Zookeeper常见命令操作和API操作

**【实验名称】**

Zookeeper 常见命令操作和API操作

**【实验目的】**

通过对Zookeeper的操作，了解Zookeeper 一些常见操作命令以及client调用api

**【实验要求】**

用Java代码实现Zookeeper的api调用，实现Zookeeper数据的写入、读取和删除等操作。

**【实验环境】**

安装成功Zookeeper 的Centos6.8操作系统的虚拟机

**【实验步骤】**

## 1.基础环境准备

### 1.1 IP地址规划

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 主机 | 角色 | IP | 主机名 | 账户 | 密码 |
| 1 | hadoop1 | Master  NameNode  JobTracker | 172.16.16.131 | hadoop1 | root | 123456 |
| 2 | hadoop2 | Slave  DataNode  TaskTracker | 172.16.16.132 | hadoop2 | root | 123456 |
| 3 | hadoop3 | Slave  DataNode  TaskTracker | 172.16.16.133 | hadoop3 | root | 123456 |

## 2.Zookeeper常见命令

### 2.1 zookeeper常见命令

#### 2.1.1 zkCli.sh连接

打开zookeeper客户端、可以用help命令来查看客户端的操作zkCli.sh 若连接不同的主机，可使用如下命令：

[root@hadoop1]# zkCli.sh -server hadoop1:2181

#### 2.1.2 创建Zookeeper节点

create命令，创建一个Zookeeper节点 **create [-s] [-e] path data acl**

[zk: hadoop1:2181(CONNECTED) 1]# create -e /zk-temp 123  **临时节点**

[zk: hadoop1:2181(CONNECTED) 2]# create -s /zk-test 123 顺序节点

[zk: hadoop1:2181(CONNECTED) 3]# create -e /zk-permanent 123  **永久节点**

创建成功如图：



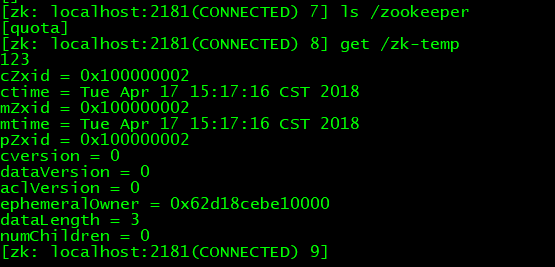
#### 2.1.3 查看节点

查看当前节点下的所有子节点 ls /**zk-temp 查看根节点下所有子节点ls/**

获取当前节点下内容和属性 get / **zk-temp**

[zk: hadoop1:2181(CONNECTED) 7]# ls /zookeeper

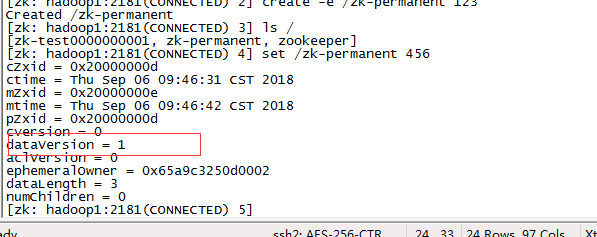
[zk: hadoop1:2181(CONNECTED) 8]# get /zk-temp



#### 2.1.4 修改节点

更新节点内容 **set /zk-permanent 456** dataVersion会变化，命令如下：

[zk: hadoop1:2181(CONNECTED) 7]# set /zk-permanent 456



#### 2.1.5 删除节点

**删除节点如果当前节点有子节点需要先删除子节点，命令如下：**

[zk: hadoop1:2181(CONNECTED) 7]# delete /zk-permanent



#### 2.1.6 退出客户端

[zk: hadoop1:2181(CONNECTED) 7]# quit

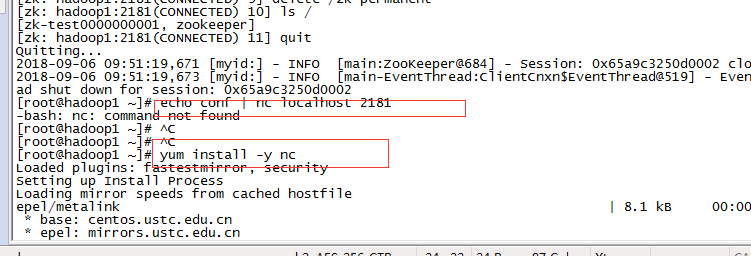
### 2.2 zookeeper四字命令

Zookeeper四字命令 echo \* | nc localhost 2181 (\*指下面命令)

