Zookeeper分布式系统开发实战 第2课

DATAGURU专业数据分析社区

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

第二讲 基础进阶---Znode

- 数据节点（Znode）
  - 不是机器的意思
  - Zk树形结构中的数据节点，用于存储数据
  - 持久节点(PERSISTENT)：一旦创建，除非主动调用删除操作，否则一直存储在zk上
  - 临时节点(EPHEMERAL)：与客户端的会话绑定，一旦客户端会话失效，这个客户端创建的所有临时节点都会被移除
  - PERSISTENT _SEQUENTIAL ：创建子节点时，如果设置属性SEQUENTIAL，则会自动在节点名后面追加一个整型数字,上限是整形的最大值

第二讲 基础进阶---临时节点

```
[zk: localhost:2181(CONNECTED) 6] ls /
[zk-book2, zk-book, zookeeper, zk-createtest]
```

1

```
[zk: localhost:2181(CONNECTED) 0] create -e /zk-createtest/ephemeraltest 111
Created /zk-createtest/ephemeraltest
[zk: localhost:2181(CONNECTED) 1]
```

2

```
[zk: localhost:2181(CONNECTED) 7] ls /zk-createtest
[ephemeraltest]
[zk: localhost:2181(CONNECTED) 8]
```

3

```
[zk: localhost:2181(CONNECTED) 2] ls /zk-createtest
[]
[zk: localhost:2181(CONNECTED) 3]
```

当创建会话关闭后，临时节点消失

第二讲 基础进阶---顺序节点

- 创建顺序节点1，节点数据为321
- 创建顺序节点2，节点数据内容为322
- 查看所有创建的顺序子节点

```
[zk: localhost:2181(CONNECTED) 2] create -s /zk-createtest/seq  321
Created /zk-createtest/seq0000000002
[zk: localhost:2181(CONNECTED) 3] create -s /zk-createtest/seq  322
Created /zk-createtest/seq0000000003
[zk: localhost:2181(CONNECTED) 4] ls /zk-createtest
[seq0000000002, 0000000001, seq0000000003]
[zk: localhost:2181(CONNECTED) 5]
```

- 创建顺序节点，不带前缀

```
[zk: localhost:2181(CONNECTED) 8] create -s /zk-createtest/seq2/   seqtest
Created /zk-createtest/seq2/0000000000
[zk: localhost:2181(CONNECTED) 9] create -s /zk-createtest/seq2/   seqtest1
Created /zk-createtest/seq2/0000000001
[zk: localhost:2181(CONNECTED) 10] create -s /zk-createtest/seq2/   seqtest2
Created /zk-createtest/seq2/0000000002
[zk: localhost:2181(CONNECTED) 11] create -s /zk-createtest/seq2/   seqtest3
Created /zk-createtest/seq2/0000000003
[zk: localhost:2181(CONNECTED) 12] ls /zk-createtest/seq2
[0000000000, 0000000001, 0000000002, 0000000003]
[zk: localhost:2181(CONNECTED) 13]
```

