Dubbo项目实战

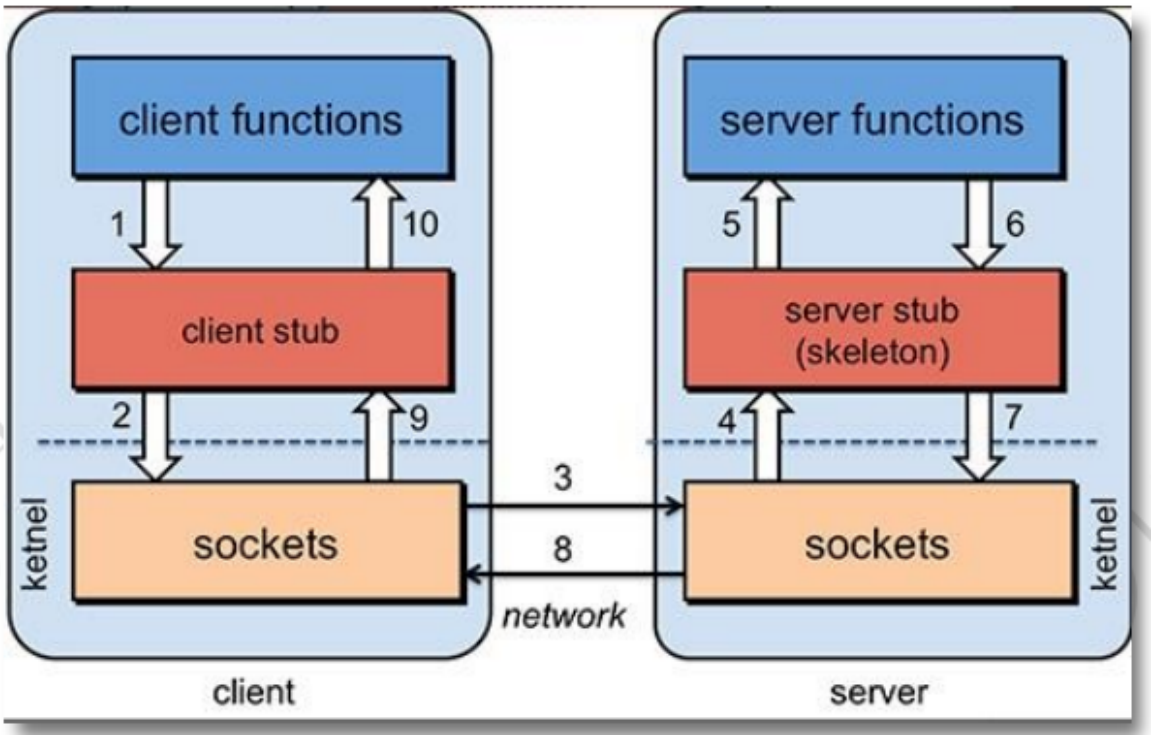
一、概述

高性能要从底层的原理说起，既然是一个RPC框架，主要干的就是远程过程(方法)调用，那么提升性能就要从最关键、最耗时的两个方面入手:序列化和网络通信。

序列化:我们学习Java网络开发的时候知道,本地的对象要在网络上传输，必须要实现 **Serializable**接口，也就是必须序列化。我们序列化的方案很多: xml、json、二进制流...其中效率最高的就是二进制流(因为计算机就是二进制的)。然而Dubbo**采用的就是效率最高的二进制。**

网络通信:不同于HTTP需要进行7步走(三次握手和四次挥手)，Dubbo**采用Socket通信机制**,一步到位,提升了通信效率,并且可以建立长连接,不用反复连接,直接传输数据

RPC基本原理



1. 调用方 client 要使用右侧 server 的功能（方法），发起对方法的调用
2. client stub 是 RPC 中定义的存根，看做是 client 的助手。**stub 把要调用的方法参数进行序列化，方法名称和其他数据包装起来。**
3. 通过网络 socket(网络通信的技术)，**把方法调用的细节内容发送给右侧的 server**
4. server 端通过 socket **接收请求的方法名称，参数等数据**，传给 stub。
5. server 端接到的数据由 serverstub(server 的助手)处理，调用 server 的真正方法，处理业务
6. server 方法处理完业务，把处理的结果对象（Object）交给了助手，助手把 Object 进行序列化，对象转为二进制数据。
7. server 助手二进制数据交给网络处理程序
8. 通过网络将二进制数据，发送给 client。
9. client 接数据，交给 client 助手。

