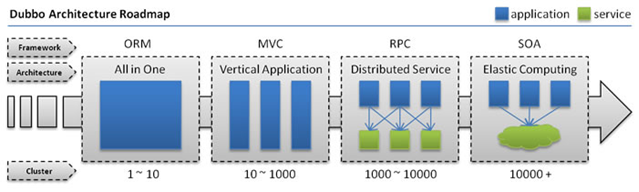
当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，使前端应用能更快速的响应多变的市场需求。此时，用于提高业务复用及整合的 **分布式服务框架(RPC)** 是关键。

流动计算架构 在分布式中进行管理调度服务

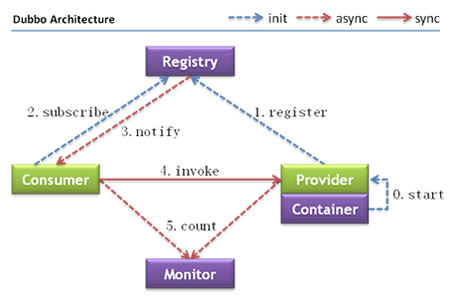
当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率。此时，用于提高机器利用率的 **资源调度和治理中心(SOA)** 是关键。

Dubbo就是资源调度和治理中心的管理工具。

下面的图就是演变过程



### Dubbo的架构



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。

### 使用方法介绍

Dubbo采用全Spring配置方式，透明化接入应用，对应用没有任何API侵入，只需用Spring加载Dubbo的配置即可，Dubbo基于Spring的Schema扩展进行加载。

单一工程中spring的配置

<bean id="xxxService" class="com.xxx.XxxServiceImpl" />

<bean id="xxxAction" class="com.xxx.XxxAction">

<property name="xxxService" ref="xxxService" />

</bean>

远程服务：

在本地服务的基础上，只需做简单配置，即可完成远程化：

将上面的local.xml配置拆分成两份，将服务定义部分放在服务提供方remote-provider.xml，将服务引用部分放在服务消费方remote-consumer.xml。

并在提供方增加暴露服务配置<dubbo:service>，在消费方增加引用服务配置<dubbo:reference>。

发布服务：

<!-- 和本地服务一样实现远程服务 -->

<bean id="xxxService" class="com.xxx.XxxServiceImpl" />

<!-- 增加暴露远程服务配置 -->

<dubbo:service interface="com.xxx.XxxService" ref="xxxService" />

调用服务：

<!-- 增加引用远程服务配置 -->

<dubbo:reference id="xxxService" interface="com.xxx.XxxService" />

<!-- 和本地服务一样使用远程服务 -->

<bean id="xxxAction" class="com.xxx.XxxAction">

<property name="xxxService" ref="xxxService" />

</bean>

## Zookeeper注册中心

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。使用dubbo-2.3.3以上版本，建议使用zookeeper注册中心。

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作用

1、可以作为集群的管理工具使用。

2、可以集中管理配置文件。

Zookeeper的安装

安装环境：

Linux：centos6.4

Jdk:1.7以上版本

Zookeeper是java开发的可以运行在windows、linux环境。需要先安装jdk。

安装步骤：

第一步：安装jdk

第二步：把zookeeper的压缩包上传到linux系统。

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。

第五步：把zoo\_sample.cfg改名为zoo.cfg

[root@localhost conf]# mv zoo\_sample.cfg zoo.cfg

第六步：修改data属性：dataDir=/root/zookeeper-3.4.6/data

第七步：启动zookeeper

[root@localhost bin]# ./zkServer.sh start

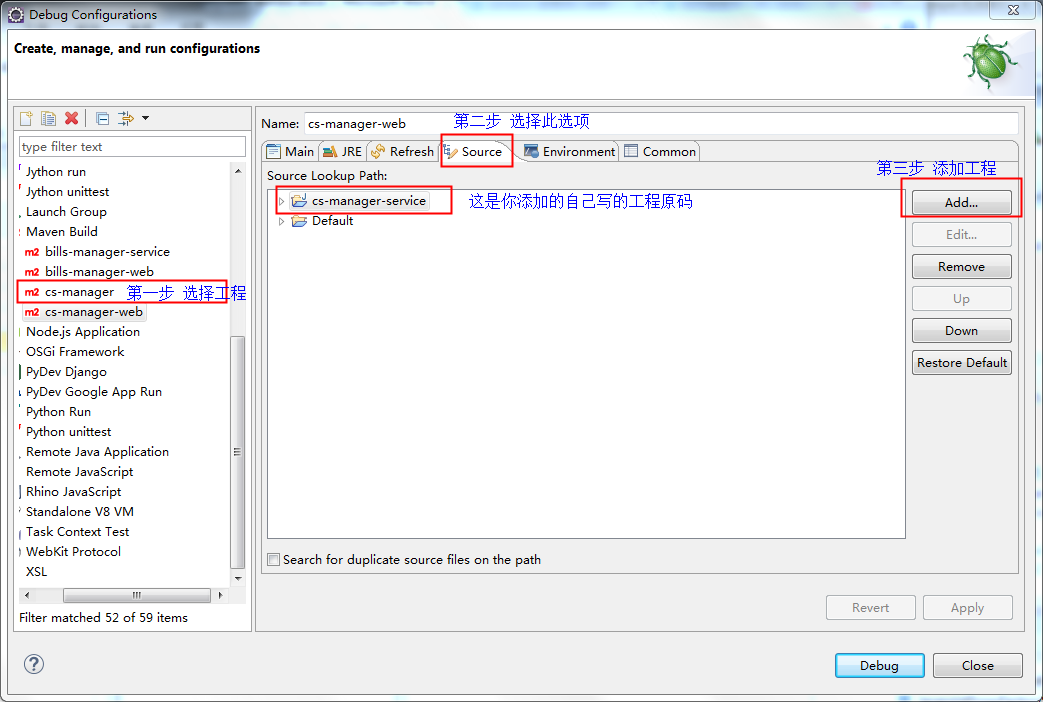
关闭：[root@localhost bin]# ./zkServer.sh stop

查看状态：[root@localhost bin]# ./zkServer.sh status

注意：需要关闭防火墙。

## debug调试

第一步 maven工程：添加自己写的源包



第二步 在服务端中设置响应时间，默认只有一秒

<!-- 使用dubbo发布服务 -->

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name="e3-manager" /> <! – 名字随便取 一般都是父工程名 –>

<dubbo:registry protocol="zookeeper"

<!—Zookeepe注册中心服务器的ip地址和端口,默认端口2181，可以配置多个 -->

address="192.168.25.154:2181,192.168.25.154:2182,192.168.25.154:2183" />

<!-- 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 service interface全路径名是接口类；ref属性值：必须是实现类以类名的小写字母开头-->

<!—timeout属性响应时间默认1000毫秒，一般用于debug调试，dubbo默认尝试3次 -->

<dubbo:service interface="cs.service.UserService" ref="userServiceImpl" timeout="600000"/>

## Dubbo监控中心

需要安装tomcat，然后部署监控中心即可。

1、部署监控中心：

[root@localhost ~]# cp dubbo-admin-2.5.4.war apache-tomcat-7.0.47/webapps/dubbo-admin.war

2、启动tomcat

3、访问http://192.168.25.167:8080/dubbo-admin/

用户名：root

密码：root

如果监控中心和注册中心在同一台服务器上，可以不需要任何配置。

如果不在同一台服务器，需要修改配置文件：

/root/apache-tomcat-7.0.47/webapps/dubbo-admin/WEB-INF/dubbo.properties

# dubbox