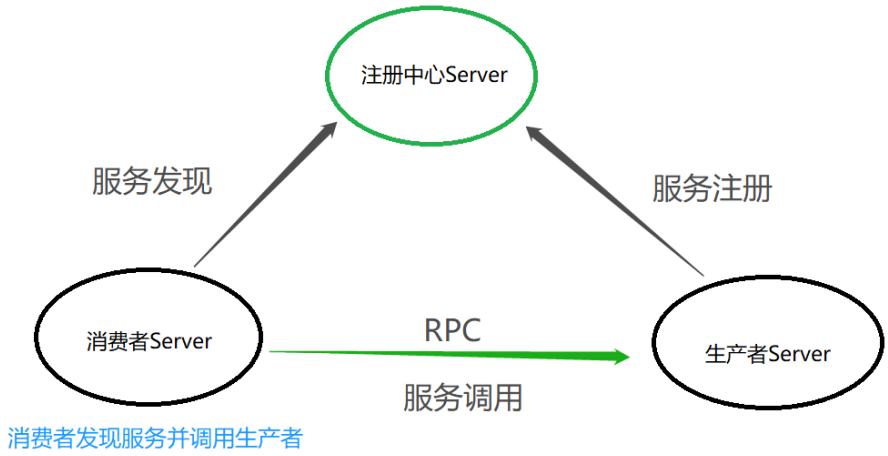
Dubbo+Zookeeper

dubbo+zookeeper作为分布式系统中的一套解决方案，zookeeper作为注册中心，dubbo作为RPC框架。



1.Zookeeper

1.1 单节点

官网地址：http://zookeeper.apache.org/releases.html#download

> windows系统也下载.tar.gz后缀的文件。

进入conf目录，将文件“zoo_sample.cfg”更名为“zoo.cfg”，并添加如下配置：

```ini
# windows下 \\ 或 / 都行
dataDir=D:\\wen_install\\cloud\\apache-zookeeper-3.6.3
# zookeeper端口号
clientPort=2181
```

控制台进入bin目录，执行“zkServer”命令启动。

1.2 zookeeper 集群

打开“zoo.cfg”，添加各个zookeeper节点的信息：

```ini
# 集群: server.编号=IP:端口:选举端口
server.0=192.168.113.201:9898:9999
server.1=192.168.113.202:9898:9999
server.2=192.168.113.203:9898:9999
```

打开“zoo.cfg”，添加 dataDir 路径（每个节点都配置）：

```ini
dataDir=D:\\wen_install\\cloud\\apache-zookeeper-3.6.3\\data
```

在 dataDir 指定的目录创建文件“myid”，并指定节点编号：

```ini
# 第1给节点内容
0
```

2.Dubbo

dubbo 实现调用的方式很简单，实际开发中分成 3 个模块：生产者、消费者、服务接口。

生产者引入服务接口的依赖进行实现，消费者引入服务接口的依赖进行服务调用。

2.1 生产者模块

1.maven 依赖

```xml
<!--服务接口模块-->
<dependency>
    <groupId>org.example</groupId>
    <artifactId>service-api</artifactId>
    <version>1.0</version>
</dependency>

<!-- dubbo spring boot 依赖 -->
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo-spring-boot-starter</artifactId>
    <version>2.7.8</version>
</dependency>

<!-- zookeeper 注册中心依赖 -->
<dependency>
    <groupId>org.apache.zookeeper</groupId>
    <artifactId>zookeeper</artifactId>
    <version>3.5.6</version>
    <!-- 排除相关日志依赖 - 和dubbo-spring-boot-starter引入的log4j冲突了 -->
    <exclusions>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
        </exclusion>
    </exclusions>
</dependency>

<!-- dubbo+zookeeper需要 -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-framework</artifactId>
    <version>4.0.1</version>
</dependency>

<!-- dubbo+zookeeper需要 -->
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-recipes</artifactId>
    <version>4.0.1</version>
</dependency>
```

2.yml 配置

YAML

```yaml
dubbo:
  scan:
    base-packages: com.zk.service   # 指定服务接口的包位置
  application:
    name: server-producer # 服务名
  protocol:
    name: dubbo
    port: 20880  #dubbo协议端口，默认20880
  registry:
    address: zookeeper://127.0.0.1:2181  #注册中心zookeeper地址
```

3.实现服务接口

Java

```java
@DubboService
public class ProducerServiceImpl implements ProducerService {

    @Override
    public void go(String msg) {
        System.out.println("生产者正在生产..."+msg);
    }

}
```

2.2 服务接口模块

1.服务接口无需任何项目依赖，maven中只配置自身模块信息。

2.服务接口无yml配置。

3.服务接口只有接口，无任何业务逻辑。

Java

```java
public interface ProducerService {
    void go(String msg);
}
```

2.3 消费者模块

1.maven依赖（其实和生产者的maven配置一样）

XML

```xml
        <!--服务接口模块-->
        <dependency>
            <groupId>org.example</groupId>
            <artifactId>service-api</artifactId>
            <version>1.0</version>
        </dependency>

        <!-- dubbo spring boot 依赖 -->
        <dependency>
            <groupId>org.apache.dubbo</groupId>
            <artifactId>dubbo-spring-boot-starter</artifactId>
            <version>2.7.8</version>
        </dependency>

        <!-- zookeeper 注册中心依赖 -->
        <dependency>
            <groupId>org.apache.zookeeper</groupId>
            <artifactId>zookeeper</artifactId>
            <version>3.5.6</version>
            <!-- 排除相关日志依赖 - 和dubbo-spring-boot-starter引入的log4j冲突了 -->
            <exclusions>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-api</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-log4j12</artifactId>
                </exclusion>
```

```
            </exclusions>
        </dependency>

        <!-- dubbo+zookeeper需要 -->
        <dependency>
            <groupId>org.apache.curator</groupId>
            <artifactId>curator-framework</artifactId>
            <version>4.0.1</version>
        </dependency>

        <!-- dubbo+zookeeper需要 -->
        <dependency>
            <groupId>org.apache.curator</groupId>
            <artifactId>curator-recipes</artifactId>
            <version>4.0.1</version>
        </dependency>
```

2.yml配置

YAML

```yaml
dubbo:
  consumer:
    check: false
  application:
    name: server-consumer
  registry:
    address: zookeeper://127.0.0.1:2181
    check: false # 启动时是否检查注册中心可用性
```

3.消费者调用服务

Java

```java
@RestController
@RequestMapping("/consumer")
public class ConsumerController {

    @DubboReference
    private ProducerService producerService;

    @GetMapping("/go")
    public String go(String msg){
        producerService.go(msg);
        return "success";
    }


}
```

注意：

- 首先启动zookeeper，然后启动生产者、消费者。
- 服务调用后关闭zookeeper仍可调用，会使用缓存进行调用。