10. client 助手，接收数据**通过反序列化为 java 对象（Object）**，作为远程方法调用结果。

二、Dubbo概述

概念

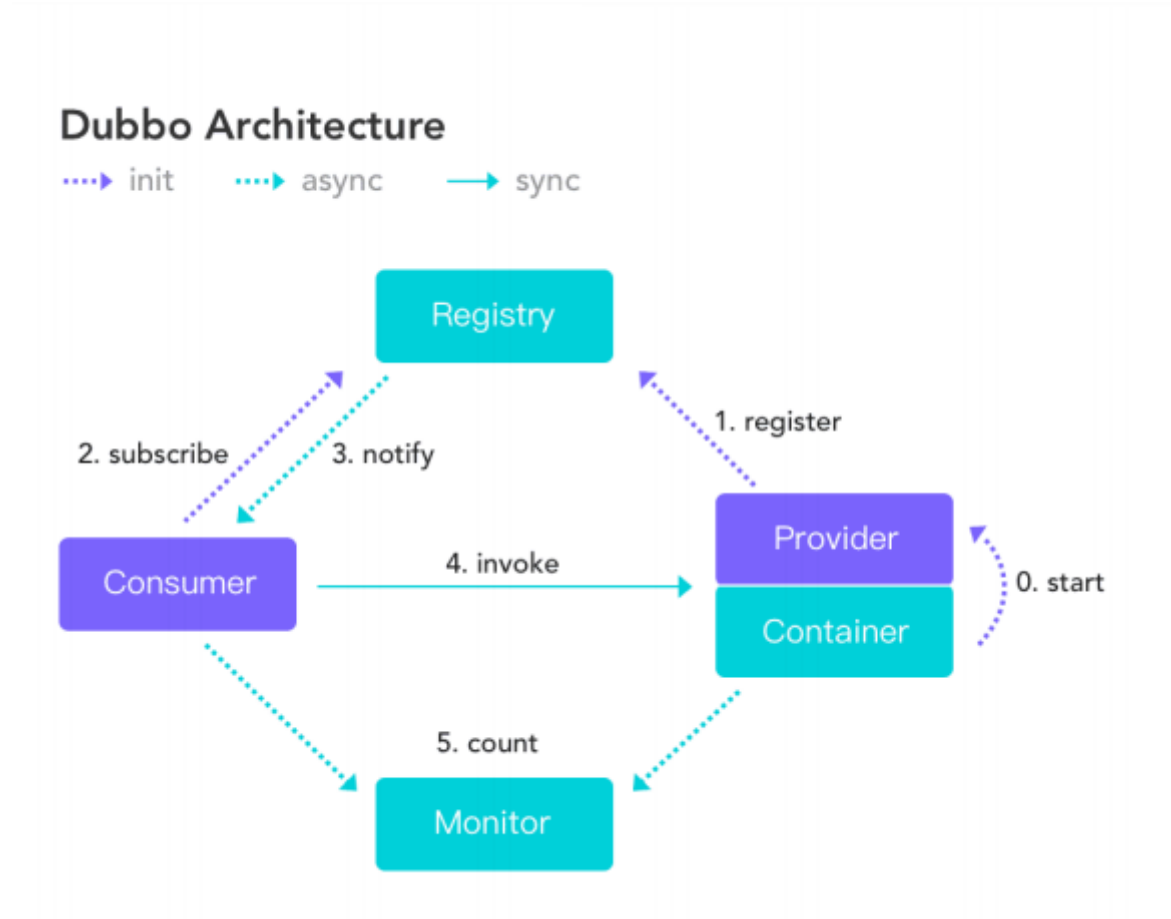
Apache Dubbo (incubating)是一款高性能、轻量级的开源Java RPC框架，一个分布式服务框架，它提供了三大核心能力:

面向接口的远程方法调用：调用接口的方法，在 A 服务器调用 B 服务器的方法，由dubbo实现对 B 的调用，无需关心实现的细节，就像MyBatis访问Dao的接口，可以操作数据库一样。不用关心 Dao 接口方法的实现。

智能容错和负载均衡

服务自动注册和发现

基本架构



服务提供者（Provider）：暴露服务的服务提供方，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(Consumer)：:调用远程服务的服务消费方，服务消费者在启动时，向注册中心订阅自己所需的服务，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。

