[今日课程大纲]

Dubbo 简介及结构图

Dubbo 支持的几种 Registry

Zookeeper 简介

Zookeeper 安装

Dubbo 支持的几种 Protocol

Dubbo 项目搭建

Admin 管理界面

Assembly 打包

监控中心搭建

[知识点详解]

一.SOA

1.英文名称(Service Oriented Ambiguity)

2.中文名称:面向服务架构

2.1 有一个专门提供服务单元.

2.2 其他所有单元都调用这个服务.

3.SOA 定位:

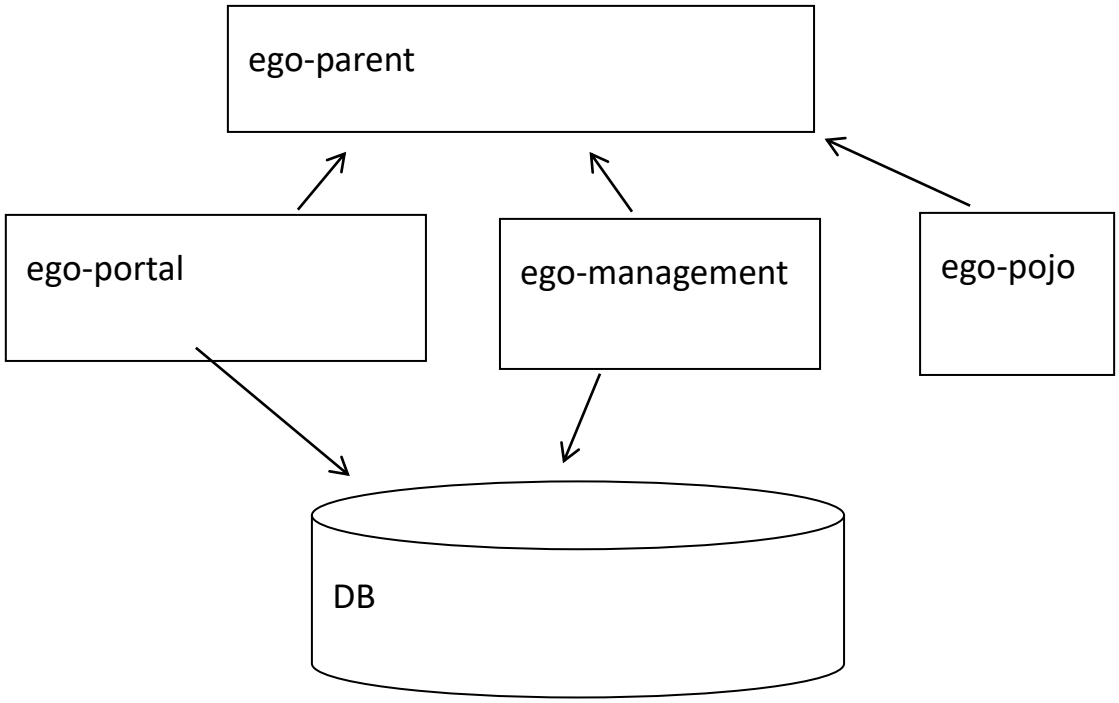
3.1 如何设计项目,让开发时更有效率.

3.2 SOA 是一种思想

4.之前项目架构设计

4.1 在公司项目不允许所有项目都访问数据库.

