02-Dubbo特性详解-注册中心

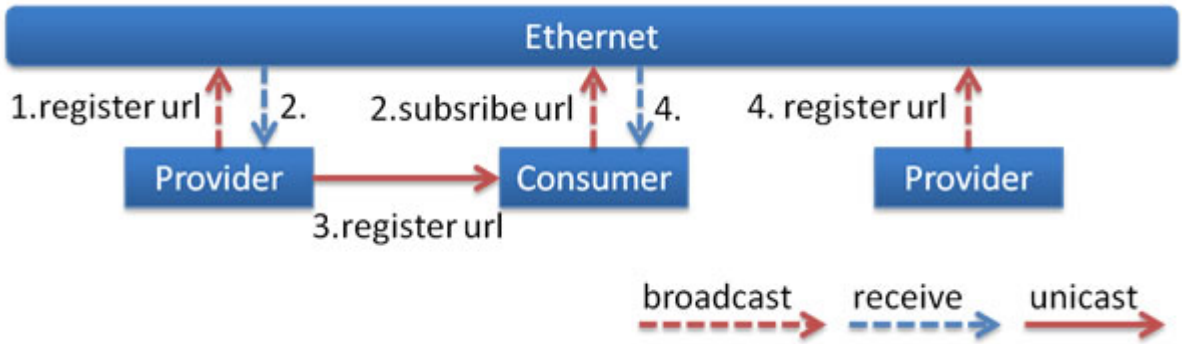
1 注册中心

1.1 支持多类型注册中心

1.1.1 Multicast 注册中心 【掌握】

工作原理

Multicast 注册中心不需要启动任何中心节点，只要广播地址一样，就可以互相发现。



1. 提供方启动时广播自己的地址
2. 消费方启动时广播订阅请求
3. 提供方收到订阅请求时，单播自己的地址给订阅者，如果设置了 `unicast=false`，则广播给订阅者
4. 消费方收到提供方地址时，连接该地址进行 RPC 调用。

组播受网络结构限制，只适合小规模应用或开发阶段使用。组播地址段: 224.0.0.0 - 239.255.255.255

配置

```
<dubbo:registry address="multicast://224.5.6.7:1234" />
```

或

```
<dubbo:registry protocol="multicast" address="224.5.6.7:1234" />
```

为了减少广播量，Dubbo 缺省使用单播发送提供者地址信息给消费者，如果一个机器上同时启了多个消费者进程，消费者需声明 `unicast=false`，否则只会有一个消费者能收到消息：

```
<dubbo:registry address="multicast://224.5.6.7:1234?unicast=false" />
```

或

```
<dubbo:registry protocol="multicast" address="224.5.6.7:1234">
    <dubbo:parameter key="unicast" value="false" />
</dubbo:registry>
```

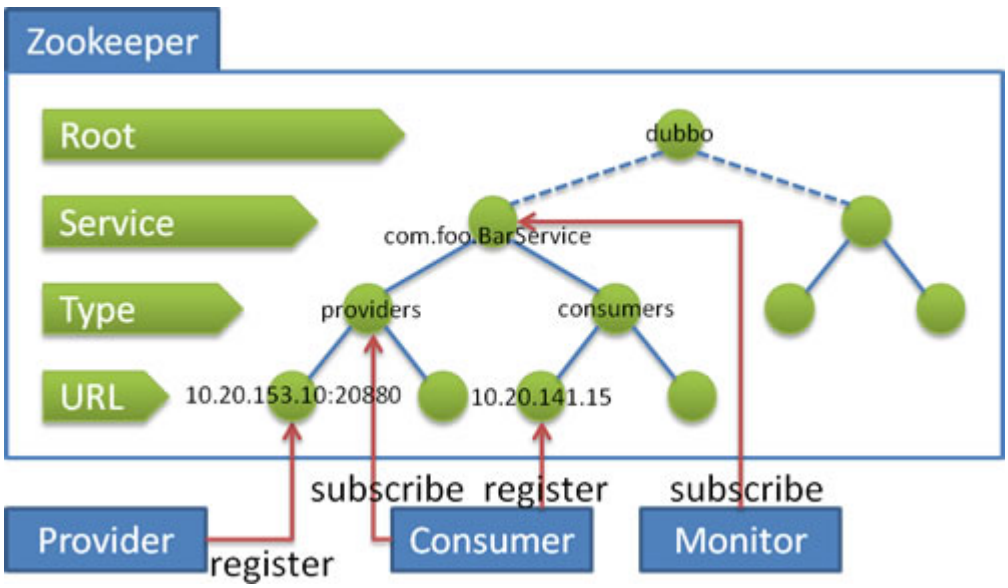
注解方式参数配置：

```
dubbo.registry.address=multicast://224.5.6.7:1234
```

1.1.2 zookeeper 注册中心 【掌握】

Zookeeper 是 Apacahe Hadoop 的子项目，是一个树型的目录服务，支持变更推送，适合作为 Dubbo 服务的注册中心，工业强度较高，可用于生产环境，并推荐使用。