[root@hadoop1]# echo conf | nc localhost 2181

如果nc未安装则需要通过yum源安装命令如下：

[root@hadoop1]# yum install -y nc



conf：服务配置的详细信息，如端口、zk数据及日志配置路径、最大连接数

cons:列出所有连接到这台服务器的客户端连接/会话的详细信息

crst:重置当前这台服务器所有连接/会话的统计信息

dump:列出未经处理的会话和临时节点（只在leader上有效）

envi:输出关于服务器的环境详细信息（不同于conf命令），比如host.name、java.version、java.home

ruok:测试服务是否处于正确运行状态。如果正常返回"imok"，否则返回空

srst:重置服务器的统计信息

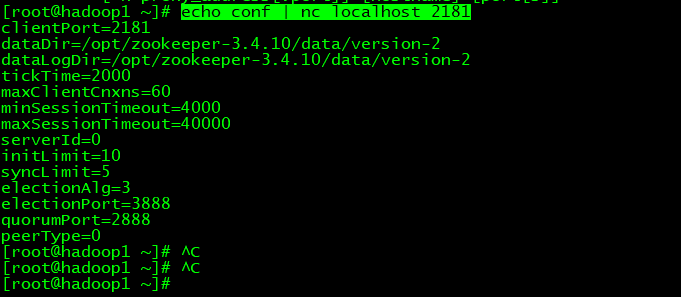
stat:输出服务器的详细信息：接收/发送包数量、连接数、模式（leader/follower）、节点总数、延迟。 所有客户端的列表

wchs:(New in 3.3.0)列出服务器watches的简洁信息：连接总数、watching节点总数和watches总数

wchc:(New in 3.3.0)通过session分组，列出watch的所有节点，它的输出是一个与 watch 相关的会话的节点列表。如果watches数量很大的话，将会产生很大的开销，会影响性能，小心使用

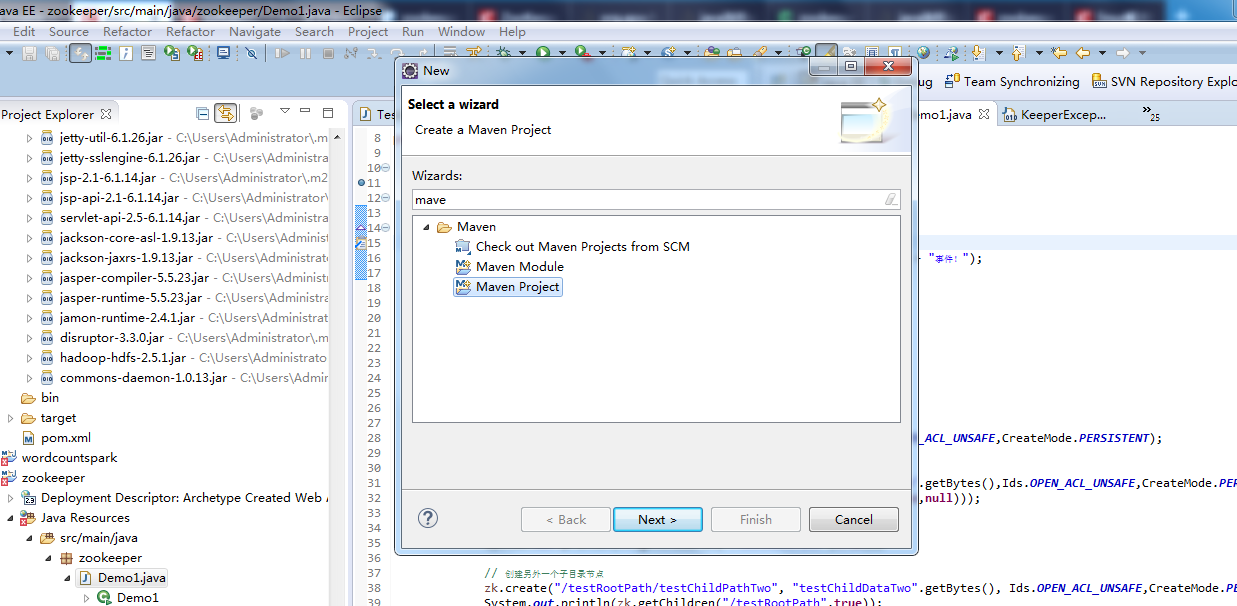
wchp:(New in 3.3.0)通过路径分组，列出所有的 watch 的session id信息。它输出一个与 session相关的路径。如果watches数量很大的话，将会产生很大的开销，会影响性能，小心使用

mntr:(New in 3.4.0)列出集群的健康状态。包括“接受/发送”的包数量、操作延迟、当前服务模式（leader/follower）、节点总数、watch总数、临时节点总数。