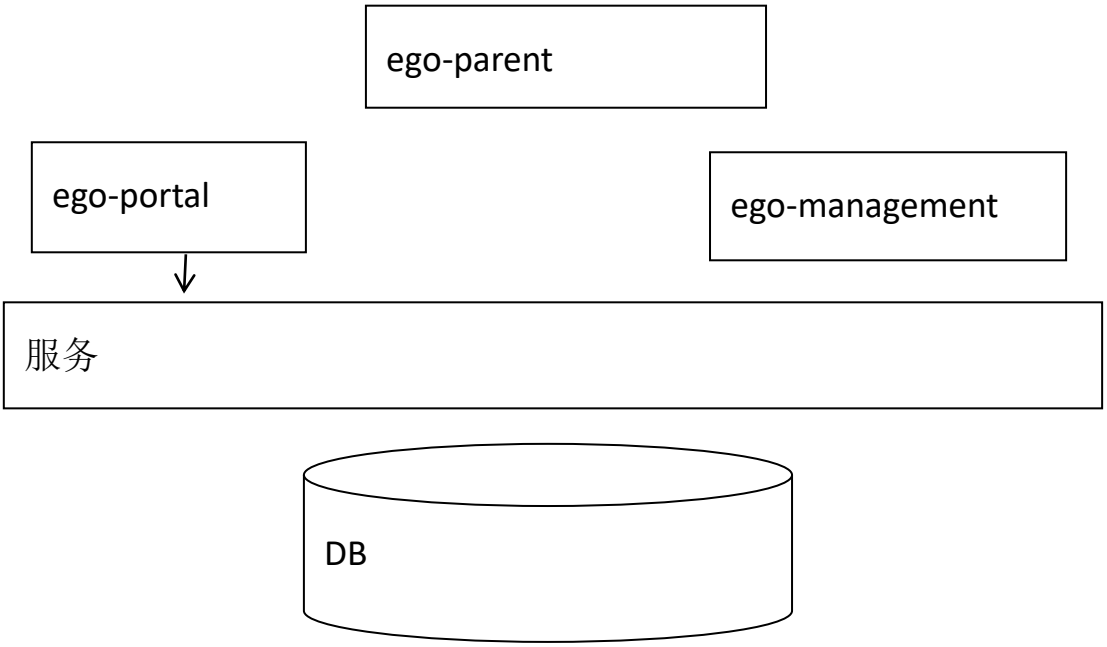
4.2 开发时,数据库访问层代码可能出现冗余



5.使用 SOA 架构

5.1 专门访问数据库**服务(项目)**.

5.2 开发时可以实现,数据访问控制和代码复用.



6.实现 SOA 架构时,常用服务.

6.1 Dubbo 做为服务.

6.2 WebService 做为服务.

6.3 Dubbox 做为服务.

6.4 服务方就是 web 项目,调用 web 项目的控制器.

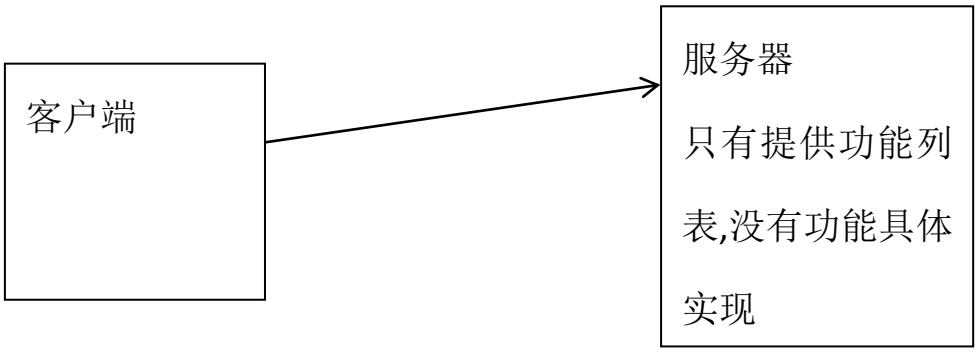
6.4.1 使用 HttpClient 可以调用其他项目的控制器.

二. RPC

1. 英文名称(Remote Procedure Call Protocol)

2. 中文名称:远程过程调用协议

3. 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.



4. RPC 最大优点:

4.1 数据安全性.

三.Dubbo 简介

1. Dubbo:一个分布式、高性能、透明化的 RPC 服务框架

2. 作用:提供服务自动注册、自动发现等高效服务治理方案.