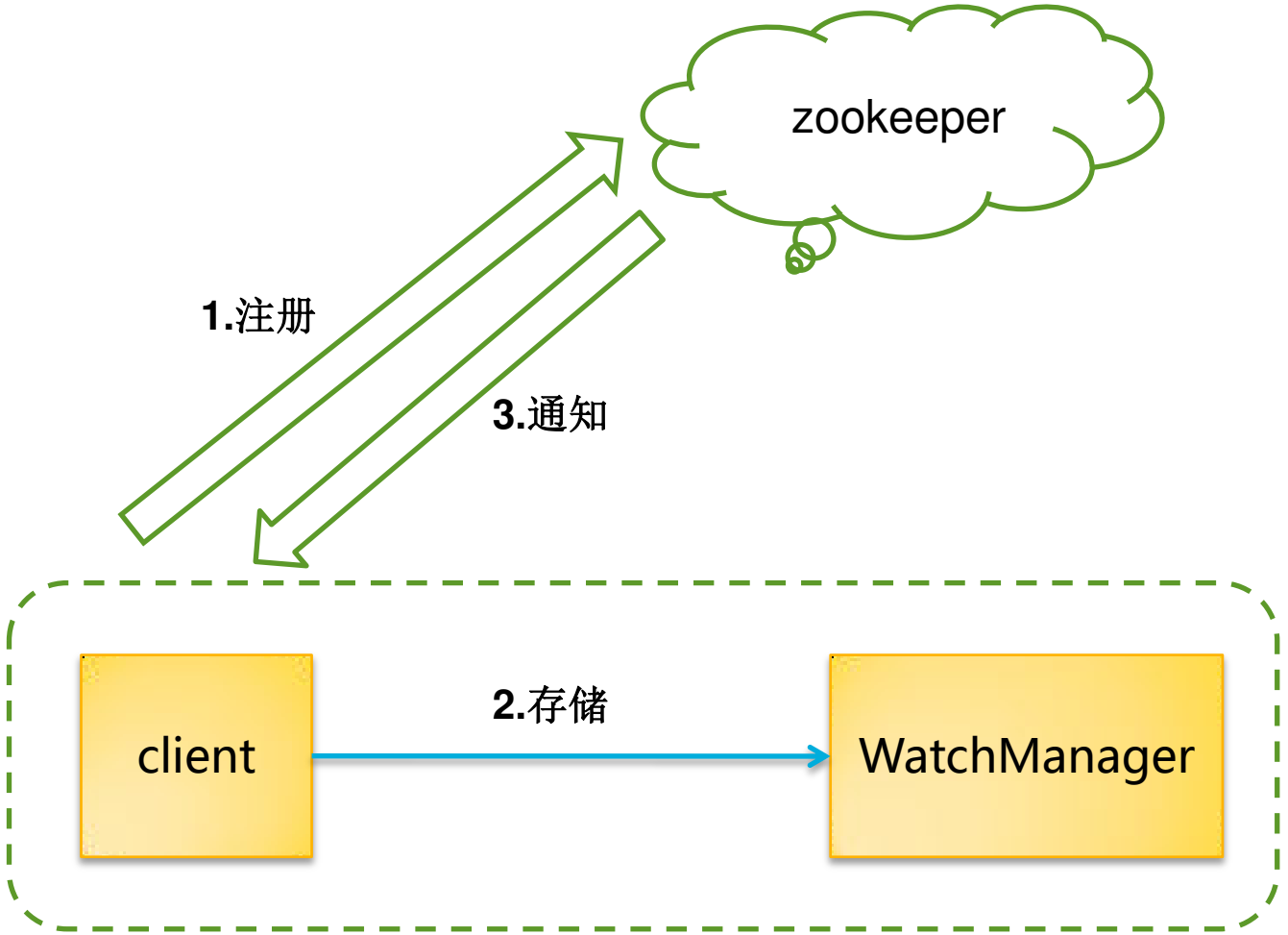
第二讲 zk进阶---watcher

- 问题
  - 集群中有多个机器，当某个通用的配置发生变化后，怎么让所有服务器的配置都统一生效？
  - 当某个集群节点宕机，其它节点怎么知道？

Zk中引入了watcher机制来实现了发布/订阅功能，能够让多个订阅者同时监听某一个主题对象，当这个主题对象自身状态变化时，会通知所有订阅者；

第二讲 zk进阶---watcher

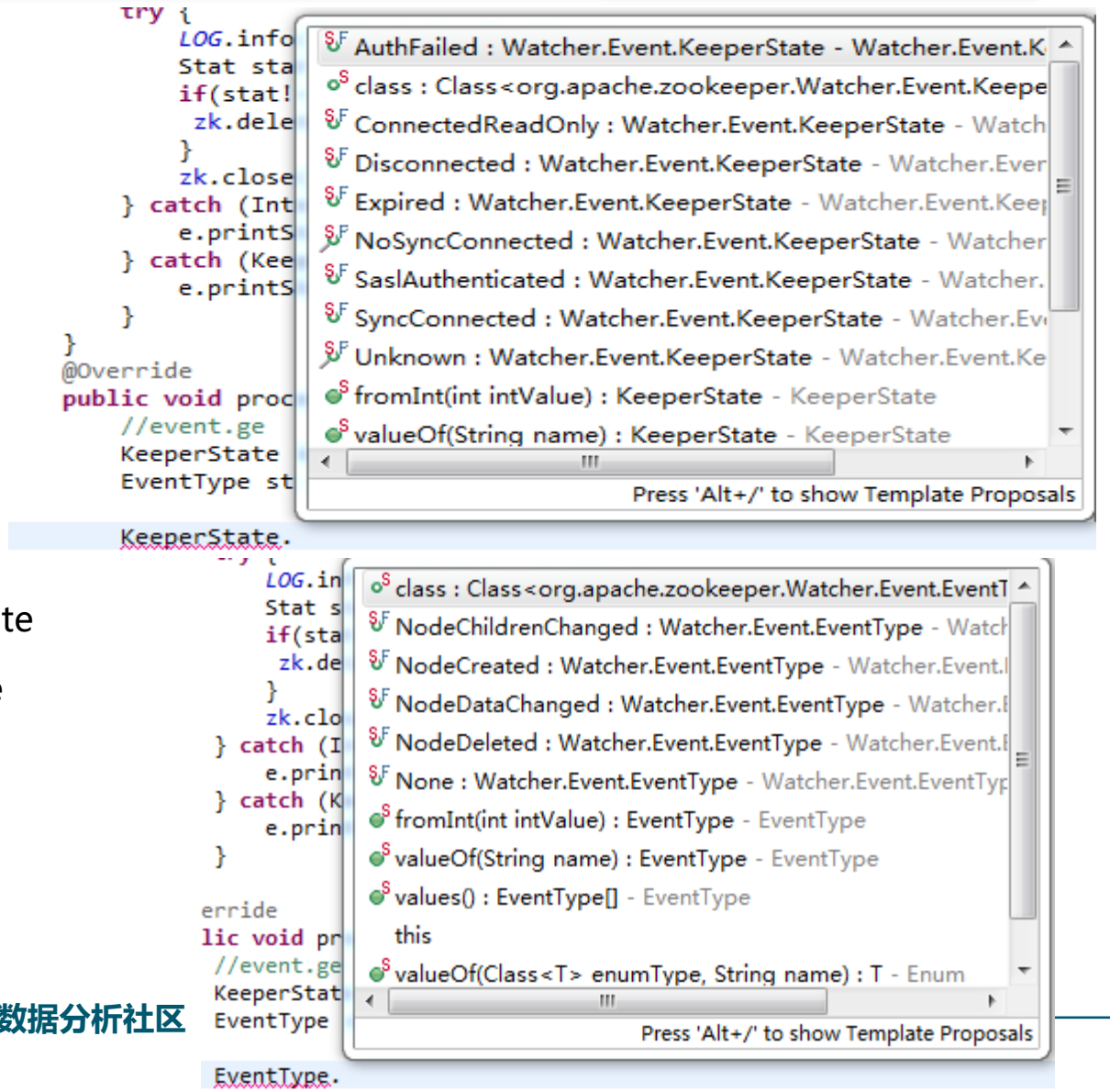
- Watcher组成
  - 客户端
  - 客户端watchManager
  - Zk服务器
- Watcher机制
  - 客户端向zk服务器注册watcher的同时，会将watcher对象存储在客户端的watchManager
  - Zk服务器触发watcher事件后，会向客户端发送通知，客户端线程从watchManager中掉起watcher执行



