Dubbo教程

> Dubbo：一个分布式、高性能、透明化的 RPC 服务框架 作用：提供服务自动注册、自动发现等高效服务治理方案 官网：http://dubbo.apache.org/zh-cn/index.html
>
> 老师笔记加上自己的一点课堂笔记，如有错误请联系QQ：734229011进行修改：）😄

一，SOA

- 英文名称(Service Oriented Ambiguity)
- 中文名称：面向服务架构
  - 有一个专门提供服务单元
  - 其他所有单元都调用这个服务
- SOA 定位：
  - 如何**设计项目**，让开发时更有效率
  - SOA 是一种思想
- 之前项目架构设计
  1. 在公司项目不允许所有项目都访问数据库
  2. 开发时，数据库访问层代码可能出现冗余
- 使用 SOA 架构
  1. 专门访问数据库**服务(项目)**(相当于把 mapper 单独提出来作为一个项目)
  2. 开发时可以实现，数据访问控制和代码复用
- 实现 SOA 架构时，常用服务
  1. Dubbo 做为服务
  2. WebService 做为服务(比较老)
  3. Dubbox 做为服务
  4. 服务方就是 web 项目，调用 web 项目的控制器

     使用 HttpClient 可以调用其他项目的控制器

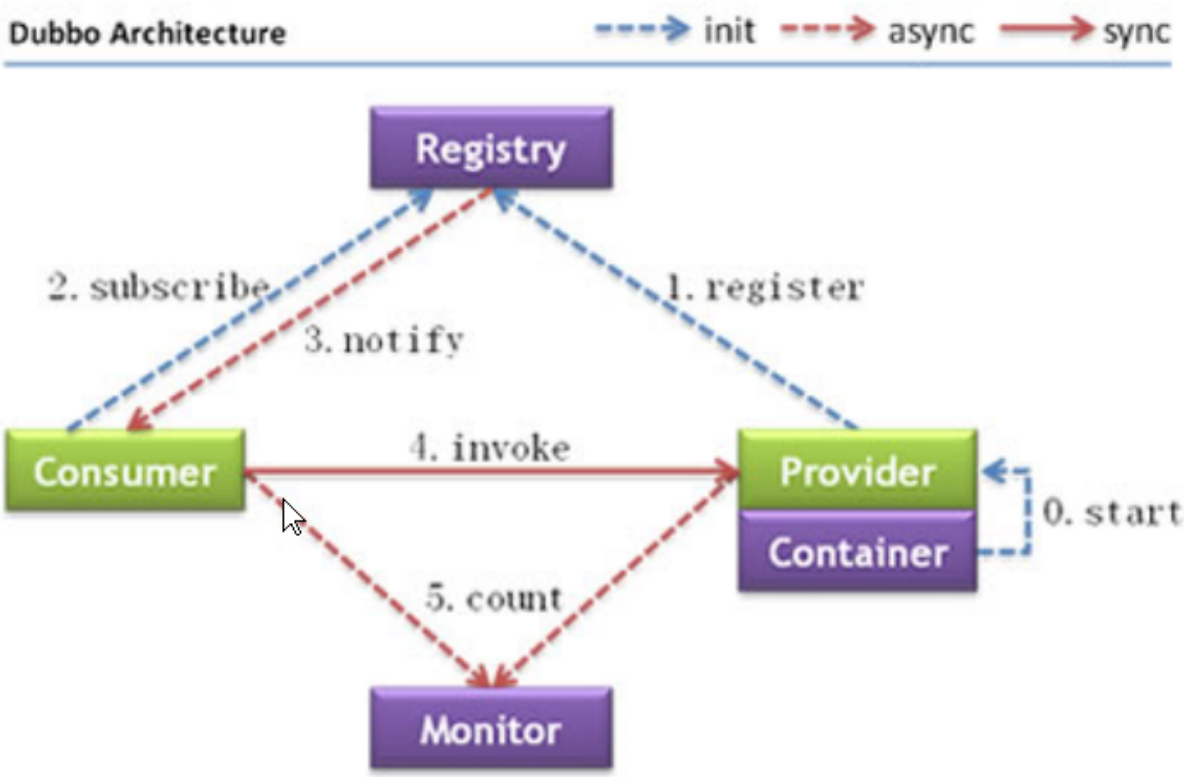
二，RPC

- 英文名称(Remote Procedure Call Protocol)
- 中文名称：远程过程调用协议
- RPC 解析：客户端(A)通过互联网调用远程服务器，不知道远程服务器具体实现，只知道远程服务器提供了什么功能
- RPC 最大优点