工作原理



流程说明：

- 服务提供者启动时: 向 `/dubbo/com.foo.BarService/providers` 目录下写入自己的 URL 地址
- 服务消费者启动时: 订阅 `/dubbo/com.foo.BarService/providers` 目录下的提供者 URL 地址。并向 `/dubbo/com.foo.BarService/consumers` 目录下写入自己的 URL 地址
- 监控中心启动时: 订阅 `/dubbo/com.foo.BarService` 目录下的所有提供者和消费者 URL 地址。

支持以下功能：

- 当提供者出现断电等异常停机时，注册中心能自动删除提供者信息
- 当注册中心重启时，能自动恢复注册数据，以及订阅请求
- 当会话过期时，能自动恢复注册数据，以及订阅请求
- 当设置 `<dubbo:registry check="false" />` 时，记录失败注册和订阅请求，后台定时重试
- 可通过 `<dubbo:registry username="admin" password="1234" />` 设置 zookeeper 登录信息
- 可通过 `<dubbo:registry group="dubbo" />` 设置 zookeeper 的根节点，不设置将使用无根树
- 支持 `*` 号通配符 `<dubbo:reference group="*" version="*" />`，可订阅服务的所有分组和所有版本的提供者

使用

在 provider 和 consumer 中增加 zookeeper 客户端 jar 包依赖：

```
<!-- 引入zookeeper服务对应版本的zookeeper jar -->
<dependency>
    <groupId>org.apache.zookeeper</groupId>
    <artifactId>zookeeper</artifactId>
    <version>3.4.11</version>
</dependency>
```

或直接下载。

Dubbo 支持 zkclient 和 curator 两种 Zookeeper 客户端实现：

使用 zkclient 客户端

从 `2.2.0` 版本开始缺省为 zkclient 实现，以提升 zookeeper 客户端的健状性。zkclient 是 Datameer 开源的一个 Zookeeper 客户端实现。

缺省配置：

```
<dubbo:registry address="zookeeper://10.20.153.10:2181" client="zkclient"
/>
```

或：

```
dubbo.registry.client=zkclient
```

或：

```
zookeeper://10.20.153.10:2181?client=zkclient
```

需依赖或直接下载：

```
<dependency>
    <groupId>com.github.sgroschupf</groupId>
    <artifactId>zkclient</artifactId>
    <version>0.1</version>
</dependency>
```

使用 curator 客户端

从 `2.3.0` 版本开始支持可选 curator 实现。Curator 是 Netflix 开源的一个 Zookeeper 客户端实现。

如果需要改为 curator 实现，请配置：

```
<dubbo:registry ... client="curator" />
```

或：

```
dubbo.registry.client=curator
```

或：

```
zookeeper://10.20.153.10:2181?client=curator
```

需依赖或直接下载：

```
<!-- 默认使用的是第三方zookeeper客户端 curator -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
    <version>4.2.0</version>
    <!-- 如果你使用的zookeeper服务版本不是3.5的，请排除自动依赖，再单独引入
zookeeper依赖 -->
    <exclusions>
        <exclusion>
            <groupId>org.apache.zookeeper</groupId>
            <artifactId>zookeeper</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Zookeeper 单机配置:

```
<dubbo:registry address="zookeeper://10.20.153.10:2181" />
```

或：

```
<dubbo:registry protocol="zookeeper" address="10.20.153.10:2181" />
```

Zookeeper 集群配置：

```
<dubbo:registry address="zookeeper://10.20.153.10:2181?
backup=10.20.153.11:2181,10.20.153.12:2181" />
```

或：

```
<dubbo:registry protocol="zookeeper"
address="10.20.153.10:2181,10.20.153.11:2181,10.20.153.12:2181" />
```

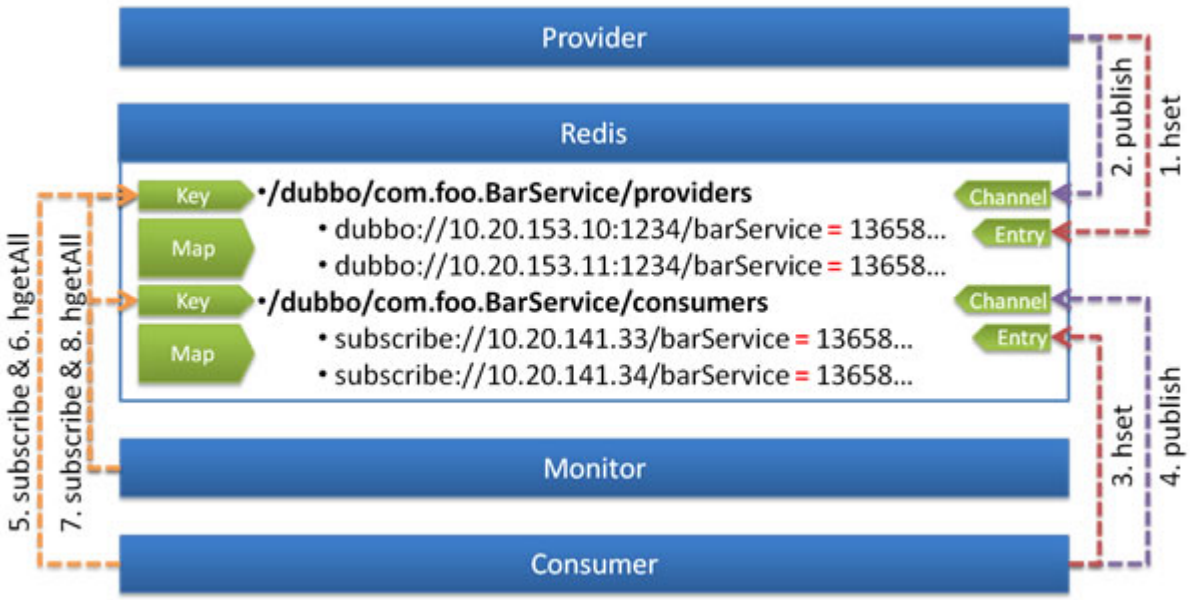
同一 Zookeeper，分成多组注册中心:

```
<dubbo:registry id="chinaRegistry" protocol="zookeeper"
address="10.20.153.10:2181" group="china" />
<dubbo:registry id="intlRegistry" protocol="zookeeper"
address="10.20.153.10:2181" group="intl" />
```

1.1.3 Redis 注册中心 【了解】

基于 Redis 实现的注册中心（从 `2.1.0` 版本开始支持）

工作原理



使用 Redis 的 Key/Map 结构存储数据结构：

- 主 Key 为服务名和类型
- Map 中的 Key 为 URL 地址
- Map 中的 Value 为过期时间，用于判断脏数据，脏数据由监控中心删除 (Redis 过期数据通过心跳的方式检测脏数据，服务器时间必须同步，并且对服务器有一定压力，否则过期检测会不准确)

使用 Redis 的 Publish/Subscribe 事件通知数据变更：

- 通过事件的值区分事件类型：`register`, `unregister`, `subscribe`, `unsubscribe`
- 普通消费者直接订阅指定服务提供者的 Key，只会收到指定服务的 `register`, `unregister` 事件
- 监控中心通过 `psubscribe` 功能订阅 `/dubbo/*`，会收到所有服务的所有变更事件

调用过程：

1. 服务提供方启动时，向 `Key:/dubbo/com.foo.BarService/providers` 下，添加当前提供者的地址
2. 并向 `Channel:/dubbo/com.foo.BarService/providers` 发送 `register` 事件
3. 服务消费方启动时，从 `Channel:/dubbo/com.foo.BarService/providers` 订阅 `register` 和 `unregister` 事件
4. 并向 `Key:/dubbo/com.foo.BarService/consumers` 下，添加当前消费者的地址

5. 服务消费方收到 `register` 和 `unregister` 事件后，从 `Key:/dubbo/com.foo.BarService/providers` 下获取提供者地址列表
6. 服务监控中心启动时，从 `Channel:/dubbo/*` 订阅 `register` 和 `unregister`，以及 `subscribe` 和 `unsubsribe` 事件
7. 服务监控中心收到 `register` 和 `unregister` 事件后，从 `Key:/dubbo/com.foo.BarService/providers` 下获取提供者地址列表
8. 服务监控中心收到 `subscribe` 和 `unsubsribe` 事件后，从 `Key:/dubbo/com.foo.BarService/consumers` 下获取消费者地址列表

配置

```
<dubbo:registry address="redis://10.20.153.10:6379" />
```

或

```
<dubbo:registry address="redis://10.20.153.10:6379?
backup=10.20.153.11:6379,10.20.153.12:6379" />
```

或

```
<dubbo:registry protocol="redis" address="10.20.153.10:6379" />
```

或

```
<dubbo:registry protocol="redis"
address="10.20.153.10:6379,10.20.153.11:6379,10.20.153.12:6379" />
```

选项

- 可通过 `<dubbo:registry group="dubbo" />` 设置 redis 中 key 的前缀，缺省为 `dubbo`。
- 可通过

  ```
  <dubbo:registry cluster="replicate" />
  ```

  设置 redis 集群策略，缺省为

  ```
  failover
  ```

  ：

- `failover`: 只写入和读取任意一台，失败时重试另一台，需要服务器端自行配置数据同步
- `replicate`: 在客户端同时写入所有服务器，只读取单台，服务器端不需要同步，注册中心集群增大，性能压力也会更大

可靠性声明

阿里内部并没有采用 Redis 做为注册中心，而是使用自己实现的基于数据库的注册中心，即：Redis 注册中心并没有在阿里内部长时间运行的可靠性保障，此 Redis 桥接实现只为开源版本提供，其可靠性依赖于 Redis 本身的可靠性。

1.1.4 Simple 注册中心 【了解】

Simple 注册中心本身就是一个普通的 Dubbo 服务，可以减少第三方依赖，使整体通讯方式一致。

配置

将 Simple 注册中心暴露成 Dubbo 服务：

```
<?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-4.3.xsd
http://dubbo.apache.org/schema/dubbo
http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
    <!-- 当前应用信息配置 -->
    <dubbo:application name="simple-registry" />
    <!-- 暴露服务协议配置 -->
    <dubbo:protocol port="9090" />
    <!-- 暴露服务配置 -->
    <dubbo:service interface="org.apache.dubbo.registry.RegistryService"
ref="registryService" registry="N/A" ondisconnect="disconnect"
callbacks="1000">
        <dubbo:method name="subscribe"><dubbo:argument index="1"
callback="true" /></dubbo:method>
        <dubbo:method name="unsubscribe"><dubbo:argument index="1"
callback="false" /></dubbo:method>
    </dubbo:service>
```