第二讲 zk进阶---watcher

- Watcher接口
  - **public class ZLock implements Watcher**
  - **public void process(WatchedEvent event)**

- Watcher事件
  - 通知状态：org.apache.zookeeper.Watcher.Event.KeeperState
  - 事件类型：org.apache.zookeeper.Watcher.Event.EventType



第二讲 zk进阶---watcher

keeperState	EventType	触发条件	说明
SyncConnected	None(-1)	客户端与服务器成功建立会话	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	
	NodeDeleted(2)	Watcher监听的对应数据节点被删除	
	NodeDataChanged(3)	数据节点的数据内容发生变更	
	NodeChildrenChanged(4)	被监听的数据节点的字节点列表发生变更	
Disconnected(0)	None(-1)	客户端与zk服务器端口连接	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，通常会收到SessionExpiredException异常
AuthFailed(4)	None(-1)	1.使用错误的scheme进行权限检查 2.SASL权限检查失败	通常会收到AuthFailedException异常
Unknown(-1)			从3.1.0版本开始已经废弃
NoSyncConnected			

第二讲 zk进阶---watcher

- NodeDataChanged事件
  - 无论节点数据发生变化还是数据版本发生变化都会触发
  - 即使被更新数据与新数据一样，数据版本都会发生变化

```
[zk: localhost:2181(CONNECTED) 1] get /zk-book2
[zk: localhost:2181(CONNECTED) 2] set /zk-book2 123
cZxid = 0x7
ctime = Sun Nov 01 23:36:00 CST 2015
mZxid = 0x30
mtime = Sun Nov 08 12:47:27 CST 2015
pZxid = 0x7
cversion = 0
dataVersion = 1
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 3
numChildren = 0
```

- NodeChildrenChanged
  - 新增节点或者删除节点
- AuthFailed
  - 重点不是客户端会话没有权限而是授权失败（后面再细讲）

客户端只能收到相关事件通知，但是并不能获取到对应数据节点的原始数据内容以及变更后的心数据内容；因此，如果业务需要知道变更前的数据或者变更后的新数据，需要业务保存变更前的数据和调用接口获取新的数据

第二讲 zk进阶---watcher

- 创建zk客户端对象实例时注册
  - ZooKeeper(String connectString, int sessionTimeout, Watcher watcher)
  - ZooKeeper(String connectString, int sessionTimeout, Watcher watcher, boolean canBeReadOnly)
  - ZooKeeper(String connectString, int sessionTimeout, Watcher watcher, long sessionId, byte[] sessionPasswd)
  - ZooKeeper(String connectString, int sessionTimeout, Watcher watcher, long sessionId, byte[] sessionPasswd, boolean canBeReadOnly)

通过这种方式注册的watcher将会作为整个zk会话期间的默认watcher，会一直被保存在客户端ZKWatchManager的defaultWatcher中，如果有其它的设置，则这个watcher会被覆盖

第二讲 zk进阶—watcher注册

- 其它注册api
  - getChildren(String path, Watcher watcher)
  - getChildren(String path, boolean watch)
    - Boolean watch表示是否使用上下文中默认的watcher，即创建zk实例时设置的watcher
  - getData(String path, boolean watch, Stat stat)
    - Boolean watch表示是否使用上下文中默认的watcher，即创建zk实例时设置的watcher
  - getData(String path, Watcher watcher, AsyncCallback.DataCallback cb, Object ctx)
  - exists(String path, boolean watch)
    - Boolean watch表示是否使用上下文中默认的watcher，即创建zk实例时设置的watcher
  - exists(String path, Watcher watcher)

第二讲 zk进阶—watcher注册

```
public static void main(String[] args){
    WatcherExample  wx = new WatcherExample();
    try {
        ZooKeeper   zk = new ZooKeeper(wx.getZkpath(),10000, wx);
        zk.getChildren("/zk-book2",false);
        Thread.sleep(300000);
    } catch (IOException e) {
```

不使用默认watcher

```
WatcherExample [Java Application] D:\Program Files\Java\jdk1.7.0_67\bin\javaw.exe (2015年11月8日 下午
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
path=null
eventType=None
```

