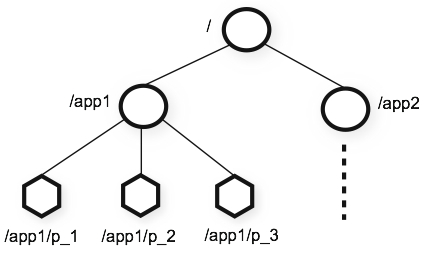
# Zookeeper核心概念

分布式配置中心（存储）：disconf（zk） diamond（mysql+http）

## 什么是znode



ZooKeeper操作和维护的为一个个数据节点，称为 znode，采用类似文件系统的层级树状结构进行管理。如果 znode 节点包含数据则存储为字节数组（byte array）。

创建 znode 时需要指定节点类型

znode 共有 4 种类型，分别为：持久（无序）、临时（无序）、持久有序和临时有序。

## 节点类型：

2大类、四种类型 持久、临时、持久有序、临时有序

PERSISTENT 持久类型，如果不手动删除 是一直存在的

PERSISTENT\_SEQUENTIAL

EPHEMERAL 临时 客户端session失效就会随着删除节点 没有子节点

EPHEMERAL\_SEQUENTIAL 有序 自增

**Stat数据结构：**

Stat中记录了这个 ZNode 的三个数据版本，分别是version（当前ZNode的版本）、cversion（当前ZNode子节点的版本）和 cversion（当前ZNode的ACL版本）。

Stat：状态信息、版本、权限相关

|  |  |
| --- | --- |
| 状态属性 | 说明 |
| 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 |
| dataLength | 节点数据的字节数. |
| numChildren | 子节点个数. |

Session会话：

客户端来创建一个和zk服务端连接的句柄。

连接状态：CONNECTING\CONNECTED\CLOSED

## 什么是watcher

Watcher（事件监听器），是Zookeeper中的一个很重要的特性。Zookeeper允许用户在指定节点上注册一些Watcher，并且在一些特定事件触发的时候，ZooKeeper服务端会将事件通知到感兴趣的客户端上去，该机制是Zookeeper实现分布式协调服务的重要特性。

Watcher：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KeeperState** | **EventType** | **触发条件** | **说明** | **操作** |
| SyncConnected （3） | None （-1） | 客**户端与服务端成功建立连接** | 此时客户端和服务器处于连接状态 |  |
| NodeCreated（1） | Watcher监听的对应数据节点被创建 | Create |
| NodeDeleted （2） | Watcher监听的对应数据节点被删除 | Delete/znode |
| NodeDataChanged （3） | Watcher监听的对应数据节点的数据内容发生变更 | setDate/znode |
| NodeChildChanged （4） | Wather监听的对应数据节点的子节点列表发生变更 | Create/child |
| Disconnected （0） | None （-1） | 客户端与ZooKeeper服务器断开连接 | 此时客户端和服务器处于断开连接状态 |  |
| Expired （-112） | None （-1） | 会话超时 | 此时客户端会话失效，通常同时也会受到SessionExpiredException异常 |  |
| AuthFailed （4） | None （-1） | 通常有两种情况，1：使用错误的schema进行权限检查 2：SASL权限检查失败 | 通常同时也会收到AuthFailedException异常 |  |

## 接口

## 什么是ACL

ACL(Access Control List)

****内置的 ACL schemes**：**

**world**：默认方式，相当于全世界都能访问  
**auth**：代表已经认证通过的用户(cli中可以通过addauth digest user:pwd 来添加当前上下文中的授权用户)  
**digest**：即用户名:密码这种方式认证，这也是业务系统中最常用的  
**ip**：使用Ip地址认证

**ACL支持权限：**

CREATE: 能创建子节点

READ：能获取节点数据和列出其子节点

WRITE: 能设置节点数据

DELETE: 能删除子节点

ADMIN: 能设置权限

### 高性能

ZooKeeper 是高性能的。 在“读”多于“写”的应用程序中尤其地高性能，因为“写”会导致所有的服务器间同步状态。（“读”多于“写”是协调服务的典型场景。）

### 顺序访问

对于来自客户端的每个更新请求，ZooKeeper 都会分配一个全局唯一的递增编号，这个编号反应了所有事务操作的先后顺序，应用程序可以使用 ZooKeeper 这个特性来实现更高层次的同步原语。 这个编号也叫做时间戳——zxid（Zookeeper Transaction Id）

# Zookeeper的Java客户端API

## 原生：

|  |
| --- |
| <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  <version>3.4.9</version> </dependency> |

## org.apache.zookeeper.ZooKeeperMain

org.apache.zookeeper.CreateMode

## ZkClient

|  |
| --- |
| <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>0.10</version>  </dependency> |

## Curator

|  |
| --- |
| <dependency>  <groupId>org.apache.curator</groupId>  <artifactId>curator-recipes</artifactId>  <version>4.0.0</version> </dependency>  <dependency>  <groupId>org.apache.curator</groupId>  <artifactId>curator-framework</artifactId>  <version>4.0.0</version> </dependency> |

# 作业

Zookeeper原生的api操作zk

CRUD WATCHER znode节点玩下