数据安全性

三，Dubbo简介

- Dubbo：一个分布式、高性能、透明化的 RPC 服务框架
- 作用：提供服务自动注册、自动发现等高效服务治理方案
- Dubbo 架构图：
  - Provider：提供者，服务发布方
  - Consumer：消费者，调用服务方
  - Container：Dubbo 容器，依赖于(基于) Spring 容器
  - Registry：注册中心，当 Container 启动时把所有可以提供的服务列表上 Registry 中进行注册
    作用:告诉 Consumer 提供了什么服务和服务方在哪里
  - Monitor：监听器，一般是做运维的人去关注(可以了解到项目的负载情况)
  - 虚线都是异步访问，实线都是同步访问
  - 蓝色虚线：在启动时完成的功能
  - 红色虚线(实线)都是程序运行过程中执行的功能
  - 所有的角色都是可以在单独的服务器上，所以必须遵守特定的协议



- 运行原理：
  1. 启动容器，相当于在启动 Dubbo 的 Provider
  2. 启动后会去注册中心进行注册，注册所有可以提供的服务列表

3. 在 Consumer 启动后会去 Registry 中获取服务列表和 Provider 的地址，进行订阅
4. 当 Provider 有修改后，注册中心会把消息推送给 Consummer

   使用了观察者设计模式(又叫发布/订阅设计模式)
5. 根据获取到的 Provider 地址，真实调用 Provider 中功能

   在 Consumer 方使用了代理设计模式，创建一个 Provider 方类的一个代理对象，通过代理对象获取 Provider 中真实功能，起到保护 Provider 真实功能的作用
6. Consumer 和 Provider 每隔 1 分钟向 Monitor 发送统计信息，统计信息包含，访问次数，频率等

四，Dubbo支持的注册中心

1. Zookeeper
   - 优点：支持网络集群
   - 缺点：稳定性受限于 Zookeeper
2. Redis
   - 优点：性能高
   - 缺点：对服务器环境要求较高
3. Multicast
   - 优点：面中心化，不需要额外安装软件
   - 缺点：建议同机房(局域网)内使用
4. Simple

   适用于测试环境.不支持集群

五，Zookeeper详解

- Zookeeper 分布式协调组件(本质是一个软件)
- Zookeeper 常用功能：
  1. 发布订阅功能，把 zookeeper 当作注册中心原因
  2. 分布式/集群管理功能
- 使用 java 语言编写的

六，Dubbo支持的协议

1. Dubbo
   - Dubbo 官方推荐的协议
   - 本质：使用 NIO 和线程池进行处理
   - 缺点：大文件传输时可能出现文件传输失败问题
2. RMI
   - JDK 提供的协议，远程方法调用协议

- 缺点：偶尔连接失败
- 优点：JDK 原生，不需要进行额外配置(导入 jar)

3. Hession
   - 优点：基于 http 协议，http 请求支持
   - 缺点：需要额外导入 jar ， 并在短连接时性能低

七，Registry搭建过程

即 zookeeper 单机版安装步骤

前提：已经配置好 JDK 环境变量

1. 上传 zookeeper 安装包到 linux 中 /usr/local/temp 中(目录随意，对安装无影响)
2. 解压 zookeeper 压缩包

   `tar zxvf /usr/local/temp/zookeeper-3.4.8.tar.gz`
3. 复制 zookeeper 解压后的文件夹到 /usr/local 下并起名为 zookpper (复制后名称任意，对安装无影响)