虽然创建了新的子节点，但是还是没有收到监听事件，说明没有对应的watcher注册

```
WatcherExample [Java Application] D:\Program Files\Java\jdk1.7.0_67\bin\javaw.exe (2015年11月8日 下午2:1
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
path=null
eventType=None
```

运行订阅程序后，只有连接监听输出，没有节点事件输出

```
[zk: localhost:2181(CONNECTED) 7] create /zk-book2/five 456
Created /zk-book2/five
[zk: localhost:2181(CONNECTED) 8]
```

创建新的子节点

第二讲 zk进阶—watcher注册

Watcher设置后，一旦触发一次即会失效，如果需要一直监听，就需要再注册
---重要的事情说三遍

Watcher设置后，一旦触发一次即会失效，如果需要一直监听，就需要再注册

Watcher设置后，一旦触发一次即会失效，如果需要一直监听，就需要再注册

第二讲 zk进阶--watcher

- 连接成功时，连接的信息通过zk构造函数注册的watcher得到通知

```
public class WatcherRegister {
    private ZooKeeper  zk = null;

    public WatcherRegister(String connectString,Watcher watcher) {
        try {
            zk = new ZooKeeper(connectString,10000,watcher);
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    public void testWatcherdisabled(String path) throws KeeperException, InterruptedException{
        WatcherExample1 we1 = new WatcherExample1();
        zk.getData(path, we1, null);
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        WatcherExample we = new WatcherExample();
        WatcherRegister wr = new WatcherRegister("localhost:2181",we);
        try {
            wr.testWatcherdisabled("/zk-book2");
            Thread.sleep(300000);
        } catch (KeeperException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

构造函数注册的watcher

通过getData又注册了另外一个watcher

```
<terminated> WatcherRegister [Java Application] D:\Program Files\Java\jdk1.
log4j:WARN No appenders could be found for logger (org.apache.zo
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noco
watcher=com.watcher.WatcherExample
path=null
eventType=None
watcher=com.watcher.WatcherExample1
path=/zk-book2
eventType=NodeDataChanged
```

连接成功后输出

修改节点数据时输出

第二讲 zk进阶—watcher

```
public void testWatcherdisabled(String path) throws KeeperException, InterruptedException{
    WatcherExample1 we1 = new WatcherExample1();
    zk.getData(path, we1, null);
}
```

Problems | Javadoc | Declaration | Search | Console | Progress | Task List | Outline | Call Hierarchy

```
WatcherRegister [Java Application] D:\Program Files\Java\jdk1.7.0_67\bin\javaw.exe (2015年11月8日 下午3:36:46)
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
watcher=com.watcher.WatcherExample
path=null
eventType=None
```

启动后的控制台输出

1

```
[zk: localhost:2181(CONNECTED) 11] set /zk-book2 789
cZxid = 0x7
ctime = Sun Nov 01 23:36:00 CST 2015
mZxid = 0x43
mtime = Sun Nov 08 15:39:09 CST 2015
pZxid = 0x3c
cversion = 5
dataVersion = 4
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 3
numChildren = 5
[zk: localhost:2181(CONNECTED) 12]
```

2

第一次修改数据节点

```
WatcherRegister [Java Application] D:\Program Files\Java\jdk1.7.0_67\bin\javaw.exe (2015年11月8日 下午3:3
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
watcher=com.watcher.WatcherExample
path=null
eventType=None
watcher=com.watcher.WatcherExample1
path=/zk-book2
eventType=NodeDataChanged
```

收到通知

3

第二讲 zk进阶—watcher

第二讲 zk进阶—watcher

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    WatcherExample we = new WatcherExample();
    WatcherRegister wr = new WatcherRegister("localhost:2181",we);
    try {
        wr.testWatcherdisabled("/zk-book2");
        Thread.sleep(300000);
    } catch (KeeperException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

```
[zk: localhost:2181(CONNECTED) 40] set /zk-book2 999
cZxid = 0x7
ctime = Sun Nov 01 23:36:00 CST 2015
mZxid = 0x6e
mtime = Sun Nov 08 16:25:30 CST 2015
pZxid = 0x3c
cversion = 5
dataVersion = 33
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 3
numChildren = 5
[zk: localhost:2181(CONNECTED) 41]
```

第一次修改

```
@Override
public void process(WatchedEvent event) {
    // TODO Auto-generated method stub
    System.out.println("watcher="+this.getClass().getName());
    System.out.println("path="+event.getPath());
    System.out.println("eventType="+event.getType().name());
    try {
        //重新设置watcher
        WatcherExample1 we1 = new WatcherExample1();
        we1.setZk(zk);
        zk.getData(event.getPath(), we1, null);
    } catch (KeeperException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

```
WatcherRegister [Java Application] D:\Program Files\Java\jdk1.7.0_67\bin\javaw.exe (2015年11月8日 下午4:26:18)
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
watcher=com.watcher.WatcherExample
path=null
eventType=None
watcher=com.watcher.WatcherExample1
path=/zk-book2
eventType=NodeDataChanged
```