3. Dubbo 架构图

3.1 Provider :提供者,服务发布方.

3.2 Consumer:消费者, 调用服务方

3.3 Container:Dubbo 容器.依赖于 Spring 容器.

3.4 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.

3.4.1 作用:告诉 Consumer 提供了什么服务和服务方在哪里.

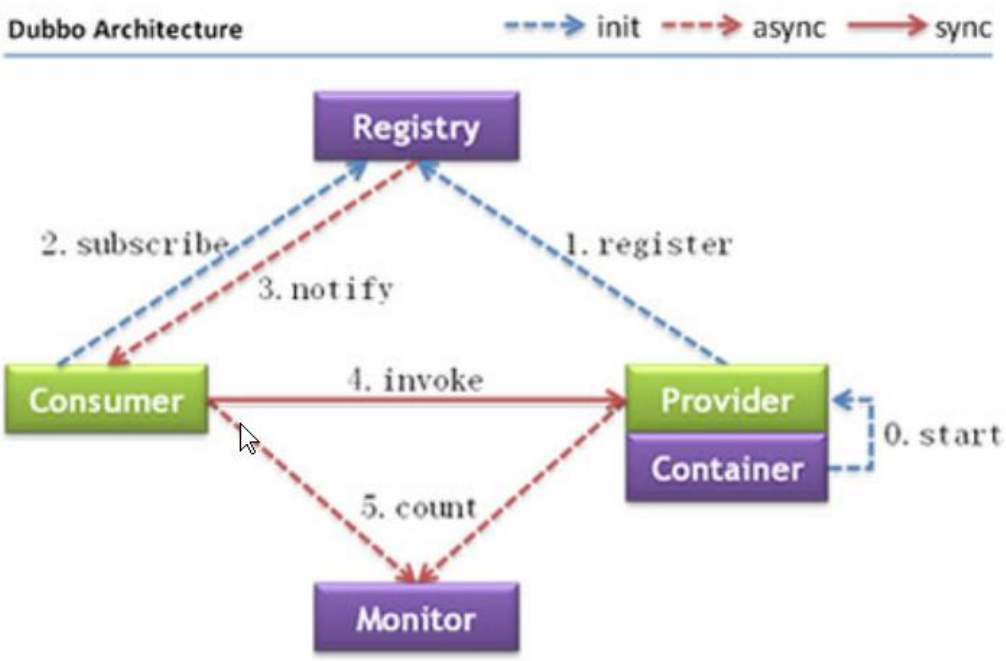
3.5 Monitor:监听器

3.6 虚线都是异步访问,实线都是同步访问

3.7 蓝色虚线:在启动时完成的功能

3.8 红色虚线(实线)都是程序运行过程中执行的功能

3.9 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.



4. 运行原理:

4.0 启动容器,相当于在启动 Dubbo 的 Provider

4.1 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

4.2 在 Consumer 启动后会去 Registry 中获取服务列表和 Provider 的地址.进行订阅.

4.3 当 Provider 有修改后,注册中心会把消息推送给 Consummer

4.3.1 使用了观察者设计模式(又叫发布/订阅设计模式)

4.4 根据获取到的 Provider 地址,真实调用 Provider 中功能.

4.4.1 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.

4.5 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

    1.1 优点:支持网络集群

    1.2 缺点:稳定性受限于 Zookeeper

2. Redis

    2.1 优点:性能高.

    2.2 缺点:对服务器环境要求较高.

3. Multicast

    3.1 优点:面中心化,不需要额外安装软件.

    3.2 缺点:建议同机房(局域网)内使用

4. Simple

    4.1 适用于测试环境.不支持集群.

五.Zookeeper 讲解

1. Zookeeper 分布式协调组件.

    1.1 本质一个软件.

2. Zookeeper 常用功能

    2.1 发布订阅功能.把 zookeeper 当作注册中心原因.