注册中心（Registry）：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

监控中心（Monitor）：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

调用关系说明

1. 服务容器负责启动，加载，运行服务提供者
2. 服务提供者在启动时，向注册中心注册自己提供的服务。
3. 服务消费者在启动时，向注册中心订阅自己所需的服务。
4. 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。
5. 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。
6. 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。

Dubbo支持的协议

支持多种协议：dubbo , hessian , rmi , http, webservice , thrift , memcached , redis。

dubbo 官方推荐使用 dubbo 协议。**dubbo 协议默认端口 20880**

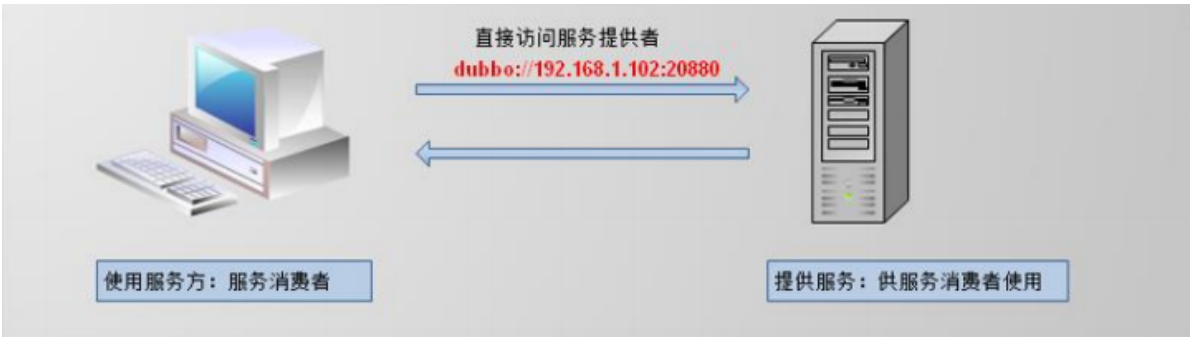
使用 dubbo 协议，spring 配置文件加入：

```
<dubbo:protocol name="dubbo" port="20880"/>
```

Dubbo直连方式模式

点对点的直连项目:消费者直接访问服务提供者，没有注册中心。消费者必须指定服务提供者的访问地址（url）。

消费者直接通过 url 地址访问固定的服务提供者。这个 url 地址是不变的。



一、创建服务提供者

1. **新建java project**

   项目名称:001-link-provider

   版本：1.0-SNAPSHOT

2. **编写maven的pom.xml**

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

```xml
<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.qiangliu8.dubbo</groupId>
  <artifactId>001-link-provider</artifactId>
  <version>1.0-SNAPSHOT</version>
    <!--安装到仓库时改成jar方式，运行改成war-->
  <packaging>war</packaging>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
      <!--在<dependency>中加入 spring依赖-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.2.5.RELEASE</version>
    </dependency>
      <!--在<dependency>中加入 springmvc依赖-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.2.5.RELEASE</version>
    </dependency>
      <!--在<dependency>中加入dubbo依赖-->
    <dependency>
      <groupId>com.alibaba</groupId>
      <artifactId>dubbo</artifactId>
      <version>2.6.2</version>
    </dependency>
  </dependencies>

  <build>
    <!--在<build>中加入 plugin-->
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>15</source>
          <target>15</target>
        </configuration>
      </plugin>
    </plugins>
```

```
  </build>
</project>
```

3. **创建类：User和UserService接口和UserService接口实现类**

```
public class User implements Serializable {
    private Integer id;
    private String username;
    private Integer age;
}
```

```
public interface UserService {
    public User queryUserById(Integer id);
}
```

```
public class UserServiceImpl implements UserService {
    @Override
    public User queryUserById(Integer id) {
        User user = new User(id,"刘强",23);
        return user;
    }
}
```