第一次收到通知

第二讲 zk进阶—watcher

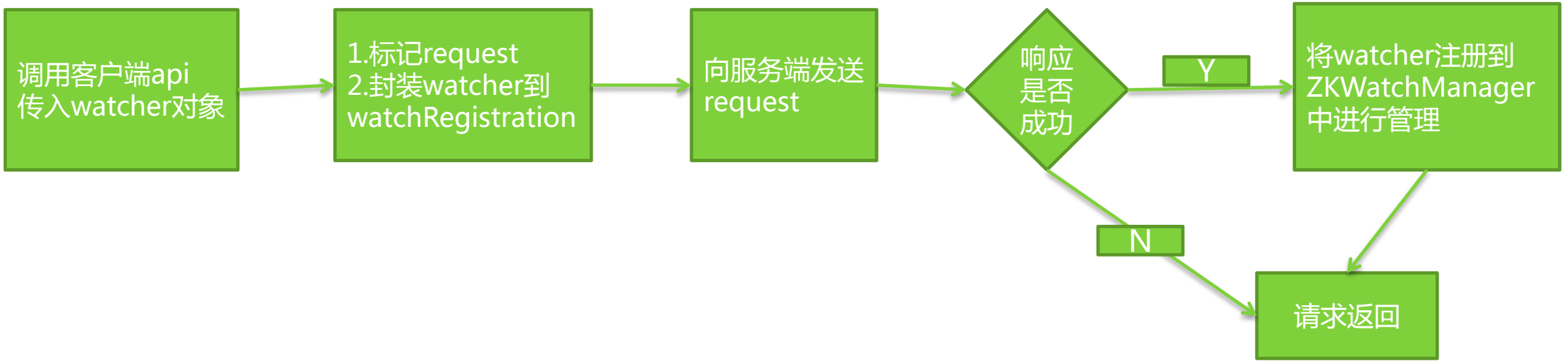
```
[zk: localhost:2181(CONNECTED) 42] set /zk-book2 99900
cZxid = 0x7
ctime = Sun Nov 01 23:36:00 CST 2015
mZxid = 0x72
mtime = Sun Nov 08 16:27:14 CST 2015
pZxid = 0x3c
cversion = 5
dataVersion = 35
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 5
numChildren = 5
[zk: localhost:2181(CONNECTED) 43]
```

第二次修改

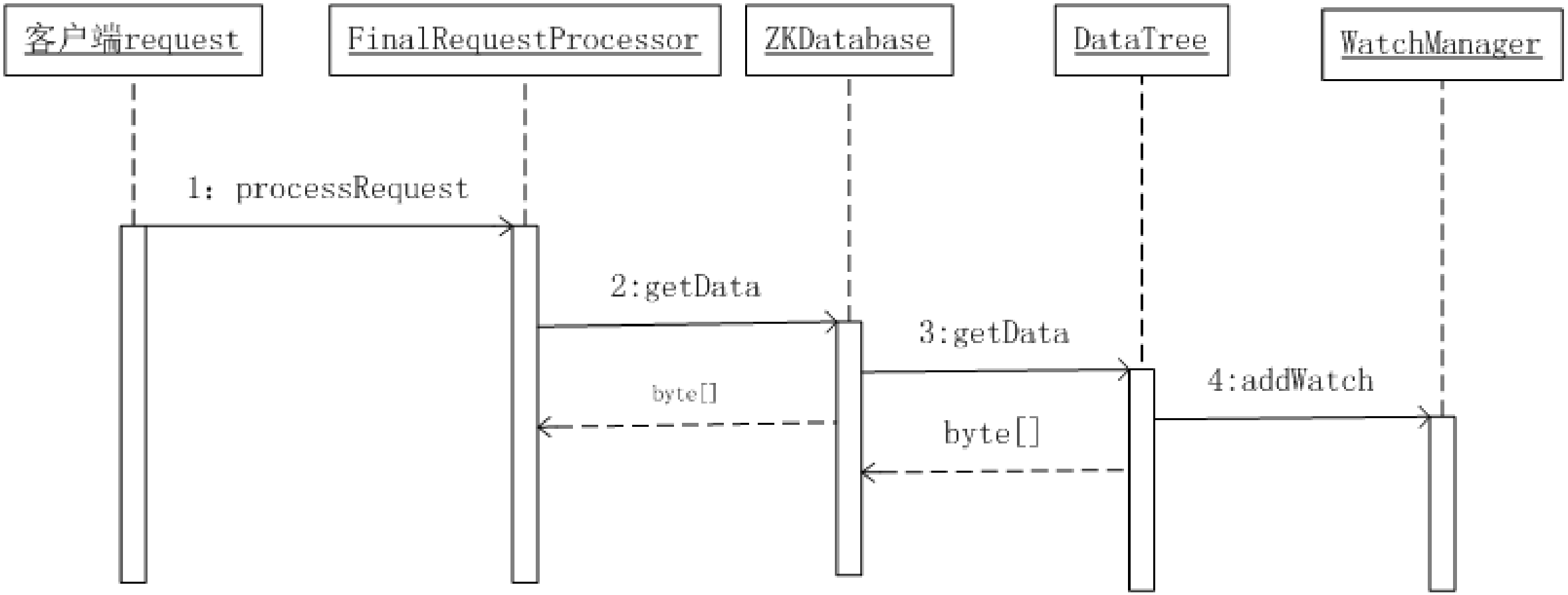
```
WatcherRegister [Java Application] D:\Program Files\Java\jdk1.7.0_67\bin\javaw.exe (2015年11月8日 下
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
watcher=com.watcher.WatcherExample
path=null
eventType=None
watcher=com.watcher.WatcherExample1
path=/zk-book2
eventType=NodeDataChanged
watcher=com.watcher.WatcherExample1
path=/zk-book2
eventType=NodeDataChanged
```