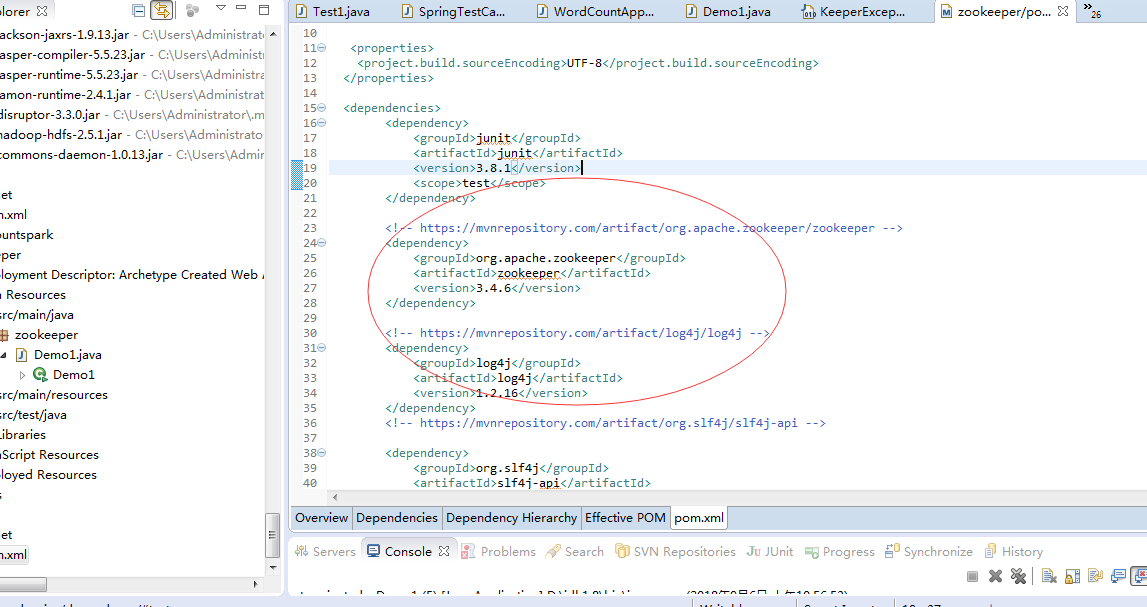


## 3.zookeeper java客户端操作

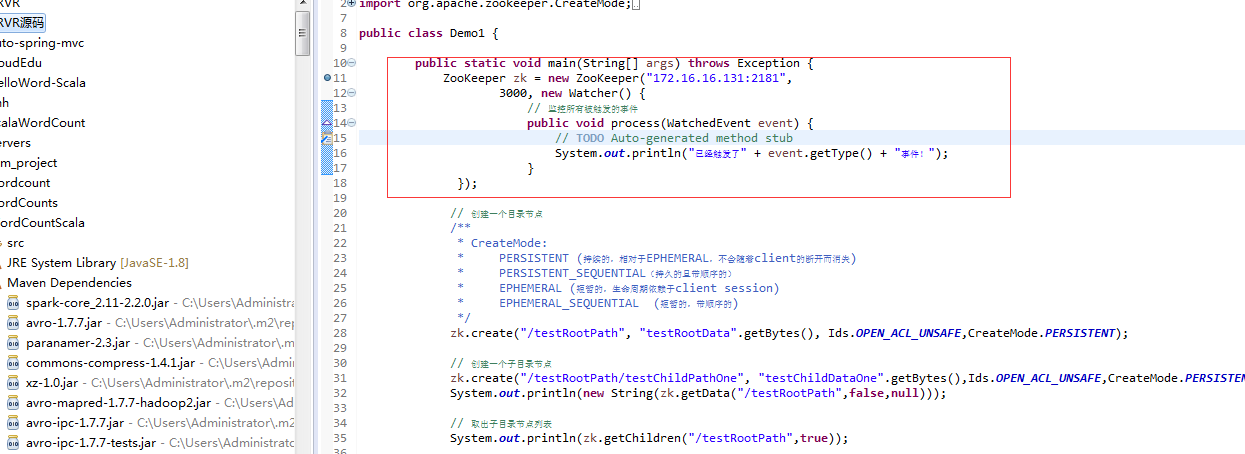
### 3.1创建java maven 项目



### 3.2配置pom.xml文件，加载jar包



### 3.3创建测试项目建立zookeeper连接



### 3.4对zookeeper进行创建、删除等操作



### 3.5 java代码

package zookeeper;

