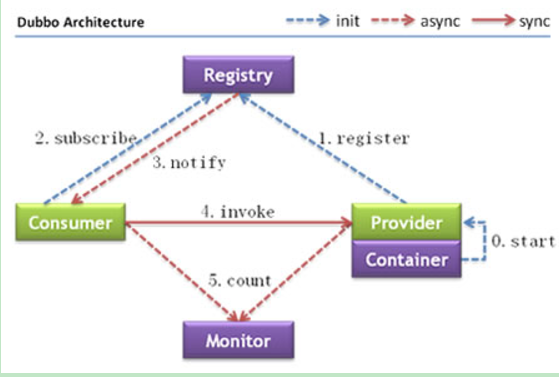
1.1Dubbo架构



各个角色的说明:

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：

服务容器负责启动，加载，运行服务提供者。

服务提供者在启动时，向注册中心注册自己提供的服务。

服务消费者在启动时，向注册中心订阅自己所需的服务。

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。

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。

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。

(1) 连通性： 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 监控中心负责统计各服务调用次数，调用时间等，统计先在内存汇总后每分钟一次发送到监控中心服务器，并以报表展示 服务提供者向注册中心注册其提供的服务，并汇报调用时间到监控中心，此时间不包含网络开销 服务消费者向注册中心获取服务提供者地址列表，并根据负载算法直接调用提供者，同时汇报调用时间到监控中心，此时间包含网络开销 注册中心，服务提供者，服务消费者三者之间均为长连接，监控中心除外 注册中心通过长连接感知服务提供者的存在，服务提供者宕机，注册中心将立即推送事件通知消费者 注册中心和监控中心全部宕机，不影响已运行的提供者和消费者，消费者在本地缓存了提供者列表 注册中心和监控中心都是可选的，服务消费者可以直连服务提供者

(2) 健状性：

1)监控中心宕掉不影响使用，只是丢失部分采样数据

2)数据库宕掉后，注册中心仍能通过缓存提供服务列表查询，但不能注册新服务

3)注册中心对等集群，任意一台宕掉后，将自动切换到另一台

4)注册中心全部宕掉后，服务提供者和服务消费者仍能通过本地缓存通讯

5)服务提供者无状态，任意一台宕掉后，不影响使用

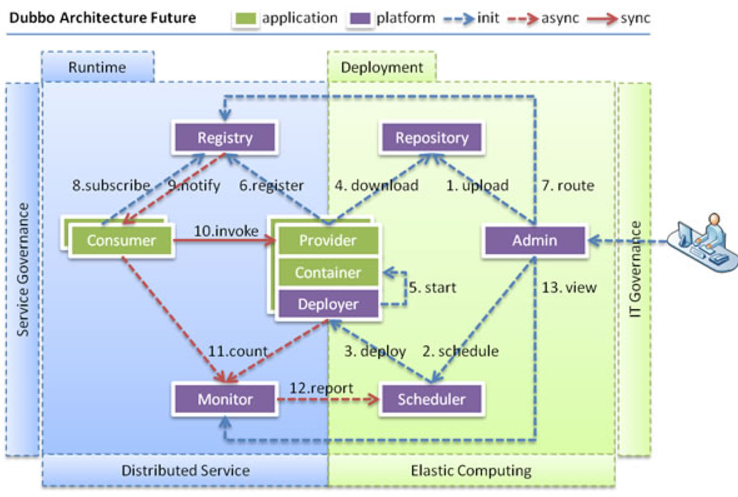
6)服务提供者全部宕掉后，服务消费者应用将无法使用，并无限次重连等待服务提供者恢复

(3) 伸缩性：

1)注册中心为对等集群，可动态增加机器部署实例，所有客户端将自动发现新的注册中心

2)服务提供者无状态，可动态增加机器部署实例，注册中心将推送新的服务提供者信息给消费者.