4. **创建 dubbo 配置文件**

```
<?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://dubbo.apache.org/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://dubbo.apache.org/schema/dubbo
http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
    <!--服务提供者声明名称：必须保证服务名称的唯一性-->
    <dubbo:application name="001-dubbodemo-userservice-provider"/>
    <!--访问服务协议的名称及端口号，dubbo官网推荐使用dubbo协议，端口号默认为20880-
->
    <!--name：指定协议的名称 port：指定协议的端口号-->
    <dubbo:protocol name="dubbo" port="20880"/>

    <!--暴露接口服务
        interface:暴露服务接口的全限定类名
        ref:接口引用的实现类在spring容器中的标识
        registry:如果不使用注册中心，值为N/A-->
    <dubbo:service interface="com.qiangliu8.dubbo.service.UserService"
ref="UserService" registry="N/A" />

    <!--
        将接口的实现类加载到spring容器中
    -->
    <bean id="UserService"
class="com.qiangliu8.dubbo.service.UserServiceImpl"/>

</beans>
```

5. **编写web.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <!--注册监听器-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:dubbo-userservice-provider.xml</param-
value>
    </context-param>
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
    </listener>
</web-app>
```

二、创建消费者

1. asd

```
<?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.qiangliu8.dubbo</groupId>
  <artifactId>002-link-consumer</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <!--spring依赖-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.2.5.RELEASE</version>
```

```
    </dependency>
     <!--springmvc依赖-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.2.5.RELEASE</version>
    </dependency>
      <!--dubbo依赖-->
    <dependency>
      <groupId>com.alibaba</groupId>
      <artifactId>dubbo</artifactId>
      <version>2.6.2</version>
    </dependency>
    <!--依赖服务提供者-->
    <dependency>
      <groupId>com.qiangliu8.dubbo</groupId>
      <artifactId>001-link-provider</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>
</project>
```

2. **创建 dubbo 配置文件**

```
<?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">

    <!--声明服务消费者的名称，保证唯一性-->
    <dubbo:application name="002-consumer" />

    <!--
        启用远程服务接口
        id:远程服务接口对象的名称
        interface:调用远程接口的全限定类名
        url:访问服务接口气质
        registry：不适用注册中心，N/A
    -->
    <dubbo:reference id="UserService"
                     interface="com.qiangliu8.dubbo.service.UserService"
                     url="dubbo://localhost:20880"
                     registry="N/A"/>
</beans>
```

3. **springmvc配置文件**

```
<?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:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
```

```
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        https://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!--扫描组件-->
    <context:component-scan base-
package="com.qiangliu8.dubbo.Controller"/>
    <!--配置注解驱动-->
    <mvc:annotation-driven/>

    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver"
>
        <property name="prefix" value="/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

</beans>
```

4. **编写controller类**

```
@Controller
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping(value = "/user")
    public String userDetail(Model model, Integer id){
        User user = userService.queryUserById(id);
        model.addAttribute("user",user);
        return "userDetail";
    }
}
```

5. **web.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

<servlet>
  <servlet-name>dispatcherServlet</servlet-name>
  <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:application.xml,classpath:dubbo-
consumer.xml</param-value>
  </init-param>
</servlet>
```

```
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

dubbo服务化最佳实践

分包

建议将服务接口、服务模型、服务异常等**均放在公共包中。**

粒度

服务接口尽可能大粒度，每个服务方法应代表一个功能，而不是某功能的一个步骤，否则将面临分布式事务问题，Dubbo 暂未提供分布式事务支持。

服务接口建议以业务场景为单位划分，并对相近业务做抽象，防止接口数量爆炸。

不建议使用过于抽象的通用接口，如：Map query(Map)，这样的接口没有明确语义，会给后期维护带来不便。

版本

每个接口都应定义版本号，为后续不兼容升级提供可能，如：

