**[知识点总揽]**

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]**

# 项目架构分析

1. 项目架构

1.1 前台和后台在代码中出现大量重复代码

1.2 很多项目都可以直接操作数据库,而数据是企业的核心.不应该让所有人都随意访问.

1.3 随着平台的增加,增加了Android,IOS等,添加和前台项目和后台项目同级别的项目,项目越多,重复代码越多.

父项目

前台项目

后台项目

1. 优化项目架构.

访问数据库,对外提供接口(服务)

父项目

前台

后台

Android

1. SOA

3.1 英文名称:(Service-Oriented Architecture)

3.2 中文名称:面向服务架构.

3.2.1 面向:围绕发布服务项目

3.2.2 里面”服务”:对外提供接口(服务)的项目

3.3 SOA是一种架构,只要项目结构中有一个对外发布服务的项目,并且其他项目调用这个项目的服务,这种架构设计就是SOA架构

1. RPC

4.1 英文名称(Remote Procedure Call Protocol)

4.2 中文名称:远程过程调用协议.

4.3 解释:有一台计算机发布服务后,其他计算机通过网络访问这个计算机发布的服务.**其他计算机只能使用服务,但是不知道服务的具体实现过程.**

1. 目前可能使用的发布服务(对外提供接口的几种方式)

5.1 让其他项目直接访问控制器.

5.2 Web-Service

5.3 Dubbo

# Dubbo简介

1. 官网地址

|  |
| --- |
| http://dubbo.io/ |

1. 解释: 提供**高性能**,透明化的RPC框架.
2. Dubbo架构

3.1 Provider:提供者.发布服务的项目.

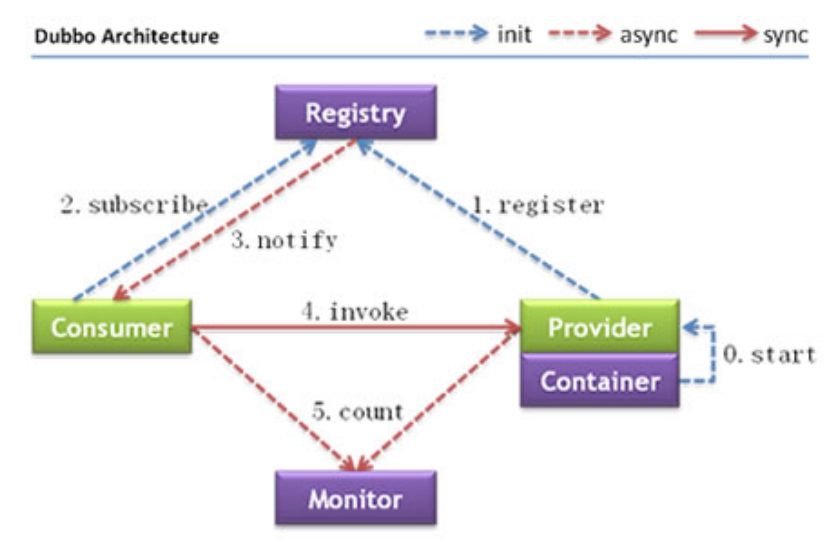
3.2 Registry:注册中心.所有提供者必须去注册中心注册自己所有能发布的服务.

3.3 Consumer:消费者.调用服务的项目.

3.4 Monitor:监控中心.监控消费者和提供者调用服务的时间及次数.默认每1分钟向监控中心生成一次统计数据.

3.5 之间调用必须遵守Dubbo支持的协议.

3.6 container:容器.Dubbo依赖于Spring容器.



1. 执行顺序.

4.0:由Spring容器启动服务.

4.1 向注册中心注册服务.

4.2 消费者向注册中心订阅需要调用的服务.在注册中心的服务列表中寻找需要调用的服务.获取到提供者真实地址.

4.3 注册中心通知消费提供者的真实地址.如果提供者的服务发生变化,注册中心会自动推送信息给消费者.

4.4 消费者调用提供者的服务.