   `cp -r /usr/local/temp/zookeeper-3.4.8  /usr/local/zookeeper`
4. 进入到 zookeeper 文件夹中

   `cd /usr/local/zookeeper`
5. 在 zookeeper 中新建 data 文件夹，做为 zookeeper 数据存储文件夹

   `mkdir data`
6. 进入到 conf 文件夹

   `cd conf`
7. 复制 zoo_sample.cfg，并给新起名的zoo.cfg

   `cp zoo_sample.cfg zoo.cfg`
8. 修改 zoo.cfg 中 dataDir 属性值为新建 data 文件夹的路径

   `vim zoo.cfg`

   修改后的效果

```
# do not use /tmp for storage, /tmp here is ju
# example sakes.
dataDir=/usr/local/zookeeper/data
# the port at which the clients will connect
```

9. 进入到 zookeeper/bin 文件夹，使用 zkServer.sh 启动 zookeeper

   `cd ../bin`

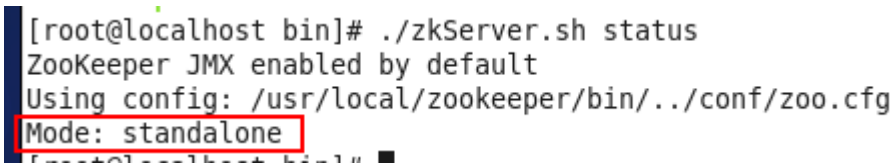
   `./zkServer.sh start`

   启动成功效果图

```
[root@localhost bin]# ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

10. 查看状态，其中 Mode：standalone 表示单机版

    `./zkServer.sh status`

11. 为了外部能访问，需要在防火墙中放行 2181 端口

八，Dubbo中Provider搭建

1. 新建 Maven Project，里面只有接口(dubbo-service)

   为什么这么做?

   RPC 框架，不希望 Consumer 知道具体实现，如果实现类和接口在同一个项目中，Consumer 依赖这个项目时，就会知道实现类具体实现，在注册过程中有实体类，实体类就必须要被序列化 `implements Serializable`

2. 新建 Maven Project，写接口的实现类(dubbo-service-impl)
3. 在 duboo-service-impl 中配置 pom.xml
   1. 依赖接口
   2. 依赖 dubbo ， 去掉老版本 spring
   3. 依赖新版本 spring
   4. 依赖 zookeeper 客户端工具 zkClient(为了访问操作 Zookeeper)

```
<dependencies>
    <dependency>
        <groupId>com.bjsxt</groupId>
        <artifactId>dubbo-service</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>dubbo</artifactId>
        <version>2.5.3</version>
        <exclusions>
            <!-- 去掉老版本的 spring -->
            <exclusion>
                <artifactId>spring</artifactId>
                <groupId>org.springframework</groupId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>4.1.6.RELEASE</version>
    </dependency>
    <!-- 访问zookeeper的客户端jar -->
```

```
    <dependency>
        <groupId>com.101tec</groupId>
        <artifactId>zkclient</artifactId>
        <version>0.10</version>
    </dependency>
</dependencies>
```

4. 新建实现类，并实现接口方法
5. 新建配置文件 applicationContext-dubbo.xml ， 并配置
    1. `<dubbo:application/>` 给 provider 起名，在 monitor 或管理工具中区别是哪个 provider
    2. `<dubbo:registry/>` 配置注册中心
        - address：注册中心的 ip 和端口
        - protocol：使用哪种注册中心
    3. `<dubbo:protocol/>` 配置协议
        - name：使用什么协议
        - port：consumer invoke provider 时的端口号
    4. `<dubbo:service/>` 注册接口

       ref 引用接口实现类 `<bean>` 的 id 值

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    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://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://code.alibabatech.com/schema/dubbo
        http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
    <!-- 给当前Provider自定义个名字 -->
    <dubbo:application name="dubbo-service"/>
    <!-- 配置注册中心  -->
    <dubbo:registry address="192.168.139.130:2181" protocol="zookeeper"></dubbo:registry>
    <!-- 配置端口 -->
    <dubbo:protocol name="dubbo" port="20888"></dubbo:protocol>
    <!-- 注册功能 -->
    <!-- 接口声明 -->
    <dubbo:service interface="com.bjsxt.service.DemoService" ref="demoServiceImpl">
</dubbo:service>
    <!-- 实现类 -->
    <!-- 因为 dubbo 基于 spring ，所以这也意味着这个类交给了 spring 容器管理 可以直接用注解调用 -
->
    <bean id="demoServiceImpl" class="com.bjsxt.service.impl.DemoServiceImpl"></bean>
    <!-- 如果 java 项目需要用到 spring 可以用这个方法加载 spring 配置文件 -->
    <!-- 意思是当这个文件被加载的时候，再引入 spring 文件加载 -->
    <!-- 因为 java 项目没有 web.xml 所以可以用这个方法，实现两个配置文件都被加载 -->
    <import resource="../../applicationContext.xml"/>

</beans>
```