```
    <!-- 简单注册中心实现，可自行扩展实现集群和状态同步 -->
    <bean id="registryService"
class="org.apache.dubbo.registry.simple.SimpleRegistryService" />
</beans>
```

引用 Simple Registry 服务：

```
<dubbo:registry address="127.0.0.1:9090" />
```

或者：

```
<dubbo:service interface="org.apache.dubbo.registry.RegistryService"
group="simple" version="1.0.0" ... >
```

或者：

```
<dubbo:registry address="127.0.0.1:9090" group="simple" version="1.0.0" />
```

适用性说明

此 `SimpleRegistryService` 只是简单实现，不支持集群，可作为自定义注册中心的参考，但不适合直接用于生产环境。

1.2 支持多注册中心

Dubbo 支持同一服务向多注册中心同时注册，或者不同服务分别注册到不同的注册中心上去，甚至可以同时引用注册在不同注册中心上的同名服务。另外，注册中心是支持自定义扩展的[1]。

多注册中心注册

比如：中文站有些服务来不及在青岛部署，只在杭州部署，而青岛的其它应用需要引用此服务，就可以将服务同时注册到两个注册中心。

```
<?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-4.3.xsd
http://dubbo.apache.org/schema/dubbo
http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
    <dubbo:application name="world"  />
    <!-- 多注册中心配置 -->
    <dubbo:registry id="hangzhouRegistry" address="10.20.141.150:9090" />
    <dubbo:registry id="qingdaoRegistry" address="10.20.141.151:9010"
default="false" />
    <!-- 向多个注册中心注册 -->
    <dubbo:service interface="com.alibaba.hello.api.HelloService"
version="1.0.0" ref="helloService"
registry="hangzhouRegistry,qingdaoRegistry" />
</beans>
```

不同服务使用不同注册中心

比如：CRM 有些服务是专门为国际站设计的，有些服务是专门为中文站设计的。

```
<?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-4.3.xsd
http://dubbo.apache.org/schema/dubbo
http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
    <dubbo:application name="world"  />
    <!-- 多注册中心配置 -->
    <dubbo:registry id="chinaRegistry" address="10.20.141.150:9090" />
    <dubbo:registry id="intlRegistry" address="10.20.154.177:9010"
default="false" />
    <!-- 向中文站注册中心注册 -->
    <dubbo:service interface="com.alibaba.hello.api.HelloService"
version="1.0.0" ref="helloService" registry="chinaRegistry" />
    <!-- 向国际站注册中心注册 -->
```

```
    <dubbo:service interface="com.alibaba.hello.api.DemoService"
version="1.0.0" ref="demoService" registry="intlRegistry" />
</beans>
```

多注册中心引用

比如：CRM 需同时调用中文站和国际站的 PC2 服务，PC2 在中文站和国际站均有部署，接口及版本号都一样，但连的数据库不一样。

```
<?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-4.3.xsd
http://dubbo.apache.org/schema/dubbo
http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
    <dubbo:application name="world"  />
    <!-- 多注册中心配置 -->
    <dubbo:registry id="chinaRegistry" address="10.20.141.150:9090" />
    <dubbo:registry id="intlRegistry" address="10.20.154.177:9010"
default="false" />
    <!-- 引用中文站服务 -->
    <dubbo:reference id="chinaHelloService"
interface="com.alibaba.hello.api.HelloService" version="1.0.0"
registry="chinaRegistry" />
    <!-- 引用国际站站服务 -->
    <dubbo:reference id="intlHelloService"
interface="com.alibaba.hello.api.HelloService" version="1.0.0"
registry="intlRegistry" />
</beans>
```

如果只是测试环境临时需要连接两个不同注册中心，使用竖号分隔多个不同注册中心地址：

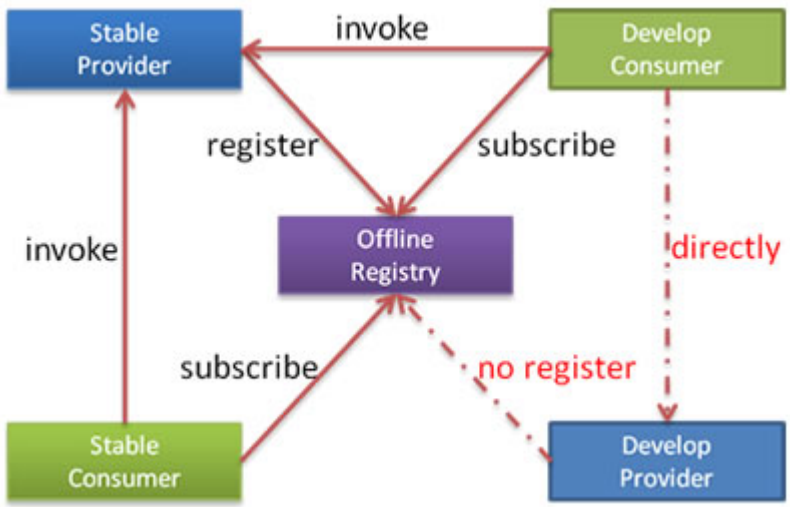
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://dubbo.apache.org/schema/dubbo
http://dubbo.apache.org/schema/dubbo/dubbo.xsd">
    <dubbo:application name="world"  />
    <!-- 多注册中心配置，竖号分隔表示同时连接多个不同注册中心，同一注册中心的多个集群地
址用逗号分隔 -->
    <dubbo:registry address="10.20.141.150:9090|10.20.154.177:9010" />
    <!-- 引用服务 -->
    <dubbo:reference id="helloService"
interface="com.alibaba.hello.api.HelloService" version="1.0.0" />
</beans>
```

