# Zookeeper简介

* 中间件，提供协调服务
* 作用于分布式系统，发挥其优势，可以为大数据服务
* 支持java，提供java和c语言的客户端api

# 什么是分布式系统

* 很多台计算机组成一个整体，一个整体一致对外并且处理同一请求
* 内部的每台计算机都可以互相通信(rest/rpc)
* 客户端到服务端的一次请求到相应结束会历经多台计算机

# Zookeeper的特性

* 一致性：数据一致性，数据按照顺序分批入库
* 原子性：事务要么成功要么失败，不会局部化
* 单一视图：客户端连接集群中的任一zk节点，数据都是一致的
* 可靠性：每次对zk的操作状态都会保存在服务端
* 实时性：客户端可以读取到zk服务端的最新数据

# Zookeeper目录详解

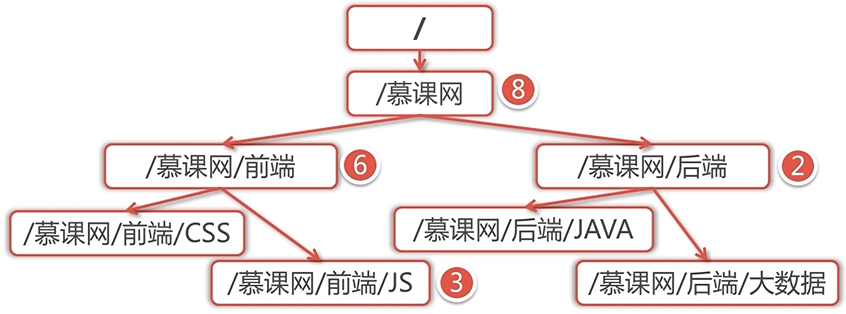
* bin：主要的一些运行命令
* conf：存放配置文件，其中我们需要修改的zk.conf
* contrib：附加的一些功能
* dist-maven：mvn编译后的目录
* docs：文档
* lib：需要依赖的jar包
* recipes：案例demo代码
* src：源码

# zoo.cfg配置

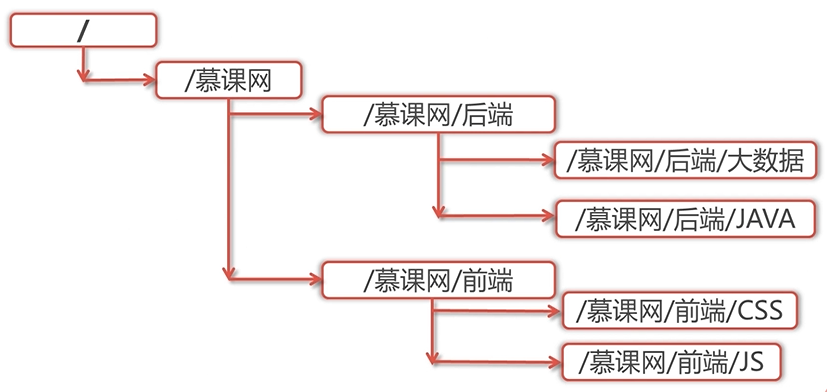
* tickTime：用于计算的时间单元。比如session超时：N\*tickTime
* initLimit：用于集群，允许从节点连接并同步到master节点的初始化连接时间，以tickTime的倍数来表示
* syncLimit：用于集群，master主节点与从节点之间发送消息，请求和应答时间长度。（心跳机制）
* dataDir：必须配置
* dataLogDir：日志目录，如果不配置会和dataDir公用
* clientPort：连接服务器的端口，默认2181

# zookeeper基本数据模型介绍一

* 是一个树形结构，类似于前端开发中的tree.js组件



# zookeeper基本数据模型介绍二



# zookeeper基本数据模型介绍三

* Zk的数据模型也可以理解为Linux、Unix的文件目录：/usr/local/…
* 每一个节点都称之为znode，它可以有子节点，也可以有数据
* 每一个节点分为临时节点和永久节点，临时节点在客户端断开后消失

# zookeeper基本数据模型介绍四

* 每个zk节点都有各自的版本号，可以通过命令行来显示节点信息
* 每当节点数据发生变化，那么该节点的版本号会累加（乐观锁）
* 删除/修改过时节点，版本号不匹配则报错

# zookeeper基本数据模型介绍五

* 每个zk节点存储的数据不宜过大，几K即可
* 节点可以设置权限acl，可以通过权限来限制用户的访问

# zookeeper基本数据模型基本操作

* 客户端连接
* 查看znode结构
* 关闭客户端连接

# zk的作用体现

* Master节点选举，主节点挂了以后，从节点就会接手工作，并且保证这个节点是唯一的，这也是所谓首脑模式，从而保证我们的集群是高可用的
* 统一配置文件管理，即只需要部署一台服务器，则可以把相同的配置文件同步更新到其他所有服务器，此操作在云计算中用的特别多(假设修改了Redis统一配置)
* 发布与订阅，类似消息队列MQ，dubbo发布者吧数据存在znode上，订阅者会读取这个数据
* 提供分布式锁，分布式环境中不同进程之间争夺资源，类似于多线程中的锁
* 集群管理，集群中保证数据的强一致性

# Zk常用的命令行操作1

* 通过./zkCli.sh打开zk的客户端进行命令行后台
* ls与ls2命令 ls (path)
* get与stat命令

# zk特性-session的基本原理

* 客户端与服务端之间的连接存在会话
* 每个会话都会可以设置一个超时时间
* 心跳结束，session则过期
* Session过期，则临时节点znode会被抛弃
* 心跳机制：客户端向服务端的ping包请求

# Zk常用的命令行操作2

* create命令 create -s /imooc/seq创建顺序节点，create -e /imooc/tmp创建临时节点。
* set命令 set [path]/imooc [data]new-data [version]1是一个乐观锁
* delete命令delete [path]/imooc [version]1，和set命令一样加一个版本号可以实现乐观锁。

# Zk特性-watcher机制一

* 针对每个节点的操作，都会有一个监督者->watcher
* 当监控的某个对象(znode)发生了变化，则触发watcher事件
* Zk的watcher是一次性的，触发后立即销毁

# Zk特性-watcher机制二

* 父节点，子节点增删改查都能够触发其watcher
* 针对不同类型的操作，触发的watcher事件也不同：

1. (子)节点创建事件
2. (子)节点删除事件
3. (子)节点数据变化事件