## 使用ZooKeeper实现服务发现

“微服务”大行其道的今天，如果在架构上不使用微服务就显得跟不上时代的脚步。当然也不能只是为了使用而使用，还要看看微服务能为我们解决什么难题。

下面主要介绍微服务能够解决什么样的问题，以及介绍使用ZooKeeper来实现微服务中重要的一部分“服务注册”与“服务发现”。

### 什么是微服务

按照《微服务设计》一书的定义，微服务是指一些协同工作的小而自治的服务。这个解释读起来好像有点拗口，而且不好理解。

按本人的理解，微服务指的是：根据业务场景将一个庞大的系统划分为多个小而独立的服务，而这些服务通过RPC（远程过程调用）来相互通信。这些小而独立的服务都是高度自治的，也就是可以由不同的团队来维护，有完善的监控、容灾和告警等功能。

当然，微服务这个主题完全可以写一本书的，所以这里也不打算过多讨论微服务，下面主要介绍在微服务中扮演着重要角色的“服务发现”。

### 什么是服务发现

服务发现组件记录着分布式系统中所有服务的信息，其他服务可以通过此组件来找到这些服务。

那么服务发现组件是怎么获得系统中所有服务的信息呢？当然可以通过配置文件的方式来获取，但是这种方式的缺点是不灵活，每次新增服务的时候都要修改配置文件的内容，而且在微服务应用中，一次增加上百个服务是很常见的，如果通过手工修改配置文件，那么这将是一件痛苦的事情。现在流行的方案是通过服务注册来通知服务发现组件，这种方案的优点是可以实时让服务发现组件更新服务列表信息，而且不用手工修改任何东西。

接下来我们会介绍使用ZooKeeper服务注册与实现服务发现。

### ZooKeeper的使用

什么是ZooKeeper？官网的解释如下：

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

从官方的介绍可以知道，ZooKeeper可以胜任以下工作：

1. 作为配置中心服务器

2. 命名服务

3. 分布式的协调

4. Master选举等

下面我们来看看怎么使用ZooKeeper实现服务注册与服务发现功能。

ZooKeeper提供类似Unix文件系统形式的存储结构，如下图：

上图中的目录在ZooKeeper中被称为znode，每个znode都由唯一的路径所标识，如Child1这个znode的标识为 **/Parent1/Child1** 。每个znode都可以有子目录（子znode），而且每个znode可以存储数据。

ZooKeeper可以创建一个临时的znode，当创建这个临时znode的客户端与ZooKeeper失联的时候，这个znode就会ZooKeeper自动被删除。我们可以使用这个功能来实现服务注册与服务发现。

下面我们来介绍一下ZooKeeper的使用吧。

**1. ZooKeeper的安装与配置**