1.3 支持多使用场景

1.3.1 只订阅

为方便开发测试，经常会在线下共用一个所有服务可用的注册中心，这时，如果一个正在开发中的服务提供者注册，可能会影响消费者不能正常运行。

可以让服务提供者开发方，只订阅服务(开发的服务可能依赖其它服务)，而不注册正在开发的服务，通过直连测试正在开发的服务。



禁用注册配置

```
<dubbo:registry address="10.20.153.10:9090" register="false" />
```

或者

```
<dubbo:registry address="10.20.153.10:9090?register=false" />
```

1.3.2 只注册

如果有两个镜像环境，两个注册中心，有一个服务只在其中一个注册中心有部署，另一个注册中心还没来得及部署，而两个注册中心的其它应用都需要依赖此服务。这个时候，可以让服务提供者方只注册服务到另一注册中心，而不从另一注册中心订阅服务。

禁用订阅配置

```
<dubbo:registry id="hzRegistry" address="10.20.153.10:9090" />
<dubbo:registry id="qdRegistry" address="10.20.141.150:9090"
subscribe="false" />
```

或者

```
<dubbo:registry id="hzRegistry" address="10.20.153.10:9090" />
<dubbo:registry id="qdRegistry" address="10.20.141.150:9090?
subscribe=false" />
```

1.3.3 直连提供者

在开发及测试环境下，经常需要绕过注册中心，只测试指定服务提供者，这时候可能需要点对点直连，点对点直连方式，将以服务接口为单位，忽略注册中心的提供者列表，A 接口配置点对点，不影响 B 接口从注册中心获取列表。

通过 XML 配置

如果是线上需求需要点对点，可在 `<dubbo:reference>` 中配置 url 指向提供者，将绕过注册中心，多个地址用分号隔开，配置如下 [1]：

```
<dubbo:reference id="xxxService" interface="com.alibaba.xxx.XxxService"
url="dubbo://localhost:20890" />
```

通过 -D 参数指定

在 JVM 启动参数中加入-D参数映射服务地址 [2]，如：

```
java -Dcom.alibaba.xxx.XxxService=dubbo://localhost:20890
```

通过文件映射

如果服务比较多，也可以用文件映射，用 `-Ddubbo.resolve.file` 指定映射文件路径，此配置优先级高于 `<dubbo:reference>` 中的配置 [3]，如：

```
java -Ddubbo.resolve.file=xxx.properties
```

然后在映射文件 `xxx.properties` 中加入配置，其中 key 为服务名，value 为服务提供者 URL：

```
com.alibaba.xxx.XxxService=dubbo://localhost:20890
```

注意 为了避免复杂化线上环境，不要在线上使用这个功能，只应在测试阶段使用。

1.3.4 静态服务

有时候希望人工管理服务提供者的上线和下线，此时需将注册中心标识为非动态管理模式。

```
<dubbo:registry address="10.20.141.150:9090" dynamic="false" />
```

