[**ZooKeeper 数据模型**](http://coolxing.iteye.com/blog/1871328)

**博客分类：**

* [zookeeper](http://coolxing.iteye.com/category/278800)

[zookeeper](http://www.iteye.com/blogs/tag/zookeeper)[java](http://www.iteye.com/blogs/tag/java)[分布式](http://www.iteye.com/blogs/tag/%E5%88%86%E5%B8%83%E5%BC%8F)

[转载请注明作者和[原文链接](http://coolxing.iteye.com/blog/1871328),  如有谬误, 欢迎在评论中指正. ]

ZooKeeper的数据结构, 与普通的文件系统极为类似. 见下图:



图片引用自[developerworks](http://www.ibm.com/developerworks/cn/opensource/os-cn-zookeeper/)

图中的每个节点称为一个znode. 每个znode由3部分组成:

* stat. 此为状态信息, 描述该znode的版本, 权限等信息.
* data. 与该znode关联的数据.
* children. 该znode下的子节点.

**ZooKeeper命令**

在深入znode的各个部分之前, 首先需要熟悉一些常用的ZooKeeper命令.

连接server

**Bash代码  [收藏代码](javascript:void())**

1. bin/zkCli.sh -server 10.1.39.43:4180

列出指定node的子node

**Bash代码  [收藏代码](javascript:void())**

1. [zk: 10.1.39.43:4180(CONNECTED) 9] ls /
2. [hello, filesync, zookeeper, xing, server, group, log]
3. [zk: 10.1.39.43:4180(CONNECTED) 10] ls /hello
4. []

创建znode节点, 并指定关联数据

**Bash代码  [收藏代码](javascript:void())**

1. create /hello world

创建节点/hello, 并将字符串"world"关联到该节点中.

获取znode的数据和状态信息

**Bash代码  [收藏代码](javascript:void())**

1. [zk: 10.1.39.43:4180(CONNECTED) 7] get /hello
2. world
3. cZxid = 0x10000042c
4. ctime = Fri May 17 17:57:33 CST 2013
5. mZxid = 0x10000042c
6. mtime = Fri May 17 17:57:33 CST 2013
7. pZxid = 0x10000042c
8. cversion = 0
9. dataVersion = 0
10. aclVersion = 0
11. ephemeralOwner = 0x0
12. dataLength = 5
13. numChildren = 0

删除znode

**Bash代码  [收藏代码](javascript:void())**

1. [zk: localhost:4180(CONNECTED) 13] delete /xing/item0000000001
2. [zk: localhost:4180(CONNECTED) 14] delete /xing
3. Node not empty: /xing

使用delete命令可以删除指定znode. 当该znode拥有子znode时, 必须先删除其所有子znode, 否则操作将失败. rmr命令可用于代替delete命令, rmr是一个递归删除命令, 如果发生指定节点拥有子节点时, rmr命令会首先删除子节点.

**znode节点的状态信息**

使用get命令获取指定节点的数据时, 同时也将返回该节点的状态信息, 称为Stat. 其包含如下字段:

* czxid. 节点创建时的zxid.
* mzxid. 节点最新一次更新发生时的zxid.
* ctime. 节点创建时的时间戳.
* mtime. 节点最新一次更新发生时的时间戳.
* dataVersion. 节点数据的更新次数.
* cversion. 其子节点的更新次数.
* aclVersion. 节点ACL(授权信息)的更新次数.
* ephemeralOwner. 如果该节点为ephemeral节点, ephemeralOwner值表示与该节点绑定的session id. 如果该节点不是ephemeral节点, ephemeralOwner值为0. 至于什么是ephemeral节点, 请看后面的讲述.
* dataLength. 节点数据的字节数.
* numChildren. 子节点个数.

**zxid**

znode节点的状态信息中包含czxid和mzxid, 那么什么是zxid呢?  
ZooKeeper状态的每一次改变, 都对应着一个递增的Transaction id, 该id称为zxid. 由于zxid的递增性质, 如果zxid1小于zxid2, 那么zxid1肯定先于zxid2发生. 创建任意节点, 或者更新任意节点的数据, 或者删除任意节点, 都会导致Zookeeper状态发生改变, 从而导致zxid的值增加.