6. 启动容器

    1. 通过 spring 方式启动

       applicationContext-dubbo.xml 位置没有要求，把下面的引用地址改为相应地址即可

```
ClassPathXmlApplicationContext ac = new ClassPathXmlApplicationContext("applicationContext-dubbo.xml");
ac.start();
System.out.println("启动成功");
System.in.read();
```

    2. 使用 dubbo 提供的方式启动(推荐使用这种方式)

       要求 applicationContext-dubbo.xml 必须放入类路径下 /META-INF/spring/*.xml (src/main/resources/META-INF/spring/*.xml) 和 webapp 下面的 /META-INF/ 不是同一个

```
Main.main(args);
```

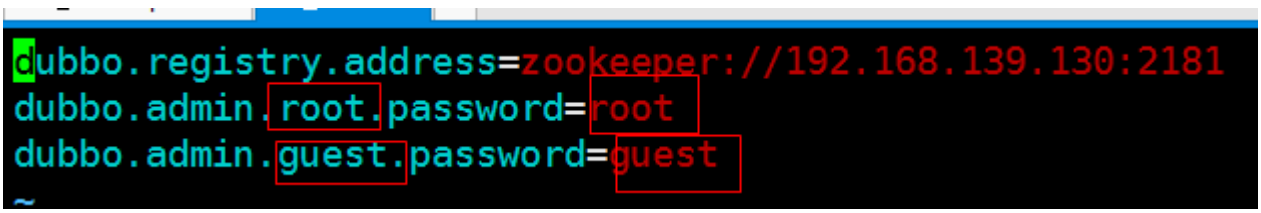
九，Admin管理界面

- 本质就是一个 web 项目
- 获取注册中心内 Provider 注册的信息，用页面呈现出来
- 实现步骤：

    1. 把 dubbo-admin-2.5.3.war 上传到服务器 tomcat 中
    2. 启动 tomcat 完成后关闭 tomcat，删除上传的 dubbo-admin-2.5.3.war

       为什么要删除：需要修改解压后的文件夹，如果不删除 .war 文件，下次重启 tomcat 会还原成未修改状态
    3. 进入 dubbo-admin-2.5.3/WEB-INF/dubbo.properties ，修改第一行为 zookeeper 的 ip 和端口

       第二行和第三行为管理界面的用户名和密码



4. 启动 tomcat，在浏览器地址栏访问 tomcat 中 dubbo 项目

十，Consumer搭建过程

1. 在 pom.xml 中除了 ssm 的依赖添加 dubbo 相关 3 个依赖(接口，dubbo.jar，zkClient)
2. web.xml 中修改 `<init-value>` applicationContext-*.xml

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext-*.xml</param-value>
</context-param>
```

3. spring 配置文件命名为 applicationContext-spring.xml，配置 dubbo 的配置文件 applicationContext-dubbo.xml

