咕泡学院 VIP 课：zookeeper 的实践指南

课程目标

1. 数据存储
2. 基于 Java API 初探 zookeeper 的使用
3. 深入分析 Watcher 机制的实现原理
4. Curator 客户端的使用，简单高效

数据存储

事务日志

快照日志

运行时日志　bin/zookeeper.out

基于 Java API 初探 zookeeper 的使用

建立连接的过程

```
public static void main(String[] args) {
        try {
            final CountDownLatch countDownLatch=new CountDownLatch(1);
            ZooKeeper zooKeeper=
                    new ZooKeeper("192.168.11.153:2181," +
                    "192.168.11.154:2181,192.168.11.155:2181",
                            4000, new Watcher() {
                        @Override
                        public void process(WatchedEvent event) {
                            if(Event.KeeperState.SyncConnected==event.getState()){
                                //如果收到了服务端的响应事件，连接成功
```

```
countDownLatch.countDown();
                                    }
                                }
                            });
                    countDownLatch.await();

System.out.println(zooKeeper.getState());//CONNECTING


                    zooKeeper.close();
            } catch (IOException e) {
                    e.printStackTrace();
            } catch (InterruptedException e) {
                    e.printStackTrace();
            }
    }
```

数据的增删改查操作

```
//添加节点
zooKeeper.create("/zk-persis-
```

```
mic","0".getBytes(),ZooDefs.Ids.OPEN_ACL_UNS
AFE,CreateMode.PERSISTENT);
Thread.sleep(1000);
Stat stat=new Stat();

//得到当前节点的值
byte[] bytes=zooKeeper.getData("/zk-persis-
mic",null,stat);
System.out.println(new String(bytes));

//修改节点值
zooKeeper.setData("/zk-persis-
mic","1".getBytes(),stat.getVersion());

//得到当前节点的值
byte[] bytes1=zooKeeper.getData("/zk-persis-
mic",null,stat);
System.out.println(new String(bytes1));

zooKeeper.delete("/zk-persis-
mic",stat.getVersion());
```

```
zooKeeper.close();


System.in.read();
```

事件机制

Watcher 监听机制是 Zookeeper 中非常重要的特性，我们基于 zookeeper 上创建的节点，可以对这些节点绑定监听事件，比如可以监听节点数据变更、节点删除、子节点状态变更等事件，通过这个事件机制，可以基于 zookeeper 实现分布式锁、集群管理等功能

watcher 特性：当数据发生变化的时候， zookeeper 会产生一个 watcher 事件，并且会发送到客户端。但是客户端只会收到一次通知。如果后续这个节点再次发生变化，那么之前设置 watcher 的客户端不会再次收到消息。（watcher 是一次性的操作）。 可以通过循环监听去达到永久监听效果

如何注册事件机制

通过这三个操作来绑定事件 ：getData、Exists、getChildren

如何触发事件？ 凡是事务类型的操作，都会触发监听事件。

create /delete /setData

watcher 事件类型

None (-1)， 客户端链接状态发生变化的时候，会收到 none 的事件

NodeCreated (1)， 创建节点的事件。 比如 zk-persis-mic

NodeDeleted (2)， 删除节点的事件

NodeDataChanged (3)， 节点数据发生变更

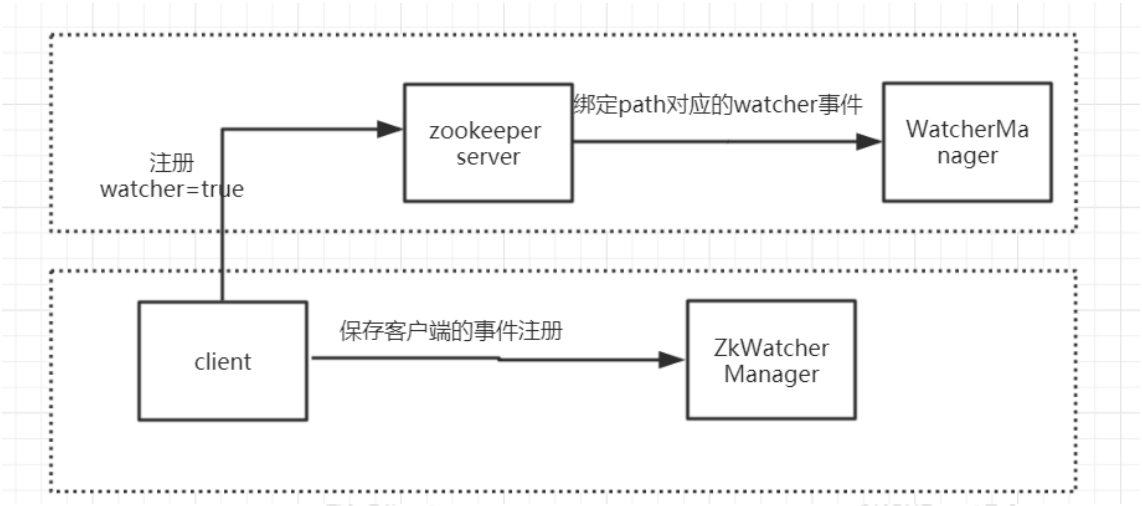
NodeChildrenChanged (4)； 子节点被创建、被删除、会发生事件触发