4.5 在调用过程中向监控中心发送数据,进行统计调用时间和调用次数.

4.6 虚线都是异步请求,实线都是同步请求.

4.6.1 只用真实调用服务时是同步,最花费时间.

# Dubbo支持的注册中心

1. Zookeeper注册中心

1.1 优点:支持集群.

1.2 缺点:稳定性受Zookeeper影响.

1. Redis注册中心

1.1 优点:基于服务器双写模式.性能高.

1.2 缺点:要求服务器时间必须一致.

1. Multicast注册中心

3.1 免中心化的注册中心,不需要额外安装软件.

3.2 不支持集群.

1. Simple注册中心

4.1 适用于测试环境的.

# 四.Dubbo支持的几种协议

1. Dubbo

1.1 基于NIO的协议.性能比较高

1.2 缺点:在单一连接时,传输大文件可能出错.

1. RMI

2.1 RMI是JDK提供的.可以和原生RMI互用.

2.2 缺点:可能出现连接无效情况.

1. Hession

3.1 第三方的Hession.jar

3.2 缺点:短连接性能低.

# Zookeeper简介

1. 分布式协调组件(就是一个软件.)
2. 好处:

2.1 高效性

2.2 可扩展性.

1. 作用:

3.1 **服务发布和订阅(注册使用这个特性)**

3.2 集群管理功能(SolrCloud使用这个特性)

3.3 负载均衡

3.4 分布式锁

1. 使用java语言编写.

# 使用Maven发布服务的步骤

1. 创建dubbo-service项目,只写接口
2. 创建dubbo-service-impl项目,需要依赖dubbo-service
3. 在pom.xml配置,导入dubbo.jar和zkClient.jar

3.1 <exclusions>不导入哪个依赖包

3.2 spring-context:我们使用的spring版本

3.3 zkClient:连接Zookeeper的客户端工具包

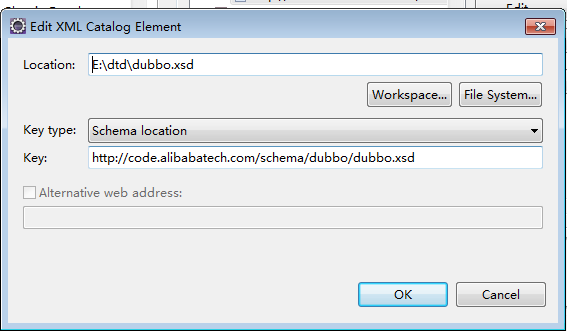
3.4 dubbo: dubbo框架的包

|  |
| --- |
| <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-context</artifactId>  <version>4.1.6.RELEASE</version>  </dependency>  <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>0.10</version>  </dependency>  </dependencies> |

1. 在resources下新建spring配置文件

4.1 如果<dubbo:>没有提示

4.1.1 window-preferences-->xml--> xml catalog-->add



4.1.2 dubbo.xsd在dubbo-2.5.3-sources.jar中

4.2 <beans>可以在dubbo-master.zip\dubbo-master\ dubbo-admin\ src\main\resources\META-INF\spring里面复制

|  |
| --- |
| <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"*>  <!-- 在监控中心/admin管理界面中看是哪个项目 -->  <dubbo:application name=*"dubbo-service-impl"*/>  <!-- 配置注册中心 -->  <dubbo:registry protocol=*"zookeeper"* address=*"192.168.92.130:2181"*></dubbo:registry>  <!-- 配置协议 -->  <dubbo:protocol name=*"dubbo"* port=*"20880"*></dubbo:protocol>  <!-- 注解扫描 -->  <dubbo:annotation package=*"com.bjsxt.service.impl"*/>  </beans> |

1. 编写服务实现类

5.1 @Service是Dubbo的注解

|  |
| --- |
| import com.alibaba.dubbo.config.annotation.Service;  **import** com.bjsxt.service.DemoService;  **@Service**  **public** **class** DemoServiceImpl **implements** DemoService {  @Override  **public** String test(String name) {  **return** "传递过来的内容:"+name;  }  } |