import org.apache.zookeeper.CreateMode;

import org.apache.zookeeper.WatchedEvent;

import org.apache.zookeeper.Watcher;

import org.apache.zookeeper.ZooDefs.Ids;

import org.apache.zookeeper.ZooKeeper;

public class Demo1 {

public static void main(String[] args) throws Exception {

ZooKeeper zk = new ZooKeeper("172.16.16.131:2181",

3000, new Watcher() {

// 监控所有被触发的事件

public void process(WatchedEvent event) {

// TODO Auto-generated method stub

System.out.println("已经触发了" + event.getType() + "事件！");

}

});

// 创建一个目录节点

/\*\*

\* CreateMode:

\* PERSISTENT (持续的，相对于EPHEMERAL，不会随着client的断开而消失)

\* PERSISTENT\_SEQUENTIAL（持久的且带顺序的）

\* EPHEMERAL (短暂的，生命周期依赖于client session)

\* EPHEMERAL\_SEQUENTIAL (短暂的，带顺序的)

\*/

zk.create("/testRootPath", "testRootData".getBytes(), Ids.OPEN\_ACL\_UNSAFE,CreateMode.PERSISTENT);

// 创建一个子目录节点

zk.create("/testRootPath/testChildPathOne", "testChildDataOne".getBytes(),Ids.OPEN\_ACL\_UNSAFE,CreateMode.PERSISTENT);

System.out.println(new String(zk.getData("/testRootPath",false,null)));

// 取出子目录节点列表

System.out.println(zk.getChildren("/testRootPath",true));

// 创建另外一个子目录节点

zk.create("/testRootPath/testChildPathTwo", "testChildDataTwo".getBytes(), Ids.OPEN\_ACL\_UNSAFE,CreateMode.PERSISTENT);

System.out.println(zk.getChildren("/testRootPath",true));

// 修改子目录节点数据

zk.setData("/testRootPath/testChildPathOne","hahahahaha".getBytes(),-1);

byte[] datas = zk.getData("/testRootPath/testChildPathOne", true, null);

String str = new String(datas,"utf-8");

System.out.println(str);

//删除整个子目录 -1代表version版本号，-1是删除所有版本

zk.delete("/testRootPath/testChildPathOne", -1);

System.out.println(zk.getChildren("/testRootPath",true));

System.out.println(str);

/\*byte[] datas = zk.getData("/testRootPath/testChildPathTwo", true, null);

String str = new String(datas,"utf-8");

System.out.println(str); \*/

}

}

### 3.6 pom.xml配置文件

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.zookeeper/zookeeper -->

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

<version>3.4.6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/log4j/log4j -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.16</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