    2.2 分布式/集群管理功能.

3. 使用 java 语言编写的.

六. Dubbo 支持的协议

1.Dubbo

1.1 Dubbo 官方推荐的协议.

1.2 本质:使用 NIO 和线程池进行处理.

1.3 缺点:大文件传输时可能出现文件传输失败问题.

2.RMI

2.1 JDK 提供的协议,远程方法调用协议.

2.2 缺点:偶尔连接失败.

2.3 优点:JDK 原生,不需要进行额外配置(导入 jar)

3.Hession

3.1 优点:基于 http 协议,http 请求支持.

3.2 缺点:需要额外导入 jar,并在短连接时性能低

七.Dubbo 中 Provider 搭建

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

1.1 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.

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

3.1 依赖接口

3.2 依赖 dubbo,去掉老版本 spring

3.3 依赖新版本 spring

3.4 依赖 zookeeper 客户端工具 zkClient

```
<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>
            <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

5.1 <dubbo:application/> 给 provider 起名,在 monitor 或管理工具中区别是哪个 provider

5.2 <dubbo:registry/> 配置注册中心

5.2.1 address:注册中心的 ip 和端口

5.2.2 protocol 使用哪种注册中心

5.3 <dubbo:protocol/> 配置协议

5.3.1 name 使用什么协议

5.3.2 port: consumer invoke provider 时的端口号

5.4 <dubbo:service/> 注册接口

5.4.1 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-instanc
e"
   xmlns:context="http://www.springframework.org/schem
a/context"
   xmlns:dubbo="http://code.alibabatech.com/schema/dub
bo"
   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/sprin
g-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 -->
    <dubbo:service
interface="com.bjsxt.service.DemoService"
ref="demoServiceImpl"></dubbo:service>
    <bean id="demoServiceImpl"
class="com.bjsxt.service.impl.DemoServiceImpl"></bean
>
</beans>
```

6. 启动容器

6.1 通过 spring 方式启动

6.1.1 applicationContext-dubbo.xml 位置没有要求

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

```
        System.in.read();
```

6.2 使用 dubbo 提供的方式启动(推荐使用这种方式)

6.2.1 要求 applicationContext-dubbo.xml 必须放入类路径下 /META-INF/spring/*.xml

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

八. Admin 管理界面

1. 本质就是一个 web 项目

2. 获取注册中心内 Provider 注册的信息.用页面呈现出来.

3. 实现步骤

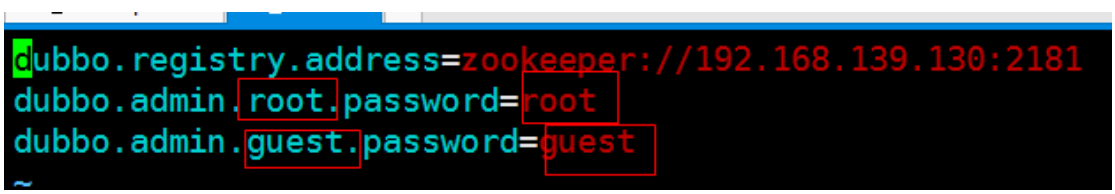
3.1 把 dubbo-admin-2.5.3.war 上传到服务器 tomcat 中.