```
<!-- 给当前Provider自定义个名字 -->
<dubbo:application name="dubbo-consumer"/>
<!-- 配置注册中心  -->
<dubbo:registry address="192.168.139.130:2181" protocol="zookeeper"></dubbo:registry>
<!-- 配置注解扫描 -->
<dubbo:annotation package="com.bjsxt.service.impl"/>
```

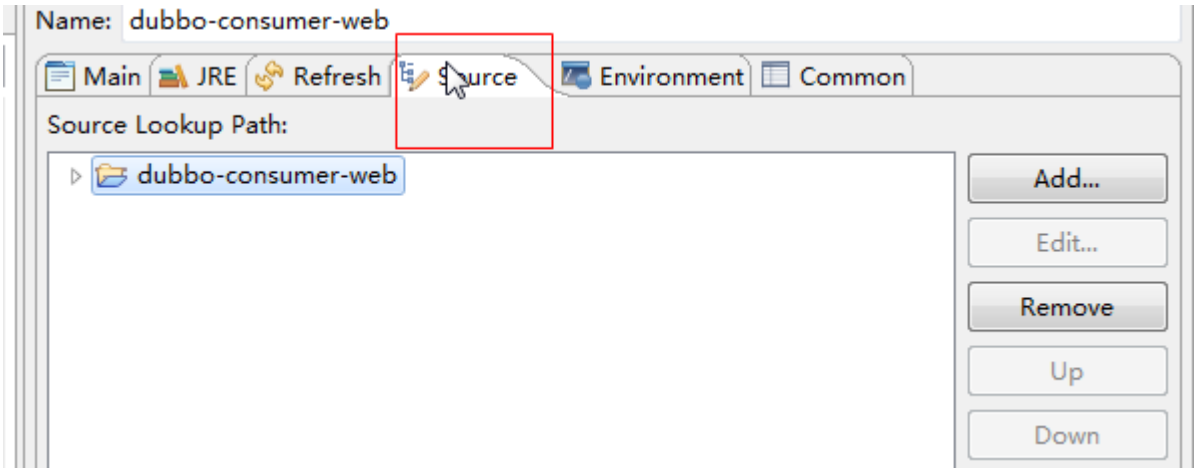
4. 不需要编写 mapper
5. 除了 ServiceImpl 中引用 Provider 中接口对象改变，其他代码都一样

```
@Service
public class TestServiceImpl implements TestService {
//  @Resource
//  private xxMapper xxxMapper;
    @Reference
    private DemoService demoService;
```

注意：有时电脑自带的的 WIFI 共享精灵在运行的时候，可能会连接不上 Zookeeper 或者是调用不了 Provider 的接口，把 WIFI 共享精灵关了就好了

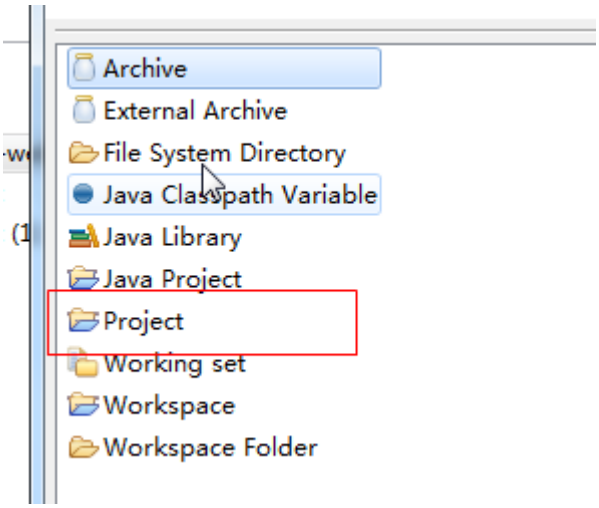
十一，Maven项目debug步骤

1. 右键项目 -> debug as -> 最下面 debug configuration
2. 选择 source



3. 删除默认 default

4. 点击 add



5. 选择对应的项目
6. 点击 debug

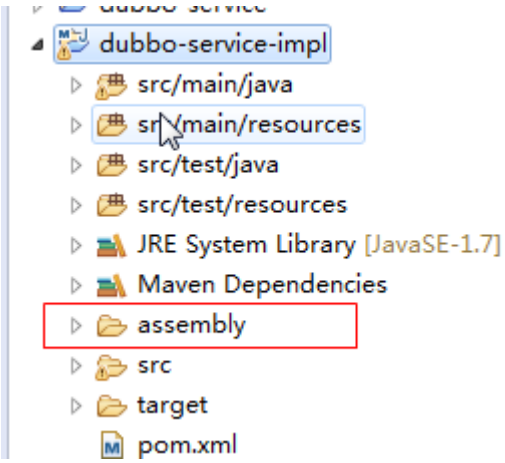
十二，Maven打包插件Assembly

解析：因为 Dubbo 项目的 Provider 项目一般都为 java(jar) 项目(tomcat 无法运行)，所以需要以这样的方式打包出去单独运行，为 Consumer 提供服务。

1. 在 dubbo 的 provider 项目(实现类项目)中 pom.xml 配置 assembly 插件信息