第二次收到的通知

第二讲 zk进阶—客户端watcher注册流程



第二讲 zk进阶—服务器端处理watcher



第二讲 zk进阶—服务器端处理watcher

- 判断调用是否需要注册watcher

```
case OpCode.getData: {
    lastOp = "GETD";
    GetDataRequest getDataRequest = new GetDataRequest();
    ByteBufferInputStream.byteBuffer2Record(request.request,
            getDataRequest);
    DataNode n = zks.getZKDatabase().getNode(getDataRequest.getPath());
    if (n == null) {
        throw new KeeperException.NoNodeException();
    }
    Long aclL;
    synchronized(n) {
        aclL = n.acl;
    }
    PrepRequestProcessor.checkACL(zks, zks.getZKDatabase().convertLong(aclL),
            ZooDefs.Perms.READ,
            request.authInfo);
    Stat stat = new Stat();
    byte b[] = zks.getZKDatabase().getData(getDataRequest.getPath(), stat,
            getDataRequest.getWatch() ? cnxn : null);
    rsp = new GetDataResponse(b, stat);
    break;
}
```

- 如果需要注册，则传入对象cnxn：

```
ServerCnxn cnxn = request.cnxn;
```

第二讲 zk进阶—服务器端处理watcher

- ServerCnxn类及cnxn对象
  - Zk客户端与服务器之间的tcp连接
  - 实现了watcher接口
  - 总结：即包含了连接信息又包含watcher信息

- watchManager
  - Zk服务器端Watcher的管理者
  - 从两个维度维护watcher
    - watchTable:从数据节点的粒度来维护
    - watch2Paths从watcher的粒度来维护
  - 负责watcher事件的触发

```
 */
public abstract class ServerCnxn implements Stats, Watcher {
    // This is just an arbitrary object to represent requests issued by
    // (aka owned by) this class
    final public static Object me = new Object();

    protected ArrayList<Id> authInfo = new ArrayList<Id>();

    /**
     * If the client is of old version, we don't send r-o mode info to it.
     * The reason is that if we would, old C client doesn't read it, which
     * results in TCP RST packet, i.e. "connection reset by peer".
     */
    boolean isOldClient = true;

    abstract int getSessionTimeout();

    abstract void close();

    public abstract void sendResponse(ReplyHeader h, Record r, String tag)
        throws IOException;

    /* notify the client the session is closing and close/cleanup socket */
    abstract void sendCloseSession();

    public abstract void process(WatchedEvent event);

    abstract long getSessionId();

    abstract void setSessionId(long sessionId);
```

第二讲 zk进阶—服务器端处理watcher

```
/**
 * This class manages watches. It allows watches to be associated with a string
 * and removes watchers and their watches in addition to managing triggers.
 */
public class WatchManager {
    private static final Logger LOG = LoggerFactory.getLogger(WatchManager.class);
    private final HashMap<String, HashSet<Watcher>> watchTable =
        new HashMap<String, HashSet<Watcher>>();
    private final HashMap<Watcher, HashSet<String>> watch2Paths =
        new HashMap<Watcher, HashSet<String>>();
    public synchronized int size(){}
    public synchronized void addWatch(String path, Watcher watcher) {}
    public synchronized void removeWatcher(Watcher watcher) {}
    public Set<Watcher> triggerWatch(String path, EventType type) {
    }
    public Set<Watcher> triggerWatch(String path, EventType type, Set<Watcher> supress) {}
    public synchronized void dumpWatches(PrintWriter pwriter, boolean byPath) {}
}
```

通过数据节点路径找到watcher

通过watcher找到对应的数据节点

第二讲 zk进阶—服务器端处理watcher

- Watcher触发
  - DataTree类
    - 维护节点目录树的数据结构

```
public Stat setData(String path, byte data[], int version, long zxid,
        long time) throws KeeperException.NoNodeException {
    Stat s = new Stat();
    DataNode n = nodes.get(path);
    if (n == null) {
        throw new KeeperException.NoNodeException();
    }
    byte lastdata[] = null;
    synchronized (n) {
        lastdata = n.data;
        n.data = data;
        n.stat.setMtime(time);
        n.stat.setMzxid(zxid);
        n.stat.setVersion(version);
        n.copyStat(s);
    }
    // now update if the path is in a quota subtree.
    String lastPrefix;
    if((lastPrefix = getMaxPrefixWithQuota(path)) != null) {
      this.updateBytes(lastPrefix, (data == null ? 0 : data.length)
          - (lastdata == null ? 0 : lastdata.length));
    }
    dataWatches.triggerWatch(path, EventType.NodeDataChanged);
    return s;
```

更新数据

触发事件

第二讲 zk进阶—客户端回调watcher

- 客户端回调watcher步骤
  - 反序列化,将字节流转换成WatcherEvent对象
  - 处理chrootPath
  - 还原watchedEvent:把WatcherEvent对象转换成WatchedEvent
  - 回调Watcher:把WatchedEvent对象交给EventThread线程