或者

```
<dubbo:registry address="10.20.141.150:9090?dynamic=false" />
```

服务提供者初次注册时为禁用状态，需人工启用。断线时，将不会被自动删除，需人工禁用。

1.4 可配置项说明

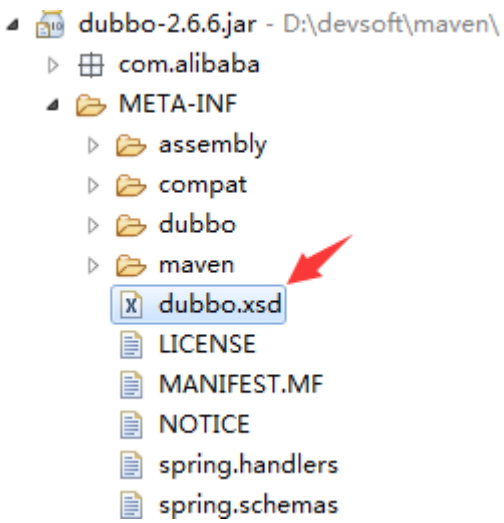
com.alibaba.dubbo.config.RegistryConfig

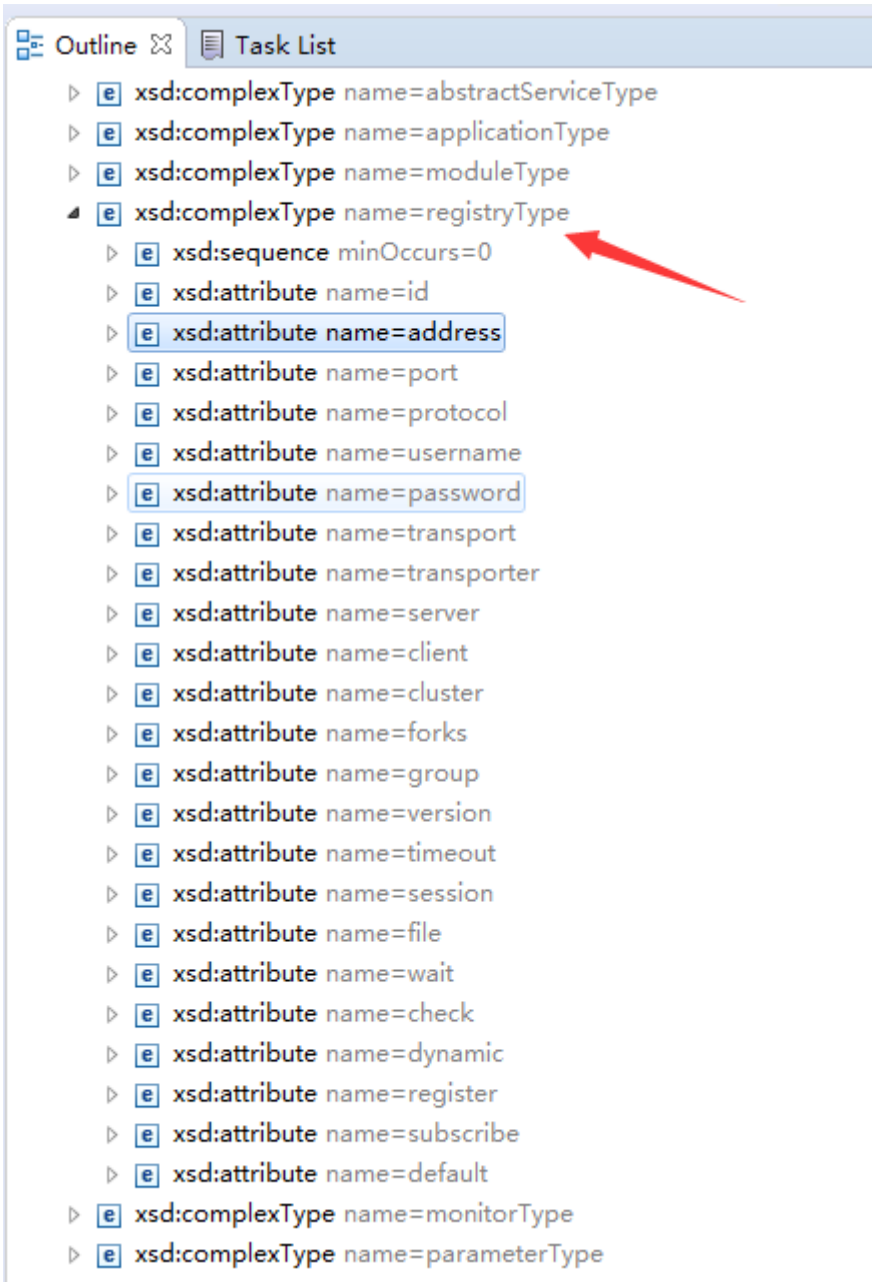
了解RegistryConfig中可设置哪些属性。

spring xml schema 配置

```
<dubbo:registry address="zookeeper://127.0.0.1:2181"/>
```

可配置的属性是RegistryConfig中的属性，可查看它的xml schema定义（有说明信息）来了解。





dubbo:registry

注册中心配置。对应的配置类： `org.apache.dubbo.config.RegistryConfig`。同时如果有多个不同的注册中心，可以声明多个 `<dubbo:registry>` 标签，并在 `<dubbo:service>` 或 `<dubbo:reference>` 的 `registry` 属性指定使用的注册中心。

属性	对应URL参数	类型	是否必填	缺省值	作用	描述	兼容性
id		string	可选		配置关联	注册中心引用BeanId，可以在<dubbo:service registry="">或<dubbo:reference registry="">中引用此ID	1.0.16以上版本
address	host:port	string	**必填**		服务发现	注册中心服务器地址，如果地址没有端口缺省为9090，同一集群内的多个地址用逗号分隔，如：ip:port,ip:port，不同集群的注册中心，请配置多个dubbo:registry标签	1.0.16以上版本
protocol		string	可选	dubbo	服务发现	注册中心地址协议，支持dubbo, http, local三种协议，分别表示：dubbo地址、http地址、本地注册中心	2.0.0以上版本
port		int	可选	9090	服务发现	注册中心缺省端口，当address没有带端口时使用此端口做为缺省值	2.0.0以上版本
username		string	可选		服务治理	登录注册中心用户名，如果注册中心不需要验证可不填	2.0.0以上版本
password		string	可选		服务治理	登录注册中心密码，如果注册中心不需要验证可不填	2.0.0以上版本
transport	registry.transporter	string	可选	netty	性能调优	网络传输方式，可选mina,netty	2.0.0以上版本
timeout	registry.timeout	int	可选	5000	性能调优	注册中心请求超时时间(毫秒)	2.0.0以上版本
session	registry.session	int	可选	60000	性能调优	注册中心会话超时时间(毫秒)，用于检测提供者非正常断线后的脏数据，比如用心跳检测的实现，此时间就是心跳间隔，不同注册中心实现不一样。	2.1.0以上版本
file	registry.file	string	可选		服务治理	使用文件缓存注册中心地址列表及服务提供者列表，应用重启时将基于此文件恢复，注意：两个注册中心不能使用同一文件存储	2.0.0以上版本
wait	registry.wait	int	可选	0	性能调优	停止时等待通知完成时间(毫秒)	2.0.0以上版本
check	check	boolean	可选	true	服务治理	注册中心不存在时，是否报错	2.0.0以上版本

