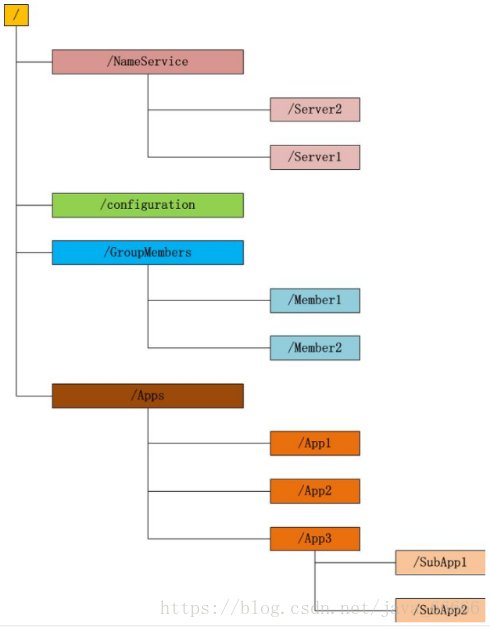
**Zookeeper入门看这篇就够了**

[Zookeeper是什么](https://www.roncoo.com/course/list.html?courseName=Zookeeper)  
  
官方文档上这么解释zookeeper，它是一个分布式服务框架，是Apache Hadoop 的一个子项目，它主要是用来解决分布式应用中经常遇到的一些数据管理问题，如：统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理等。  
  
上面的解释有点抽象，简单来说zookeeper=文件系统+监听通知机制。  
  
1、 文件系统

Zookeeper维护一个类似文件系统的数据结构：



每个子目录项如 NameService 都被称作为 znode(目录节点)，和文件系统一样，我们能够自由的增加、删除znode，在一个znode下增加、删除子znode，唯一的不同在于znode是可以存储数据的。

有四种类型的znode：

* **PERSISTENT-持久化目录节点**

客户端与zookeeper断开连接后，该节点依旧存在

* **PERSISTENT\_SEQUENTIAL-持久化顺序编号目录节点**

客户端与zookeeper断开连接后，该节点依旧存在，只是Zookeeper给该节点名称进行顺序编号

* **EPHEMERAL-临时目录节点**

客户端与zookeeper断开连接后，该节点被删除

* **EPHEMERAL\_SEQUENTIAL-临时顺序编号目录节点**

客户端与zookeeper断开连接后，该节点被删除，只是Zookeeper给该节点名称进行顺序编号

**2、 监听通知机制**

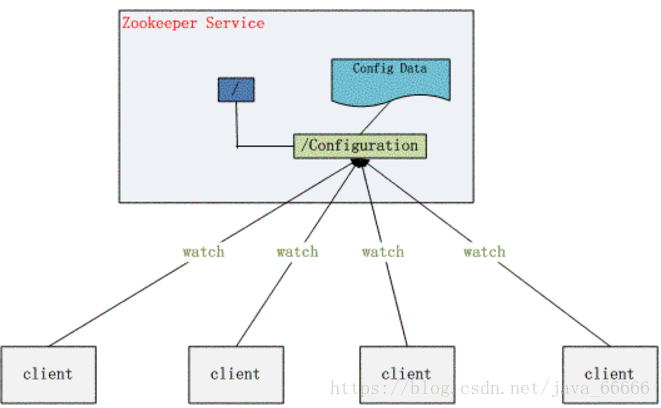
客户端注册监听它关心的目录节点，当目录节点发生变化（数据改变、被删除、子目录节点增加删除）时，zookeeper会通知客户端。

就这么简单，下面我们看看Zookeeper能做点什么呢？

**Zookeeper能做什么**

zookeeper功能非常强大，可以实现诸如分布式应用配置管理、统一命名服务、状态同步服务、集群管理等功能，我们这里拿比较简单的分布式应用配置管理为例来说明。

假设我们的程序是分布式部署在多台机器上，如果我们要改变程序的配置文件，需要逐台机器去修改，非常麻烦，现在把这些配置全部放到zookeeper上去，保存在 zookeeper 的某个目录节点中，然后所有相关应用程序对这个目录节点进行监听，一旦配置信息发生变化，每个应用程序就会收到 zookeeper 的通知，然后从 zookeeper 获取新的配置信息应用到系统中。



如上，你大致应该了解zookeeper是个什么东西，大概能做些什么了，我们马上来学习下zookeeper的安装及使用，并开发一个小程序来实现zookeeper这个分布式配置管理的功能。

**Zookeeper单机模式安装**

**Step1**：配置JAVA环境，检验环境：java -version

**Step2**：下载并解压zookeeper

1. # cd /usr/local
2. # wget http://mirror.bit.edu.cn/apache/zookeeper/stable/zookeeper-3.4.12.tar.gz
3. # tar -zxvf zookeeper-3.4.12.tar.gz
4. # cd zookeeper-3.4.12