- EventThread
  - 从客户端的ZKWatchManager中取出Watcher,并放入waitingEvents队列中

第二讲 zk进阶—服务器端处理watcher

```
if (replyHdr.getXid() == -1) {
    // -1 means notification
    if (LOG.isDebugEnabled()) {
        LOG.debug("Got notification sessionid:0x"
            + Long.toHexString(sessionId));
    }
    WatcherEvent event = new WatcherEvent();
    event.deserialize(bbia, "response");

    // convert from a server path to a client path
    if (chrootPath != null) {
        String serverPath = event.getPath();
        if(serverPath.compareTo(chrootPath)==0)
            event.setPath("/");
        else if (serverPath.length() > chrootPath.length())
            event.setPath(serverPath.substring(chrootPath.length()));
        else {
            LOG.warn("Got server path " + event.getPath()
                    + " which is too short for chroot path "
                    + chrootPath);
        }
    }

    WatchedEvent we = new WatchedEvent(event);
    if (LOG.isDebugEnabled()) {
        LOG.debug("Got " + we + " for sessionid 0x"
                + Long.toHexString(sessionId));
    }

    eventThread.queueEvent( we );
    return;
}
```

反序列化

处理ChrootPath

封装得到watchedEvent

添加到时间处理线程

第二讲 zk进阶—acl

- Acl组成
  - Scheme:id:permission 比如：world:anyone:crdwa
  - Scheme：验证过程中使用的检验策略
  - Id：权限被赋予的对象，比如ip或者某个用户
  - Permission为权限，上面的crdwa，表示五个权限组合
  - 通过setAcl命令设置节点的权限
  - 节点的acl不具有继承关系
  - getAcl可以查看节点的Acl信息

```
[zk: localhost:2181(CONNECTED) 13] setAcl /zk-acl world:anyone:ca
cZxid = 0xa4
ctime = Mon Nov 16 23:52:46 CST 2015
mZxid = 0xa4
mtime = Mon Nov 16 23:52:46 CST 2015
pZxid = 0xa7
cversion = 1
dataVersion = 0
aclVersion = 3
ephemeralOwner = 0x0
dataLength = 7
numChildren = 1
[zk: localhost:2181(CONNECTED) 14]
[zk: localhost:2181(CONNECTED) 14] getAcl /zk-acl/test1
'world,'anyone
: cdrwa
[zk: localhost:2181(CONNECTED) 15]
```

设置acl

查询acl

第二讲 zk进阶—acl

- Scheme类型--world
  - Scheme:id:permission
  - Id为固定值：anyone，表示任何用户
  - world:anyone:crdwa表示任何用户都具有crdwa权限

```
[zk: localhost:2181(CONNECTED) 13] setAcl /zk-acl world:anyone:ca
cZxid = 0xa4
ctime = Mon Nov 16 23:52:46 CST 2015
mZxid = 0xa4
mtime = Mon Nov 16 23:52:46 CST 2015
```

没有读权限

```
[zk: localhost:2181(CONNECTED) 15] get /zk-acl
Authentication is not valid : /zk-acl
[zk: localhost:2181(CONNECTED) 16]
```

读取数据时提示没有权限

第二讲 zk进阶—acl

- Scheme类型---auth
  - Scheme:id:permission ，比如： auth:username:password:crdwa
  - 表示给认证通过的所有用户设置acl权限
  - 同时可以添加多个用户
  - 通过addauth命令进行认证用户的添加
    - addauth digest <username>:<password>
  - Auth策略的本质就是digest
  - 如果通过addauth创建多组用户和密码，当使用setAcl修改权限时，所有的用户和密码的权限都会跟着修改
  - 通过addauth新创建的用户和密码组需要重新调用setAcl才会加入到权限组中去

第二讲 zk进阶—acl

- Scheme类型---auth

```
[zk: localhost:2181(CONNECTED) 2] create /node2 node2data
Created /node2
[zk: localhost:2181(CONNECTED) 3] get /node2
node2data
cZxid = 0xd5
```

创建新节点

```
[zk: localhost:2181(CONNECTED) 4] getAcl /node2
'world,'anyone
: cdrwa
```

新节点的默认访问控制权限

```
[zk: localhost:2181(CONNECTED) 5] setAcl /node2 auth:node2u:111111:crdwa
Acl is not valid : /node2
[zk: localhost:2181(CONNECTED) 6]
```

如果没有提前通过addauth添加
则设置Acl失败

```
[zk: localhost:2181(CONNECTED) 6] addauth digest node2u:111111
[zk: localhost:2181(CONNECTED) 7]
[zk: localhost:2181(CONNECTED) 7]
[zk: localhost:2181(CONNECTED) 7] setAcl /node2 auth:node2u:111111:crdwa
cZxid = 0xd5
```

添加后，设置成功

```
[zk: localhost:2181(CONNECTED) 2] get /node2
Authentication is not valid : /node2
[zk: localhost:2181(CONNECTED) 3]
```

从另外一个客户端访问没有权限

```
[zk: localhost:2181(CONNECTED) 5] addauth digest node2u:111111
[zk: localhost:2181(CONNECTED) 6] get /node2
node2data
cZxid = 0xd5
ctime = Tue Nov 17 23:01:40 CST 2015
```