属性	对应URL参数	类型	是否必填	缺省值	作用	描述	兼容性
register	register	boolean	可选	true	服务治理	是否向此注册中心注册服务，如果设为false，将只订阅，不注册	2.0.5以上版本
subscribe	subscribe	boolean	可选	true	服务治理	是否向此注册中心订阅服务，如果设为false，将只注册，不订阅	2.0.5以上版本
dynamic	dynamic	boolean	可选	true	服务治理	服务是否动态注册，如果设为false，注册后将显示为disable状态，需人工启用，并且服务提供者停止时，也不会自动取消注册，需人工禁用。	2.0.5以上版本
group	group	string	可选	dubbo	服务治理	服务注册分组，跨组的服务不会相互影响，也无法相互调用，适用于环境隔离。	2.0.5以上版本
simplified	simplified	boolean	可选	false	服务治理	注册到注册中心的URL是否采用精简模式的（与低版本兼容）	2.7.0以上版本
extra-keys	extraKeys	string	可选		服务治理	在simplified=true时，extraKeys允许你在默认参数外将额外的key放到URL中，格式："interface,key1,key2"。	2.0.5以上版本

注册中心配置参数说明链接：

http://dubbo.apache.org/zh-cn/docs/user/references/xml/dubbo-registry.html

1.5 注册中心扩展

扩展说明

负责服务的注册与发现。

扩展接口

- `org.apache.dubbo.registry.RegistryFactory`
- `org.apache.dubbo.registry.Registry`

扩展配置

```
<!-- 定义注册中心 -->
<dubbo:registry id="xxx1" address="xxx://ip:port" />
<!-- 引用注册中心，如果没有配置registry属性，将在ApplicationContext中自动扫描
registry配置 -->
<dubbo:service registry="xxx1" />
<!-- 引用注册中心缺省值，当<dubbo:service>没有配置registry属性时，使用此配置 -->
<dubbo:provider registry="xxx1" />
```

扩展契约

RegistryFactory.java：

```
public interface RegistryFactory {
    /**
     * 连接注册中心.
     *
     * 连接注册中心需处理契约：<br>
     * 1. 当设置check=false时表示不检查连接，否则在连接不上时抛出异常。<br>
     * 2. 支持URL上的username:password权限认证。<br>
     * 3. 支持backup=10.20.153.10备选注册中心集群地址。<br>
     * 4. 支持file=registry.cache本地磁盘文件缓存。<br>
     * 5. 支持timeout=1000请求超时设置。<br>
     * 6. 支持session=60000会话超时或过期设置。<br>
     *
     * @param url 注册中心地址，不允许为空
     * @return 注册中心引用，总不返回空
     */
    Registry getRegistry(URL url);
}
```

RegistryService.java：

```
public interface RegistryService { // Registry extends RegistryService
    /**
     * 注册服务.
     *
     * 注册需处理契约：<br>
     * 1. 当URL设置了check=false时，注册失败后不报错，在后台定时重试，否则抛出异常。
<br>
     * 2. 当URL设置了dynamic=false参数，则需持久存储，否则，当注册者出现断电等情况
异常退出时，需自动删除。<br>
```

```
     * 3. 当URL设置了category=overrides时，表示分类存储，缺省类别为providers，可按分类部分通知数据。<br>
     * 4. 当注册中心重启，网络抖动，不能丢失数据，包括断线自动删除数据。<br>
     * 5. 允许URI相同但参数不同的URL并存，不能覆盖。<br>
     *
     * @param url 注册信息，不允许为空，如：dubbo://10.20.153.10/com.alibaba.foo.BarService?version=1.0.0&application=kylin
     */
    void register(URL url);

    /**
     * 取消注册服务.
     *
     * 取消注册需处理契约：<br>
     * 1. 如果是dynamic=false的持久存储数据，找不到注册数据，则抛IllegalStateException，否则忽略。<br>
     * 2. 按全URL匹配取消注册。<br>
     *
     * @param url 注册信息，不允许为空，如：dubbo://10.20.153.10/com.alibaba.foo.BarService?version=1.0.0&application=kylin
     */
    void unregister(URL url);

    /**
     * 订阅服务.
     *
     * 订阅需处理契约：<br>
     * 1. 当URL设置了check=false时，订阅失败后不报错，在后台定时重试。<br>
     * 2. 当URL设置了category=overrides，只通知指定分类的数据，多个分类用逗号分隔，并允许星号通配，表示订阅所有分类数据。<br>
     * 3. 允许以interface,group,version,classifier作为条件查询，如：interface=com.alibaba.foo.BarService&version=1.0.0<br>
     * 4. 并且查询条件允许星号通配，订阅所有接口的所有分组的所有版本，或：interface=*&group=*&version=*&classifier=*<br>
     * 5. 当注册中心重启，网络抖动，需自动恢复订阅请求。<br>
     * 6. 允许URI相同但参数不同的URL并存，不能覆盖。<br>
     * 7. 必须阻塞订阅过程，等第一次通知完后再返回。<br>
     *
```

```
     * @param url 订阅条件，不允许为空，如：
consumer://10.20.153.10/com.alibaba.foo.BarService?
version=1.0.0&application=kylin
     * @param listener 变更事件监听器，不允许为空
     */
    void subscribe(URL url, NotifyListener listener);

    /**
     * 取消订阅服务.
     *
     * 取消订阅需处理契约：<br>
     * 1. 如果没有订阅，直接忽略。<br>
     * 2. 按全URL匹配取消订阅。<br>
     *
     * @param url 订阅条件，不允许为空，如：
consumer://10.20.153.10/com.alibaba.foo.BarService?
version=1.0.0&application=kylin
     * @param listener 变更事件监听器，不允许为空
     */
    void unsubscribe(URL url, NotifyListener listener);

    /**
     * 查询注册列表，与订阅的推模式相对应，这里为拉模式，只返回一次结果。
     *
     * @see org.apache.dubbo.registry.NotifyListener#notify(List)
     * @param url 查询条件，不允许为空，如：
consumer://10.20.153.10/com.alibaba.foo.BarService?
version=1.0.0&application=kylin
     * @return 已注册信息列表，可能为空，含义同{@link
org.apache.dubbo.registry.NotifyListener#notify(List<URL>)}的参数。
     */
    List<URL> lookup(URL url);

}
```