**session**

在client和server通信之前, 首先需要建立连接, 该连接称为session. 连接建立后, 如果发生连接超时, 授权失败, 或者显式关闭连接, 连接便处于CLOSED状态, 此时session结束.

**节点类型**

讲述节点状态的ephemeralOwner字段时, 提到过有的节点是ephemeral节点, 而有的并不是. 那么节点都具有哪些类型呢? 每种类型的节点又具有哪些特点呢?  
persistent. persistent节点不和特定的session绑定, 不会随着创建该节点的session的结束而消失, 而是一直存在, 除非该节点被显式删除.  
ephemeral. ephemeral节点是临时性的, 如果创建该节点的session结束了, 该节点就会被自动删除. ephemeral节点不能拥有子节点. 虽然ephemeral节点与创建它的session绑定, 但只要该该节点没有被删除, 其他session就可以读写该节点中关联的数据. 使用-e参数指定创建ephemeral节点.

**Bash代码  [收藏代码](javascript:void())**

1. [zk: localhost:4180(CONNECTED) 4] create -e /xing/ei world
2. Created /xing/ei

sequence. 严格的说, sequence并非节点类型中的一种. sequence节点既可以是ephemeral的, 也可以是persistent的. 创建sequence节点时, ZooKeeper server会在指定的节点名称后加上一个数字序列, 该数字序列是递增的. 因此可以多次创建相同的sequence节点, 而得到不同的节点. 使用-s参数指定创建sequence节点.

**Bash代码  [收藏代码](javascript:void())**

1. [zk: localhost:4180(CONNECTED) 0] create -s /xing/item world
2. Created /xing/item0000000001
3. [zk: localhost:4180(CONNECTED) 1] create -s /xing/item world
4. Created /xing/item0000000002
5. [zk: localhost:4180(CONNECTED) 2] create -s /xing/item world
6. Created /xing/item0000000003
7. [zk: localhost:4180(CONNECTED) 3] create -s /xing/item world
8. Created /xing/item0000000004

**watch**

watch的意思是监听感兴趣的事件. 在命令行中, 以下几个命令可以指定是否监听相应的事件.

ls命令. ls命令的第一个参数指定znode, 第二个参数如果为true, 则说明监听该znode的子节点的增减, 以及该znode本身的删除事件.

**Bash代码  [收藏代码](javascript:void())**

1. [zk: localhost:4180(CONNECTED) 21] ls /xing true
2. []
3. [zk: localhost:4180(CONNECTED) 22] create /xing/item item000
5. WATCHER::
7. WatchedEvent state:SyncConnected type:NodeChildrenChanged path:/xing
8. Created /xing/item

get命令. get命令的第一个参数指定znode, 第二个参数如果为true, 则说明监听该znode的更新和删除事件.

**Bash代码  [收藏代码](javascript:void())**

1. [zk: localhost:4180(CONNECTED) 39] get /xing true
2. world
3. cZxid = 0x100000066
4. ctime = Fri May 17 22:30:01 CST 2013
5. mZxid = 0x100000066
6. mtime = Fri May 17 22:30:01 CST 2013
7. pZxid = 0x100000066
8. cversion = 0
9. dataVersion = 0
10. aclVersion = 0
11. ephemeralOwner = 0x0
12. dataLength = 5
13. numChildren = 0
14. [zk: localhost:4180(CONNECTED) 40] create /xing/item item000
15. Created /xing/item
16. [zk: localhost:4180(CONNECTED) 41] rmr /xing
18. WATCHER::
20. WatchedEvent state:SyncConnected type:NodeDeleted path:/xing

stat命令. stat命令用于获取znode的状态信息. 第一个参数指定znode, 如果第二个参数为true, 则监听该node的更新和删除事件.