```
<dubbo:service interface="com.xxxService" version="1.0"/>
```

建议使用两位版本号，要变更服务版本。先升级一半提供者为新版本，再将消费者全

部升为新版本，然后将剩下的一半提供者升为新版本。

dubbo公用标签

三、注册中心Zookeeper

对于服务提供方，它需要发布服务，而且由于应用系统的复杂性，服务的数量、类型也不断膨胀；对于服务消费方，它最关心如何获取到它所需要的服务，而面对复杂的应用系统，需要管理大量的服务调用。

而且，对于服务提供方和服务消费方来说，他们还有可能兼具这两种角色，即需要提供服务，有需要消费服务。 通过将服务统一管理起来，可以有效地优化内部应用对服务发布使用的流程和管理。服务注册中心可以通过特定协议来完成服务对外的统一。

Dubbo 提供的注册中心有如下几种类型可供选：

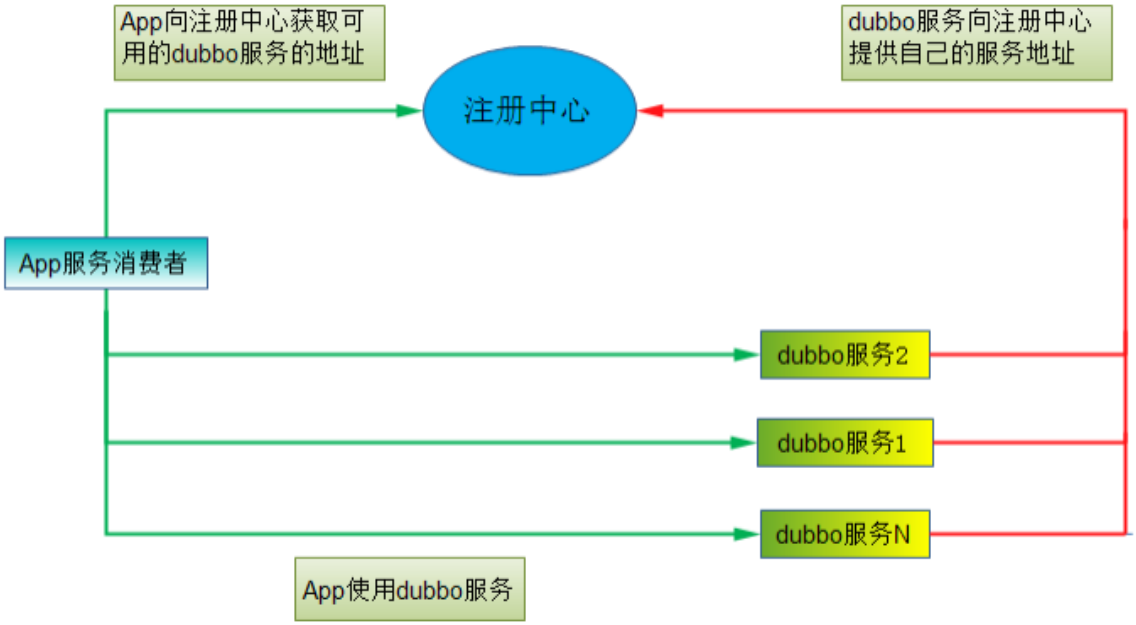
Multicast 注册中心：组播方式

Redis 注册中心：使用 Redis 作为注册中心

Simple 注册中心：就是一个 dubbo 服务。作为注册中心。提供查找服务的功能。

Zookeeper 注册中心：使用 Zookeeper 作为注册中心

注册中心工作方式



Zookeeper 注册中心

Zookeeper 是一个高性能的，分布式的，开放源码的分布式应用程序协调服务。简称 zk。Zookeeper 是翻译管理是动物管理员。可以理解为 windows 中的资源管理器或者注册表。他是一个**树形结构**。这种树形结构和标准文件系统相似。ZooKeeper 树中的每个节点被称为Znode。和文件系统的目录树一样，**ZooKeeper 树中的每个节点可以拥有子节点**。每个节点表示一个唯一服务资源。Zookeeper 运行需要 java 环境。

安装配置 Zookeeper

window下安装 Zookeeper

1. 下载的文件 zookeeper-3.5.4-beta.tar.gz. 解压后到目录就可以了，例如 d:/servers/zookeeper
2. 将zookeeper/conf/ 目录下配置文件zoo_sample.cfg复制成一份，改名zoo.cfg
3. 修改zoo.cfg

```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=F:/data/zookeeper
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
```

tickTime: 心跳的时间，单位毫秒. Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔，也就是每个 tickTime 时间就会发送一个心跳。表明存活状态。

dataDir: 数据目录，可以是任意目录。存储 zookeeper 的快照文件、pid 文件，默认为/tmp/zookeeper，建议在 zookeeper 安装目录下创建 data 目录，将 dataDir 配置改为/usr/local/zookeeper-3.4.10/data

clientPort: 客户端连接 zookeeper 的端口，即 zookeeper 对外的服务端口，默认为 2181

配置内容:

1. dataDir : zookeeper 数据的存放目录
2. admin.serverPort=1028

原因：zookeeper 3.5.x 占用 8080

4. 启动bin目录下的zkServer.cmd

linux下安装 Zookeeper

1. Zookeeper 的运行需要 jdk。使用前 Linux 系统要安装好 jdk
2. 上传 zookeeper-3.5.4-beta.tar.gz.并解压

> 执行命令：tar -zxvf zookeeper-3.5.4-beta.tar.gz -C /opt

> 改名：mv zookeeper-3.5.4-beta.tar.gz zookeeper

3. 配置文件

   在 zookeeper 的 conf 目录下，将 zoo_sample.cfg 复制一份并改名改名为 zoo.cfg，

> 复制命令：cp zoo_sample.cfg zoo.cfg

   zookeeper 启动时会读取该文件作为默认配置文件。

4. 修改配置文件，如window版本的那种改端口号
5. 启动 Zookeeper

> 启动（切换到安装目录的 bin 目录下）：./zkServer.sh start

6. 关闭 Zookeeper

> 关闭（切换到安装目录的 bin 目录下）：./zkServer.sh stop

使用注册中心

一、.006-zk-interface

1.在此创建类和接口

