1. 集群的部署回顾
2. 数据模型（节点的特性）
3. 会话的概念
4. 数据节点的信息（stat）
5. create/delete/set/get 操作zookeeper的数据节点
6. java api的使用（事件响应，watcher）

# java api的使用

*Zookeeper作为一个分布式协调框架，其内部存储的都是一些关于分布式  
系统运行时状态的元数据，尤其是设计到一些分布式锁，Master选举和协调等应用场景。我们需要有  
效地保障Zookeeper中的数据安全，Zookeeper提供了三种模式。权限模式，授权对象，权限*

权限控制模式

schema：授权对象

ip : ip模式通过ip地址粒度进行权限控制模式，例如配置了：192.168.110.135即表示权限控

制都是针对这个ip地址的，同时也支持按网段分配，比如：192.168.110.\*

Digest : digest是最常用的权限控制模式，也更符合我们对权限控制的认识，其类似于

"username:password"形式的权限标识进行权限配置。ZK会对形成的权限标识先后进

行两次编码处理，粉笔是SHA-1加密算法和Base64编码

world : 开放式的权限控制模式，数据节点的访问权限对所有用户开放。 world:anyone

super ：超级用户，可以对zookeeper上的数据节点进行操作

权限对象：值得是权限赋予的用户或者是一个指定的实体，例如ip地址或机器等。在不同的模式下，

授权对象是不同的。这种模式和权限对象一一对应。

权限：权限就是指那些通过权限检测后可以被允许执行的操作，在ZK中，对数据的操作权限分为以下

五大类：create,delete,read,write,admin

## 连接状态

KeeperStat.Expired 在一定时间内客户端没有收到服务器的通知， 则认为当前的会话已经过期了。

KeeperStat.Disconnected 断开连接的状态

KeeperStat.SyncConnected 客户端和服务器端在某一个节点上建立连接，并且完成一次version、zxid同步

KeeperStat.authFailed 授权失败

## 事件类型

NodeCreated 当节点被创建的时候，触发

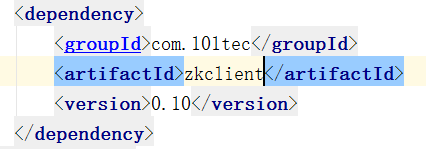
NodeChildrenChanged 表示子节点被创建、被删除、子节点数据发生变化

NodeDataChanged 节点数据发生变化

NodeDeleted 节点被删除

None 客户端和服务器端连接状态发生变化的时候，事件类型就是None

# zkclient



# curator

Curator本身是Netflix公司开源的zookeeper客户端；

curator提供了各种应用场景的实现封装

curator-framework 提供了fluent风格api

curator-replice 提供了实现封装

## curator连接的重试策略

ExponentialBackoffRetry() 衰减重试

RetryNTimes 指定最大重试次数

RetryOneTime 仅重试一次

RetryUnitilElapsed 一直重试知道规定的时间

# zookeeper的实际应用场景

zookeeper能够实现哪些场景

订阅发布

watcher机制

统一配置管理（disconf）

分布式锁

redis

zookeeper

数据库

负载均衡

ID生成器

分布式队列

统一命名服务

master选举

分布式锁

master选举