**Step3**：重命名配置文件zoo\_sample.cfg

# cp conf/zoo\_sample.cfg conf/zoo.cfg

**Step4**：启动zookeeper

# bin/zkServer.sh start

**Step5**：检测是否成功启动，用zookeeper客户端连接下服务端

# bin/zkCli.sh

**Zookeeper使用**

**使用客户端命令操作zookeeper**

1、使用 ls 命令来查看当前 ZooKeeper 中所包含的内容

https://img-blog.csdn.net/20180712143716167?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2phdmFfNjY2NjY=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

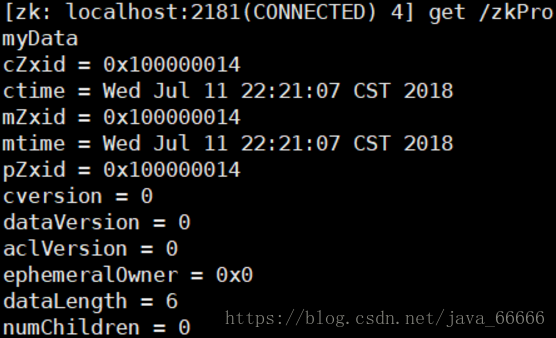
2、创建一个新的 znode ，使用 create /zkPro myData

https://img-blog.csdn.net/2018071214374133?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2phdmFfNjY2NjY=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

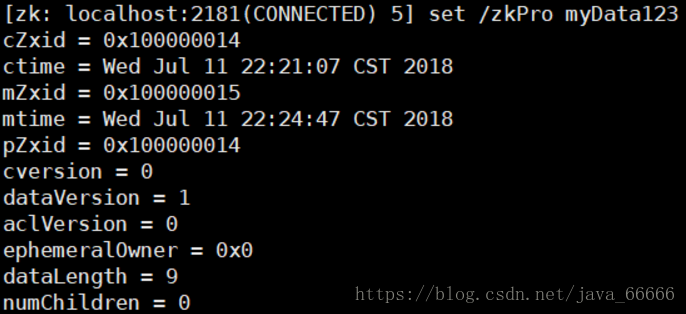
3、再次使用 ls 命令来查看现在 zookeeper 中所包含的内容：

https://img-blog.csdn.net/20180712143805771?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2phdmFfNjY2NjY=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

4、下面我们运行 get 命令来确认第二步中所创建的 znode 是否包含我们所创建的字符串：



5、下面我们通过 set 命令来对 zk 所关联的字符串进行设置：

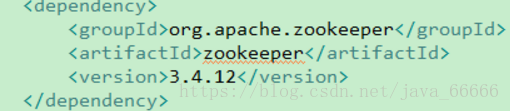


6、下面我们将刚才创建的 znode 删除

https://img-blog.csdn.net/20180712143857281?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2phdmFfNjY2NjY=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

**使用Java API操作zookeeper**

使用Java API操作zookeeper需要引用下面的包



下面我们来实现上面说的分布式配置中心：

1、在zookeeper里增加一个目录节点，并且把配置信息存储在里面

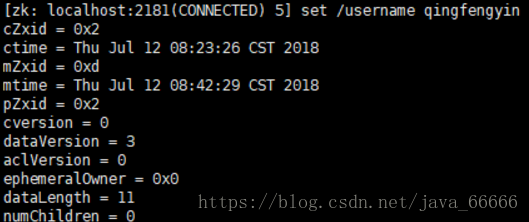
https://img-blog.csdn.net/20180712143937411?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2phdmFfNjY2NjY=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

2、启动两个zookeeper客户端程序，代码如下所示

1. import java.util.concurrent.CountDownLatch;
2. import org.apache.zookeeper.WatchedEvent;
3. import org.apache.zookeeper.Watcher;
4. import org.apache.zookeeper.Watcher.Event.EventType;
5. import org.apache.zookeeper.Watcher.Event.KeeperState;
6. import org.apache.zookeeper.ZooKeeper;
7. import org.apache.zookeeper.data.Stat;
9. /\*\*
10. \* 分布式配置中心demo
11. \* @author
12. \*
13. \*/
14. public class ZooKeeperProSync implements Watcher {
16. private static CountDownLatch connectedSemaphore = new CountDownLatch(1);
17. private static ZooKeeper zk = null;
18. private static Stat stat = new Stat();
20. public static void main(String[] args) throws Exception {
21. //zookeeper配置数据存放路径
22. String path = "/username";
23. //连接zookeeper并且注册一个默认的监听器
24. zk = new ZooKeeper("192.168.31.100:2181", 5000, //
25. new ZooKeeperProSync());
26. //等待zk连接成功的通知
27. connectedSemaphore.await();
28. //获取path目录节点的配置数据，并注册默认的监听器
29. System.out.println(new String(zk.getData(path, true, stat)));
31. Thread.sleep(Integer.MAX\_VALUE);
32. }
34. public void process(WatchedEvent event) {
35. if (KeeperState.SyncConnected == event.getState()) { //zk连接成功通知事件
36. if (EventType.None == event.getType() && null == event.getPath()) {
37. connectedSemaphore.countDown();
38. } else if (event.getType() == EventType.NodeDataChanged) { //zk目录节点数据变化通知事件
39. try {
40. System.out.println("配置已修改，新值为：" + new String(zk.getData(event.getPath(), true, stat)));
41. } catch (Exception e) {
42. }
43. }
44. }
45. }
46. }