设置权限后可以访问到数据

第二讲 zk进阶—acl

- Scheme类型---digest
  - Scheme:id:permission ，比如： digest:username:password:crdwa
  - 指定某个用户及它的密码可以访问
  - 此处的username:password必须经过SHA-1和BASE64编码
    - BASE64(SHA1(username:password))
  - 通过addauth命令进行认证用户的添加
    - addauth digest <username>:<password>

第二讲 zk进阶—acl

- Scheme类型---digest

```
[zk: localhost:2181(CONNECTED) 45] create /acl/node5 node5data
Created /acl/node5
[zk: localhost:2181(CONNECTED) 46] setAcl /acl/node5 digest:node5:xxIpKT4mXtOX6A78PvNT6k40/xg=:crdwa
cZxid = 0x100
```

```
[zk: localhost:2181(CONNECTED) 40] create /acl/node4 node4data
Created /acl/node4
[zk: localhost:2181(CONNECTED) 41] getAcl /acl/node4
'world,'anyone
: cdrwa
[zk: localhost:2181(CONNECTED) 42] setAcl /acl/node4 digest:aclnode4:111111:crdwa
cZxid = 0xfe
ctime = Thu Nov 19 21:06:11 CST 2015
mZxid = 0xfe
mtime = Thu Nov 19 21:06:11 CST 2015
pZxid = 0xfe
cversion = 0
dataVersion = 0
aclVersion = 1
ephemeralOwner = 0x0
dataLength = 9
numChildren = 0
[zk: localhost:2181(CONNECTED) 43] addauth digest aclnode4:111111
[zk: localhost:2181(CONNECTED) 44]
[zk: localhost:2181(CONNECTED) 44] get /acl/node4
Authentication is not valid : /acl/node4
[zk: localhost:2181(CONNECTED) 45]
```

已经设置了用户和密码，为什么还是没有权限呢？

```
[zk: localhost:2181(CONNECTED) 49] addauth digest node5:111111
[zk: localhost:2181(CONNECTED) 50]
[zk: localhost:2181(CONNECTED) 50] get /acl/node5
node5data
```

权限认证成功

```
public class DigestTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            System.out.print(DigestAuthenticationProvider.generateDigest("node5:111111"));
        } catch (NoSuchAlgorithmException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

```
<terminated> DigestTest [Java Application]
node5:xxIpKT4mXtOX6A78PvNT6k40/xg=
```

第二讲 zk进阶—acl

- Scheme类型---IP
  - Scheme:id:permission ，比如： ip:127.0.0.1:crdwa
  - 指定某个ip地址可以访问

```
[zk: localhost:2181(CONNECTED) 24] create /acl/node6 node6data
Created /acl/node6
[zk: localhost:2181(CONNECTED) 25]
[zk: localhost:2181(CONNECTED) 25] setAcl /acl/node6 ip:127.0.0.1:crdwa
cZxid = 0x104
ctime = Thu Nov 19 21:19:36 CST 2015
```

```
[zk: localhost:2181(CONNECTED) 26] get /acl/node6
node6data
cZxid = 0x104
ctime = Thu Nov 19 21:19:36 CST 2015
mZxid = 0x104
```

设置了127.0.0.1可以访问

```
[zk: localhost:2181(CONNECTED) 27] setAcl /acl/node6 ip:192.168.1.1:crdwa
cZxid = 0x104
ctime = Thu Nov 19 21:19:36 CST 2015
mZxid = 0x104
```

重新设置了别的ip地址

```
[zk: localhost:2181(CONNECTED) 28] get /acl/node6
Authentication is not valid : /acl/node6
[zk: localhost:2181(CONNECTED) 29]
```

新的ip地址访问失败

第二讲 zk进阶—acl

- Scheme类型---super
  - 供运维人员维护节点使用
  - 有权限操作任何节点
  - 启动时，在命令参数中配置
    - -Dzookeeper.DigestAuthenticationProvider.superDigest=admin:015uTByzA4zSglcmseJsxTo7n3c=
    - 打开zkCli.cmd，在java命令后面增加以上配置
  - 用户名和密码也需要通过sha1和base64编码

第二讲 zk进阶—acl

- Scheme类型---super

```
"-Dzookeeper.root.logger=%ZOO_LOG4J_PROP%" "-Dzookeeper.DigestAuthenticationProvider.superDigest=admin:015uTByzA4zSqlcmseJsxTo7n3c="
```

```
[zk: localhost:2181(CONNECTED) 0] addauth digest admin:111111
[zk: localhost:2181(CONNECTED) 1]
[zk: localhost:2181(CONNECTED) 1] ls /
[zk-book2, nodeauth, acl, newznode, watchtest, zookeeper, node
[zk: localhost:2181(CONNECTED) 2] getAcl /node2
'digest,'node2u:+6pRQY+9OA26aywjYeRpE0LjwyY=
: cdrwa
```

客户端通过super级别的用户admin授权
即可访问任何节点

```
[zk: localhost:2181(CONNECTED) 5] get /acl/node5
node5data
cZxid = 0x100
ctime = Thu Nov 19 21:11:03 CST 2015
```

Thanks

FAQ时间