1. 启动容器,发布服务

6.1 直接启动Spring配置文件

6.2 start()方法在ApplicationContext接口中没有

|  |
| --- |
| **public** **class** Test {  @SuppressWarnings("resource")  **public** **static** **void** main(String[] args) **throws** IOException {  ClassPathXmlApplicationContext ac = **new** ClassPathXmlApplicationContext("applicationContext-dubbo.xml");  ac.start();  System.***in***.read();  }  } |

6.2使用Dubbo提供的Main类进行启动(**推荐方式**)

6.2.1 Main类会加载classpath/META-INF/spring/\*.xml

|  |
| --- |
| **public** **class** Test {  @SuppressWarnings("resource")  **public** **static** **void** main(String[] args) **throws** IOException {  Main.*main*(args);  }  } |

# 使用Maven创建Consummer项目

1. 保证Consummer项目中服务方接口

1.1 需要知道服务方接口名称及方法名

1. 创建Maven projec,配置pom.xml

2.1 依赖的dubbo-service可以从svn上下载.

|  |
| --- |
| <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-context</artifactId>  <version>4.1.6.RELEASE</version>  </dependency>  <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>0.10</version>  </dependency>  </dependencies> |

1. 编写spirng配置文件

3.1 application需要自定义名称

3.2 扫描自己项目中ServiceImpl的@Reference注解

|  |
| --- |
| <!-- 在监控中心/admin管理界面中看是哪个项目 -->  <dubbo:application name=*"dubbo-consummer"*/>  <!-- 配置注册中心 -->  <dubbo:registry protocol=*"zookeeper"* address=*"192.168.92.130:2181"*></dubbo:registry>  <!-- 配置协议 -->  <dubbo:protocol name=*"dubbo"* port=*"20880"*></dubbo:protocol>  <!-- 注解扫描,扫描@Reference注解 -->  <dubbo:annotation package=*"com.bjsxt.service.impl"*/> |

1. 编写Service实现类代码

4.1 @Service 把类交给Spring容器

4.2 @Reference给接口注入对象.对象信息从注册中心获取.

|  |
| --- |
| import org.springframework.stereotype.Service;  **import** com.alibaba.dubbo.config.annotation.Reference;  **import** com.bjsxt.service.DemoService;  **import** com.bjsxt.service.MyService;  **@Service**  **public** **class** MyServiceImpl **implements** MyService {  **@Reference**  **private** DemoService demoService;  @Override  **public** **void** test() {  String result = demoService.test("张三");  System.***out***.println(result);  }  } |

1. 编写测试代码

|  |
| --- |
| **public** **class** Test {  @SuppressWarnings("resource")  **public** **static** **void** main(String[] args) {  ApplicationContext ac = **new** ClassPathXmlApplicationContext("applicationContext-consummer.xml");  MyServiceImpl bean = ac.getBean("myServiceImpl", MyServiceImpl.**class**);  bean.test();  }  } |

# Admin管理界面

1. 本质:web项目,war打包项目
2. 直接把dubbo-admin-2.5.3放入到tomcat(linux/window都可以)中
3. 启动tomcat.
4. 关闭tomcat.
5. 删除dubbo-admin-2.5.3.war

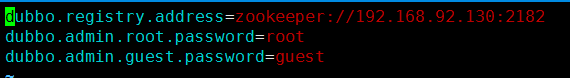
5.1 防止下次启动tomcat后重新解压,导致修改内容无效

1. 进入解压后文件夹dubbo-admin-2.5.3/WEB-INF/中修改dubbo.properties文件

6.1 第一行为注册中心地址

6.2 第二行是管理员用户名和密码

6.3 第三行是游客的用户名个和密码



# Assembly打包插件

1. 打包目的:为了部署服务更加方便.
2. 步骤:

2.1 在Provider项目根目录创建assembly文件夹

2.2 dubbo-monitor-simple-2.5.3-assembly.tar.gz 解压后,把bin和conf粘贴到项目中assembly文件夹中

2.3 清空assembly/dubbo.properties中内容

2.4 在pom.xml中配置

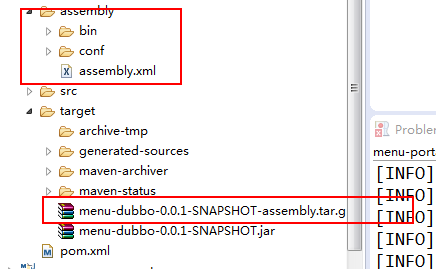
|  |
| --- |
| <build>  <plugins>  <!-- 指定项目的打包插件信息 -->  <plugin>  <artifactId>maven-assembly-plugin</artifactId>  <configuration>  <!-- 指定打包描述文件的位置：相对项目根目录的路径 -->  <!-- assembly打包的描述文件 -->  <descriptor>**assembly/assembly.xml**</descriptor>  </configuration>  <executions>  <execution>  <id>make-assembly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

2.5 在项目的assembly文件夹下新建assembly.xml并粘贴下面内容

|  |
| --- |
| <?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> |

2.6 右键项目run as --> Maven install

2.7 配置后项目结构



1. 把生成的tar.gz文件粘贴到window或linux中使用start文件运行

3.1 window系统只有start.bat文件.关闭文件关闭服务.

3.2 部署到linux后注意

3.2.1 start.sh 启动文件

3.2.2 stop.sh 关闭文件

3.2.3 运行之前需要配置JDK

3.2.4 要让防火墙放行自己设定的端口

# 监控中心

1. dubbo-monitor-simple-2.5.3-assembly.tar.gz 解压到任意非中文目录下.
2. 配置解压文件夹dubbo-monitor-simple-2.5.3\conf/dubbo.properties

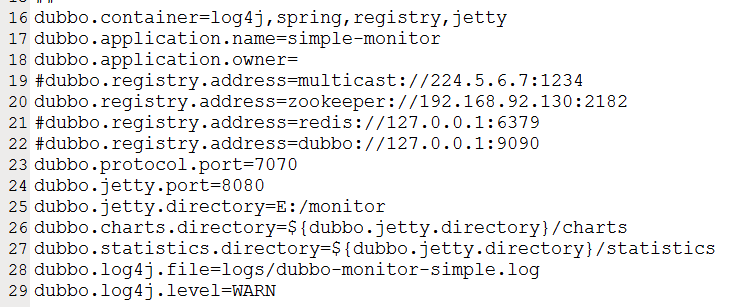
2.1 19到22行是配置注册中心,一定要修改成自己的注册中心

2.2 dubbo.protocol.port=7070 消费者访问这个服务的端口

2.3 dubbo.jetty.port=8080 浏览器访问时端口

2.4 25行一定要配置成正确路径,monitor文件夹不会自动被创建,手动创建

2.5 26和27行的charts和statistics文件夹可以自动被创建



1. 进入bin文件启动服务.
2. 在浏览器中输入http://ip:8080访问服务端.
3. 注意:提供者和消费者的配置文件中需要配置

|  |
| --- |
| dubbo:monitor protocol=*"registry"*></dubbo:monitor> |

1. 在测试时,要注意,只有等消费者结束访问,监控中心才会生成对应的数据.
2. 监控中心生成的数据默认每分钟一次.

# 十一.Dubbo中@Service注解和SpringAOP冲突问题

1. 解决办法:不使用注解方式完成,回归使用XML方式

1.1 <dubbo:service>配置接口

1.2 <bean>配置接口实现类

1.3 每个接口和实现类都需要配一遍

|  |
| --- |
| <dubbo:service interface=*"com.bjsxt.rpc.service.MenuRpcService"* ref=*"menuRpcServiceImpl"*></dubbo:service>  <bean id=*"menuRpcServiceImpl"* class=*"com.bjsxt.rpc.service.impl.MenuRpcServiceImpl"*></bean> |