3.2 启动 tomcat 完成后关闭 tomcat, 删除上传的 dubbo-admin-2.5.3.war

3.2.1 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

3.3 进入 dubbo-admin-2.5.3/WEB-INF/dubbo.properties,修改第一行为 zookeeper 的 ip 和端口

3.3.2 第二行和第三行为管理界面的用户名和密码



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

九.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

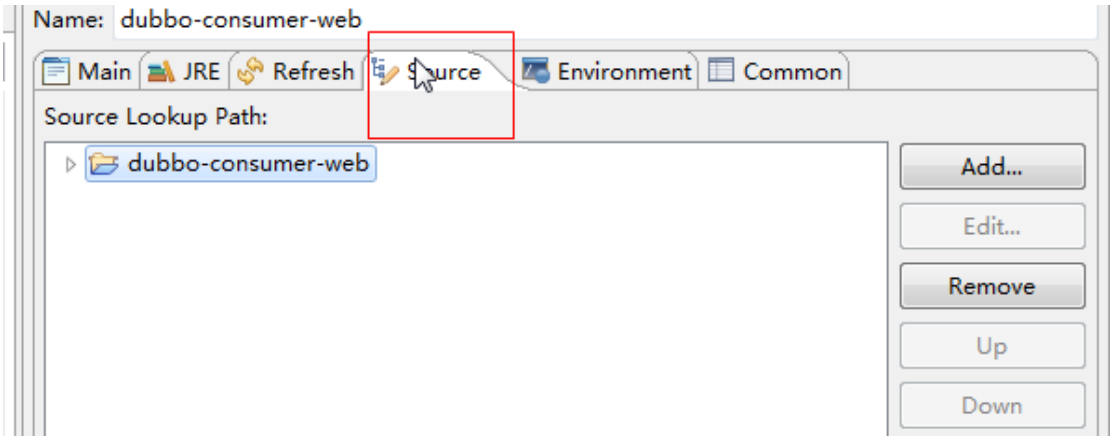
5. 除了 ServiceImpl 中引用 Provider 中接口对象改变,其他代码都一样.

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

十. Maven 项目 debug 步骤

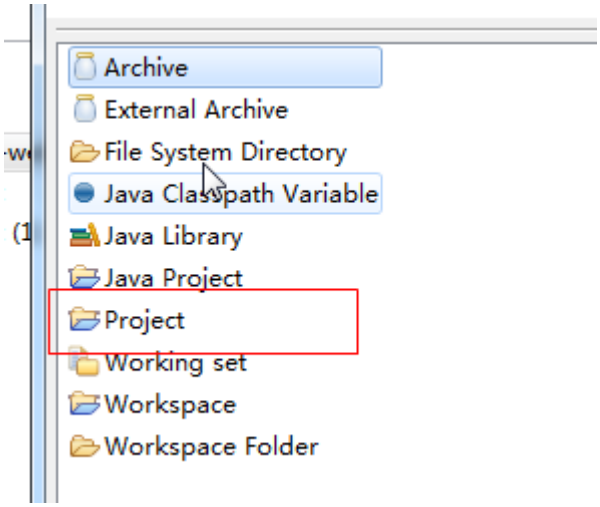
1. 右键项目 --> debug as --> 最下面 debug configuration

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

十一. Maven 打包插件 Assembly

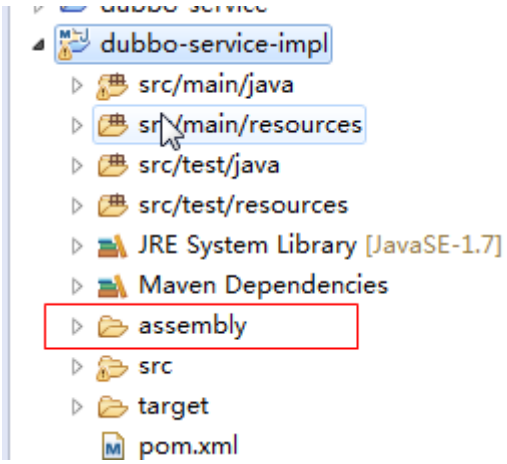
1. 在 dubbo 的 provider 项目(实现类项目)中 pom.xml 配置 assembly 插件信息

```
<!-- 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.

4.1 清空 conf/dubbo.properties 中内容.

5. 右键项目--> maven install

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

6. 把压缩包复制到 window 或 linux 中

6.1 window 中使用 start.bat 启动,关闭命令窗口关闭服务.

6.2 linux 中使用 start.sh 启动使用 stop.sh 关闭.