```
<!-- 指定项目的打包插件信息 -->
<plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
        <!-- 指定打包描述文件的位置：相对项目根目录的路径 -->
        <!-- assembly打包的描述文件 -->
        <descriptor>assembly/assembly.xml</descriptor>
    </configuration>
    <executions>
        <execution>
            <id>make-assembly</id>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
        </execution>
    </executions>
</plugin>
</plugins>
```

2. 在项目根目录下新建 assembly 文件夹



3. 在 assembly 文件夹中新建 assembly.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<assembly
    xmlns="http://maven.apache.org/plugins/maven-assembly-plugin/assembly/1.1.3"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/plugins/maven-assembly-
plugin/assembly/1.1.3 http://maven.apache.org/xsd/assembly-1.1.3.xsd">
    <!-- 该字符会添加到最终tar.gz包的名称后面，作为后缀 -->
    <id>assembly</id>
    <!-- 指定打包的格式为tar.gz，该类型压缩包在linux中比较常见 -->
    <formats>
        <format>tar.gz</format>
    </formats>
    <!-- 在tar.gz压缩包中是否包含根文件夹，该根文件夹名称和tar.gz去掉id后缀一致 -->
    <includeBaseDirectory>true</includeBaseDirectory>
    <fileSets>
        <!-- 将项目根路径下assembly/bin路径中的内容打包到压缩包中的根目录下的bin目录中 -->
        <fileSet>
            <!-- 相对项目根路径的相对路径 -->
            <directory>assembly/bin</directory>
            <outputDirectory>bin</outputDirectory>
            <!-- 设置最终tar.gz中该文件夹下的权限，跟linux权限写法一致 -->
            <fileMode>0755</fileMode>
        </fileSet>
        <!-- 将项目根路径下assembly/conf路径中的内容打包到压缩包中的根目录下的conf目录中 -->
        <fileSet>
            <directory>assembly/conf</directory>
            <outputDirectory>conf</outputDirectory>
            <!-- 设置其linux权限 -->
            <fileMode>0644</fileMode>
        </fileSet>
    </fileSets>
    <!-- 将所有依赖的jar包打包到压缩包中的根目录下的lib目录中 -->
    <!-- 此lib目录中包含自己开发的项目jar包以及demo_service.jar，还有第三方的jar包 -->
    <dependencySets>
        <dependencySet>
            <outputDirectory>lib</outputDirectory>
        </dependencySet>
    </dependencySets>
```

```
</assembly>
```

4. 解压下载的 dubbo-monitor-simple-2.5.3-assembly.tar.gz 压缩包,把解压后的 bin 和 conf 粘贴到项目下 assembly 文件夹中

   清空 conf/dubbo.properties 中内容

5. 右键项目 -> maven install

   在 target 下出现：项目名-版本-assembly.tar.gz 压缩包

6. 把压缩包复制到 windows 或 linux 中
   - window 中使用 start.bat 启动，关闭命令窗口关闭服务
   - linux 中使用 start.sh 启动使用 stop.sh 关闭