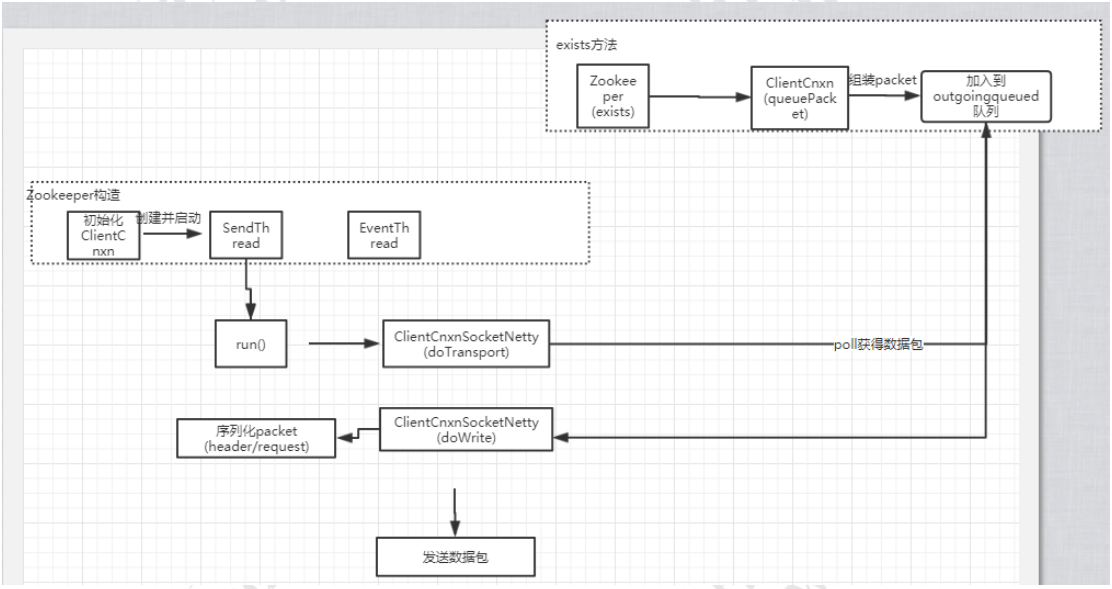
什么样的操作会产生什么类型的事件呢？

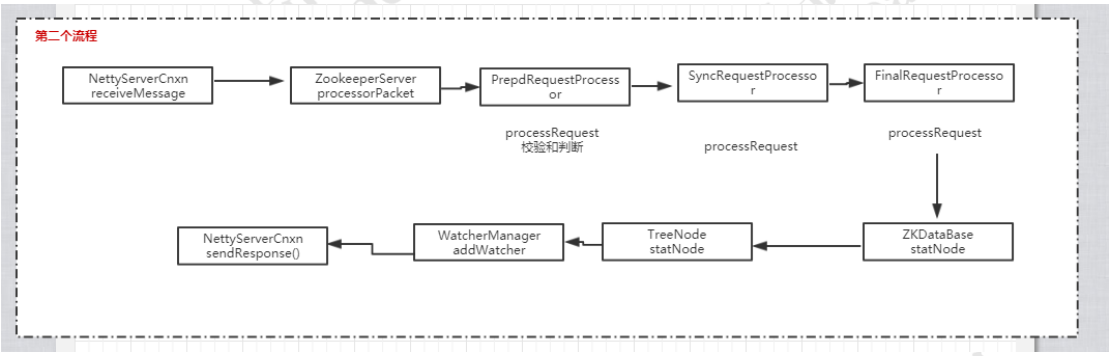
	zk-persis-mic （监听事件）	zk-persis-mic/child （监听事件）
create(/ zk-	NodeCreated(exists /	无

persis-mic)	getData)	
delete(/ zk-persis-mic)	NodeDeleted(exists / getData)	无
setData(/ zk-persis-mic)	NodeDataChanged(exists / getData)	无
create （/zk-persis-mic/children）	NodeChildrenChanged (getchild)	NodedCreated
delete （/zk-persis-mic/children）	NodeChildrenChanged (getchild)	NodedDeleted
setData （/zk-persis-mic/children）		NodeDataChanged

事件的实现原理


zookeeper server
绑定path对应的watcher事件
WatcherManager
注册
watcher=true
client
保存客户端的事件注册
ZkWatcher Manager


exists方法
Zookeeper (exists)
ClientCnxn (queuePacket)
组装packet
加入到outgoingqueued队列
Zookeeper构造
初始化ClientCnxn
创建并启动
SendThread
EventThread
run()
ClientCnxnSocketNetty (doTransport)
poll获得数据包
序列化packet (header/request)
ClientCnxnSocketNetty (doWrite)
发送数据包


第二个流程
NettyServerCnxn receiveMessage
ZookeeperServer processorPacket
PrepdRequestProcessor
SyncRequestProcessor
FinalRequestProcessor
processRequest 校验和判断
processRequest
processRequest
ZKDataBase statNode
TreeNode statNode
WatcherManager addWatcher
NettyServerCnxn sendResponse()