```
public class User implements Serializable {
    private Integer id;
    private String username;
}
```

```
public interface UserService {
    User queryUserById(Integer id);
}
```

二、007-zk-userservice-provider

1. 引入spring、springmvc、zookeeper、006接口的依赖

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.5.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.2.5.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>dubbo</artifactId>
    <version>2.6.2</version>
  </dependency>
  <dependency>
    <groupId>com.qiangliu8.dubbo</groupId>
    <artifactId>006-zk-interface</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>
  <dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
    <version>4.1.0</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>15</source>
        <target>15</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

2. 编写服务提供者的配置文件dubo-zk-userservice-provider.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">

    <!--1.声明dubbo服务提供者的名称，保证唯一性-->
    <dubbo:application name="007-zk-userservice-provider"/>
    <!--2.声明dubbo使用的协议名称和端口号-->
    <dubbo:protocol name="dubbo" port="20880"/>
    <!--3.使用zookeeper注册中心-->
    <!--指定注册中心地址和端口号-->
    <dubbo:registry address="zookeeper://222.204.55.121:2181"/>
    <!--4.暴露服务接口-->
    <dubbo:service interface="com.qiangliu8.dubbo.service.UserService"
ref="UserService" />
    <!--5.加载接口实现类-->
    <bean name="UserService"
class="com.qiangliu8.dubbo.impl.UserServiceImpl"/>
</beans>
```

3. 设置监听器，设置配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:dubo-zk-userservice-provider.xml</param-value>
  </context-param>
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
  </listener>
</web-app>
```

4. 编写接口的实现类

```
public class UserServiceImpl implements UserService {
    @Override
    public User queryUserById(Integer id) {
        User user = new User(id,"我的爱");
        return  user;
    }
}
```

三、008-zk-consumer

1. 引入spring、springmvc、zookeeper、006接口、dubbo的依赖

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.5.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.2.5.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>dubbo</artifactId>
    <version>2.6.2</version>
  </dependency>
  <dependency>
    <groupId>com.qiangliu8.dubbo</groupId>
    <artifactId>006-zk-interface</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>
  <dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
    <version>4.1.0</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>15</source>
        <target>15</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

2. 编写springmvc的配置文件，接口扫描和视图解析器

```
<?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:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       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://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <context:component-scan base-
package="com.qiangliu8.dubbo.controller"/>
    <mvc:annotation-driven/>
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver"
>
        <property name="prefix" value="/"/>
        <property name="suffix" value=".jsp"/>
    </bean>
</beans>
```

3. 编写服务消费者的配置文件dubo-zk-consumer.xml

   不需要再指定服务提供者的url了，只需要注册中心的地址，输入接口的id就可以

```
<?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="008-zk-consumer"/>
    <!--指定注册中心-->
    <dubbo:registry address="zookeeper://222.204.55.121:2181"/>
    <!--引用远程服务-->
    <dubbo:reference
            interface="com.qiangliu8.dubbo.service.UserService"
            id="UserService"
    />
</beans>
```

4. 设置springmvc的配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
```

```
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:application.xml,classpath:dubo-zk-
consumer.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

5. 编写控制器类

```
@Controller
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping("/user")
    public ModelAndView getUser(){
        ModelAndView modelAndView = new ModelAndView();
        User user = userService.queryUserById(1003);
        modelAndView.addObject("user",user);
        modelAndView.setViewName("user");
        return modelAndView;
    }
}
```

四、版本号version的使用

1. 创建两个接口实现类