两个程序启动后都正确的读取到了zookeeper的/username目录节点下的数据'qingfeng'

3、我们在zookeeper里修改下目录节点/username下的数据



修改完成后，我们看见两个程序后台都及时收到了他们监听的目录节点数据变更后的值，如下所示

https://img-blog.csdn.net/20180712144042598?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2phdmFfNjY2NjY=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

**Zookeeper集群模式安装**

本例搭建的是伪集群模式，即一台机器上启动三个zookeeper实例组成集群，真正的集群模式无非就是实例IP地址不同，搭建方法没有区别

**Step1**：配置JAVA环境，检验环境：java -version

**Step2**：下载并解压zookeeper

1. # cd /usr/local
2. # wget http://mirror.bit.edu.cn/apache/zookeeper/stable/zookeeper-3.4.12.tar.gz
3. # tar -zxvf zookeeper-3.4.12.tar.gz
4. # cd zookeeper-3.4.12

**Step3**：重命名 zoo\_sample.cfg文件

# cp conf/zoo\_sample.cfg conf/zoo-1.cfg

**Step4**：修改配置文件zoo-1.cfg，原配置文件里有的，修改成下面的值，没有的则加上

1. # vim conf/zoo-1.cfg
2. dataDir=/tmp/zookeeper-1
3. clientPort=2181
4. server.1=127.0.0.1:2888:3888
5. server.2=127.0.0.1:2889:3889
6. server.3=127.0.0.1:2890:3890

**配置说明**

* tickTime：这个时间是作为 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。
* initLimit：这个配置项是用来配置 Zookeeper 接受客户端（这里所说的客户端不是用户连接 Zookeeper 服务器的客户端，而是 Zookeeper 服务器集群中连接到 Leader 的 Follower 服务器）初始化连接时最长能忍受多少个心跳时间间隔数。当已经超过 10个心跳的时间（也就是 tickTime）长度后 Zookeeper 服务器还没有收到客户端的返回信息，那么表明这个客户端连接失败。总的时间长度就是 10\*2000=20 秒
* syncLimit：这个配置项标识 Leader 与 Follower 之间发送消息，请求和应答时间长度，最长不能超过多少个 tickTime 的时间长度，总的时间长度就是 5\*2000=10秒
* dataDir：顾名思义就是 Zookeeper 保存数据的目录，默认情况下，Zookeeper 将写数据的日志文件也保存在这个目录里。
* clientPort：这个端口就是客户端连接 Zookeeper 服务器的端口，Zookeeper 会监听这个端口，接受客户端的访问请求。
* server.A=B：C：D：其中 A 是一个数字，表示这个是第几号服务器；B 是这个服务器的 ip 地址；C 表示的是这个服务器与集群中的 Leader 服务器交换信息的端口；D 表示的是万一集群中的 Leader 服务器挂了，需要一个端口来重新进行选举，选出一个新的 Leader，而这个端口就是用来执行选举时服务器相互通信的端口。如果是伪集群的配置方式，由于 B 都是一样，所以不同的 Zookeeper 实例通信端口号不能一样，所以要给它们分配不同的端口号。

**Step4**：再从zoo-1.cfg复制两个配置文件zoo-2.cfg和zoo-3.cfg，只需修改dataDir和clientPort不同即可

1. # cp conf/zoo-1.cfg conf/zoo-2.cfg
2. # cp conf/zoo-1.cfg conf/zoo-3.cfg
3. # vim conf/zoo-2.cfg
4. dataDir=/tmp/zookeeper-2
5. clientPort=2182
6. # vim conf/zoo-2.cfg
7. dataDir=/tmp/zookeeper-3
8. clientPort=2183

**Step5**：标识Server ID

创建三个文件夹/tmp/zookeeper-1，/tmp/zookeeper-2，/tmp/zookeeper-2，在每个目录中创建文件myid 文件，写入当前实例的server id，即1.2.3

1. # cd /tmp/zookeeper-1
2. # vim myid
3. 1
4. # cd /tmp/zookeeper-2
5. # vim myid
6. 2
7. # cd /tmp/zookeeper-3
8. # vim myid
9. 3

**Step6**：启动三个zookeeper实例

1. # bin/zkServer.sh start conf/zoo-1.cfg
2. # bin/zkServer.sh start conf/zoo-2.cfg
3. # bin/zkServer.sh start conf/zoo-3.cfg

**Step7**：检测集群状态，也可以直接用命令“zkCli.sh -server IP:PORT”连接zookeeper服务端检测



至此，我们对zookeeper就算有了一个入门的了解，当然zookeeper远比我们这里描述的功能多，比如用zookeeper实现集群管理，分布式锁，分布式队列，zookeeper集群leader选举等等