(4) 升级性： 当服务集群规模进一步扩大，带动IT治理结构进一步升级，需要实现动态部署，进行流动计算，现有分布式服务架构不会带来阻力：



Deployer: 自动部署服务的本地代理。

Repository: 仓库用于存储服务应用发布包。

Scheduler: 调度中心基于访问压力自动增减服务提供者。

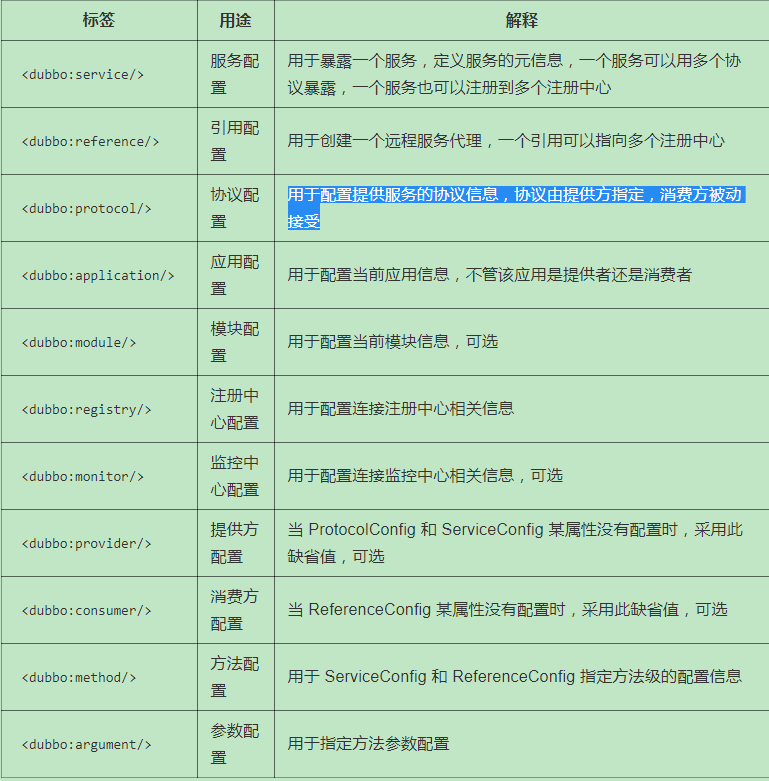
Admin: 统一管理控制台。

1.2dubbo用法

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。

Dubbo+Zooker的搭建

dubbo 的xml配置文件的解析

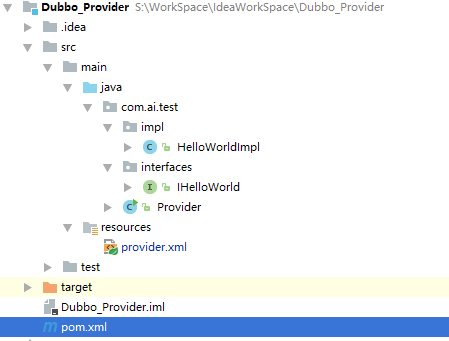


**简单的HelloWorld**

1. **provider方的配置**

前提是已经在虚机中起了zookeeper集群服务.

项目结构:



provider实现类:

**package com.ai.test**;  
  
***/\*\*  
 \* Created by LYF on 2017/9/16.  
 \*/*import org.springframework.context.support.ClassPathXmlApplicationContext**;  
  
**public class Provider** {  
 **public static void** main(**String**[] args) **throws Exception** {  
 **ClassPathXmlApplicationContext** context = **new** ClassPathXmlApplicationContext(  
 **new** String[] {**"provider.xml"**});  
 context.***start***();  
 **System**.***out***.***println***(**"dubbo已经启动了~~~~~~~~~~~`"**);  
 **System**.***in***.***read***(); ***// press any key to exit*** }  
}

provider.xml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">  
 <dubbo:application name="hello-world-app"/>***<!--  
 <dubbo:registry address="multicast://224.5.6.7:1234"/>  
-->* **<dubbo:registry address="zookeeper://192.168.85.2:2183?backup=192.168.85.3:2183,192.168.85.4:2183"/>  
 <dubbo:protocol name="dubbo" port="20880"/>  
 <dubbo:service interface="com.ai.test.interfaces.IHelloWorld" ref="helloremote"/>  
 <bean id="helloremote" class="com.ai.test.impl.HelloWorldImpl"/>  
  