```
public class UserServiceImpl1 implements UserService {
    @Override
    public User queryUserById(Integer id) {
        User user = new User(id,"我的心1");
        return  user;
    }
}
```

```
public class UserServiceImpl2 implements UserService {
    @Override
    public User queryUserById(Integer id) {
        User user = new User(id,"我的心2");
        return  user;
    }
}
```

2. **根据不同实现类设置不同的版本**

```
<?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="010-zk-mutli-consumer"/>
    <dubbo:registry address="zookeeper://localhost:2181"/>
    <dubbo:reference id="userService1"
interface="com.qiangliu8.dubbo.service.UserService" version="1.0.0"/>
    <dubbo:reference id="userService2"
interface="com.qiangliu8.dubbo.service.UserService" version="2.0.0"/>
</beans>
```

3. **消费者根据不同版本配置不同id**

```
<?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="010-zk-mutli-consumer"/>
    <dubbo:registry address="zookeeper://localhost:2181"/>
    <dubbo:reference id="userService1"
interface="com.qiangliu8.dubbo.service.UserService" version="1.0.0"/>
    <dubbo:reference id="userService2"
interface="com.qiangliu8.dubbo.service.UserService" version="2.0.0"/>

</beans>
```

4. **根据id自动注入**

```
@Controller
public class UserController {
    @Autowired
    private UserService userService1;
    @Autowired
    private UserService userService2;

    @RequestMapping("/user")
    public ModelAndView getUser(){
        ModelAndView modelAndView = new ModelAndView();
        User user1 = userService1.queryUserById(1);
        User user2 = userService2.queryUserById(2);
        modelAndView.addObject("user1",user1);
        modelAndView.addObject("user2",user2);
        modelAndView.setViewName("user");
        return modelAndView;
    }
```

```
}
```

五、dubbo配置原则

关闭检查

dubbo 缺省会在启动时检查依赖的服务是否可用，不可用时会抛出异常，阻止 Spring 初始化完成，以便上线时，能及早发现问题，默认 check=true。通过 check="false"关闭检查，

比如，测试时，有些服务不关心，或者出现了循环依赖，必须有一方先启动。

关闭某个服务的启动时检查

```
<dubbo:reference id="userService1"
interface="com.qiangliu8.dubbo.service.UserService" version="1.0.0"
check="false"/>
```

关闭注册中心启动时检查

```
<dubbo:registry address="zookeeper://localhost:2181" check="false"/>
```

默认启动服务时检查注册中心存在并已运行。注册中心不启动会报错。

重试次数

消费者访问提供者，如果访问失败，则切换重试访问其它服务器，但重试会带来更长延迟。访问时间变长，用户的体验较差。多次重新访问服务器有可能访问成功。可通过 **retries="2"** 来设置重试次数(不含第一次)。

```
<dubbo:service version="1.0.0" retries="2"/>
```

```
<dubbo:reference retries="2" />
```

超时时间

由于网络或服务端不可靠，会导致调用出现一种不确定的中间状态（超时）。为了避免超时导致客户端资源（线程）挂起耗尽，必须设置超时时间。

timeout：调用远程服务超时时间(毫秒)

dubbo服务端

```
指定接口超时配置

<dubbo:server interface="com.foo.BarService" timeout="2000" />
```

dubbo消费端

```
指定接口超时配置

<dubbo:reference interface="com.foo.BarService" timeout="2000" />
```

六、监控中心

dubbo 的使用，其实只需要有注册中心，消费者，提供者这三个就可以使用了，但是并不能看到有哪些消费者和提供者，为了更好的调试，发现问题，解决问题，因此引入 dubbo-admin。

通过 dubbo-admin 可以对消费者和提供者进行管理。可以在 dubbo 应用部署做动态的调整，服务的管理。

1. **运行管理后台dubbo-admin**

   到 dubbo-admin-0.0.1-SNAPSHOT.jar 所在的目录。执行下面命令

   java -jar dubbo-admin-0.0.1-SNAPSHOT.jar

2. **修改配置dubbo-properties文件**

   application.properties 文件，内容如下：

3. **运行dubbo-admin 应用**

   1） 先启动注册中心

   2） 执行提供者项目

   3） java -jar dubbo-admin-0.0.1-SNAPSHOT.jar 启动 dubbo 管理后台

   4） 在浏览器地址栏输入 http://localhost:7001 访问监控中心-控制台