NotifyListener.java：

```
public interface NotifyListener {
    /**
     * 当收到服务变更通知时触发。
     *
     * 通知需处理契约：<br>
```

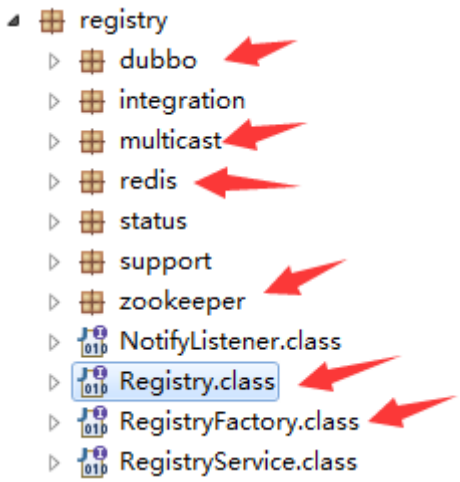
```
     * 1. 总是以服务接口和数据类型为维度全量通知，即不会通知一个服务的同类型的部分数据，用户不需要对比上一次通知结果。<br>
     * 2. 订阅时的第一次通知，必须是一个服务的所有类型数据的全量通知。<br>
     * 3. 中途变更时，允许不同类型的数据分开通知，比如：providers, consumers, routes, overrides，允许只通知其中一种类型，但该类型的数据必须是全量的，不是增量的。<br>
     * 4. 如果一种类型的数据为空，需通知一个empty协议并带category参数的标识性URL数据。<br>
     * 5. 通知者(即注册中心实现)需保证通知的顺序，比如：单线程推送，队列串行化，带版本对比。<br>
     *
     * @param urls 已注册信息列表，总不为空，含义同{@link org.apache.dubbo.registry.RegistryService#lookup(URL)}的返回值。
     */
    void notify(List<URL> urls);

}
```

已知扩展

```
org.apache.dubbo.registry.support.dubbo.DubboRegistryFactory
```

已有实现



扩展示例

Maven 项目结构：

```
src
 |-main
    |-java
        |-com
            |-xxx
                |-XxxRegistryFactoryjava（实现RegistryFactory接口）
                |-XxxRegistry.java（实现Registry接口）
    |-resources
        |-META-INF
            |-dubbo
                |-org.apache.dubbo.registry.RegistryFactory（纯文本文件，内容为：xxx=com.xxx.XxxRegistryFactory）
```

XxxRegistryFactory.java：

```
package com.xxx;

import org.apache.dubbo.registry.RegistryFactory;
import org.apache.dubbo.registry.Registry;
import org.apache.dubbo.common.URL;

public class XxxRegistryFactory implements RegistryFactory {
    public Registry getRegistry(URL url) {
        return new XxxRegistry(url);
    }
}
```

XxxRegistry.java：

```
package com.xxx;

import org.apache.dubbo.registry.Registry;
import org.apache.dubbo.registry.NotifyListener;
import org.apache.dubbo.common.URL;

public class XxxRegistry implements Registry {
    public void register(URL url) {
        // ...
    }
    public void unregister(URL url) {
        // ...
```

```
    }
    public void subscribe(URL url, NotifyListener listener) {
        // ...
    }
    public void unsubscribe(URL url, NotifyListener listener) {
        // ...
    }
}
```

META-INF/dubbo/org.apache.dubbo.registry.RegistryFactory：

```
xxx=com.xxx.XxxRegistryFactory
```

请查看dubbo.jar 中的 META-INF/dubbo/org.apache.dubbo.registry.RegistryFactory文件

1.6源码导读

掌握注册中心部分源码的工作流程

- 注册中心标签解析、实例创建
- 注解方式配置的解析生效
- 服务选定注册中心
- 服务引用指定注册中心