</beans>**

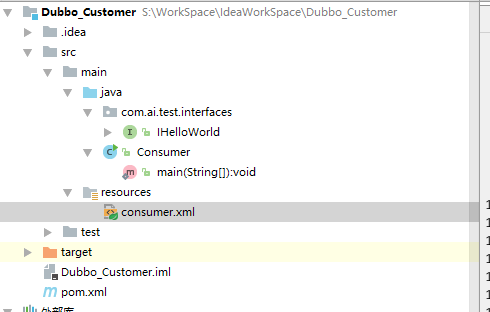
pom.xml

*<?***xml version="1.0" encoding="UTF-8"***?>***<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>**com.ai.test**</groupId>  
 <artifactId>**Dubbo\_Provider**</artifactId>  
 <version>**1.0-SNAPSHOT**</version>  
 <dependencies>  
 <dependency>  
 <groupId>**com.alibaba**</groupId>  
 <artifactId>**dubbo**</artifactId>  
 <version>**2.5.5**</version>  
 </dependency>  
 <dependency>  
 <groupId>**org.apache.zookeeper**</groupId>  
 <artifactId>**zookeeper**</artifactId>  
 <version>**3.3.3**</version>  
 </dependency>  
 <dependency>  
 <groupId>**com.101tec**</groupId>  
 <artifactId>**zkclient**</artifactId>  
 <version>**0.4**</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>**org.apache.zookeeper**</artifactId>  
 <groupId>**zookeeper**</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
  
 </dependencies>  
  
  
  
</project>**

1. **consumer方的配置:**

服务是基于接口调用的,所以服务方和消费方都应该有同样的接口,消费方才可以调用服务提供方..

项目结构:



consumer实现类:

**import com.ai.test.interfaces.**IHelloWorld;  
**import org.springframework.context.support.ClassPathXmlApplicationContext**;  
  
**public class Consumer** {  
 **public static void** main(**String**[] args) **throws Exception** {  
 **ClassPathXmlApplicationContext** context = **new** ClassPathXmlApplicationContext(  
 **new** String[]{**"consumer.xml"**});  
 context.***start***();  
 IHelloWorld demoService = (IHelloWorld) context.***getBean***(**"hhhh"**); ***// obtain proxy object for remote invocation* String** hello = demoService.***sayHello***(**"world"**); ***// execute remote invocation* System**.***out***.***println***(hello); ***// show the result*** }  
}

pom.xml

*<?***xml version="1.0" encoding="UTF-8"***?>***<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>**com.ai.test**</groupId>  
 <artifactId>**Dubbo\_Customer**</artifactId>  
 <version>**1.0-SNAPSHOT**</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>**com.alibaba**</groupId>  
 <artifactId>**dubbo**</artifactId>  
 <version>**2.5.5**</version>  
 </dependency>  
 <dependency>  
 <groupId>**org.apache.zookeeper**</groupId>  
 <artifactId>**zookeeper**</artifactId>  
 <version>**3.3.3**</version>  
 </dependency>  
 <dependency>  
 <groupId>**com.101tec**</groupId>  
 <artifactId>**zkclient**</artifactId>  
 <version>**0.4**</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>**org.apache.zookeeper**</artifactId>  
 <groupId>**zookeeper**</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 </dependencies>  
  
</project>**

consumer.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="demo-consumer"/>  
 <dubbo:registry address="zookeeper://192.168.85.2:2183?backup=192.168.85.3:2183,192.168.85.4:2183"/>***<!--  
 <dubbo:registry address="multicast://224.5.6.7:1234"/>  
-->* **<dubbo:reference id="hhhh" interface="com.ai.test.interfaces.IHelloWorld"/>  
</beans>**