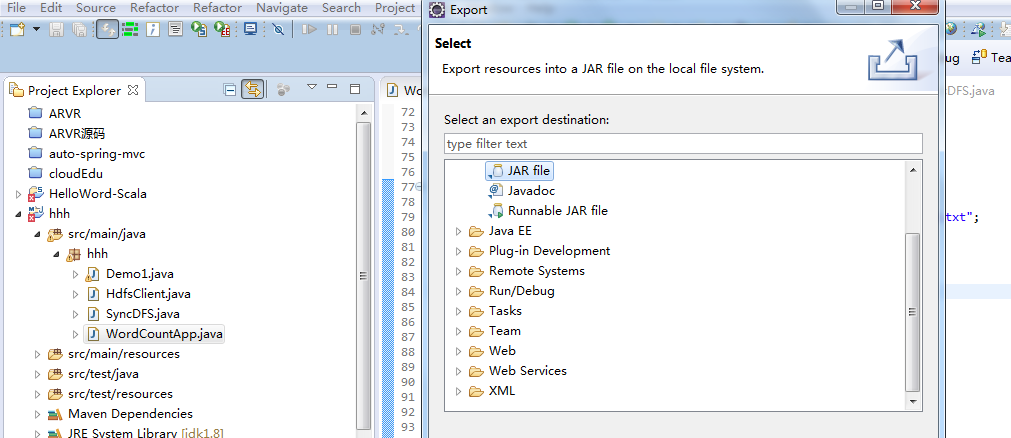
<version>1.6.1</version>

</dependency>

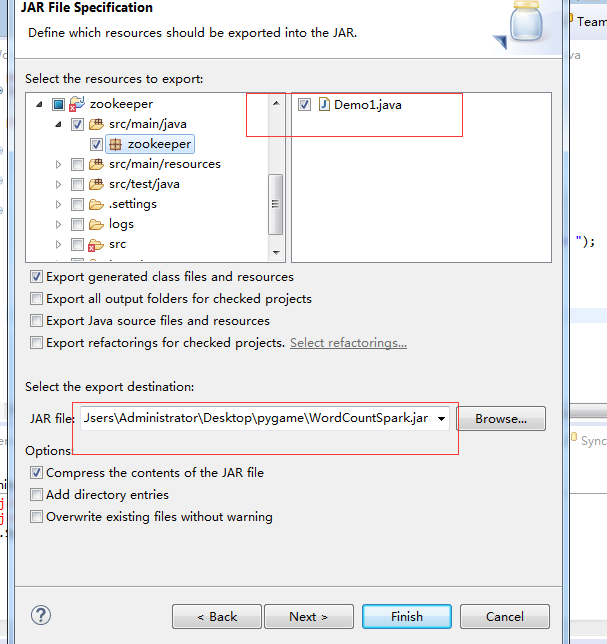
</dependencies>

### 3.7 把java打包成jar

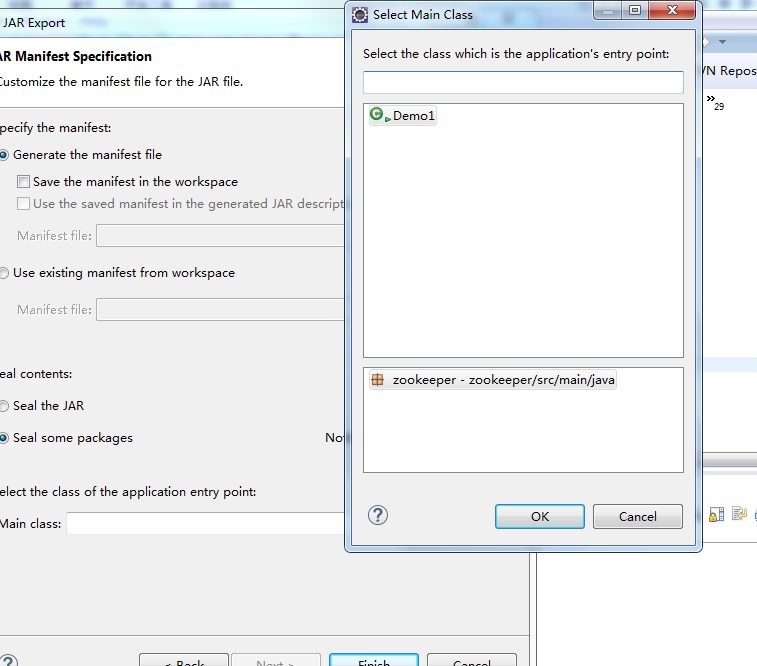
#### 3.7.1选中项目右键，选择Export,选中JAR file,然后next下一步



#### 3.7.2选择要打包的项目和地址,然后next下一步



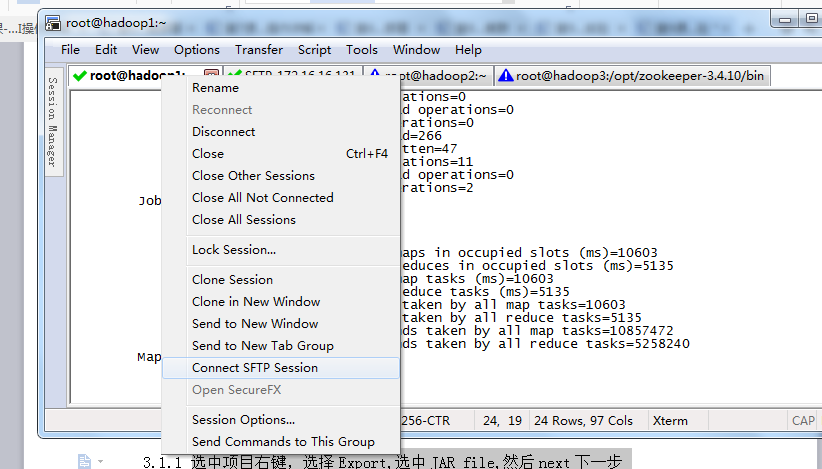
#### 3.7.3选择打包的class,选择OK 然后Finish



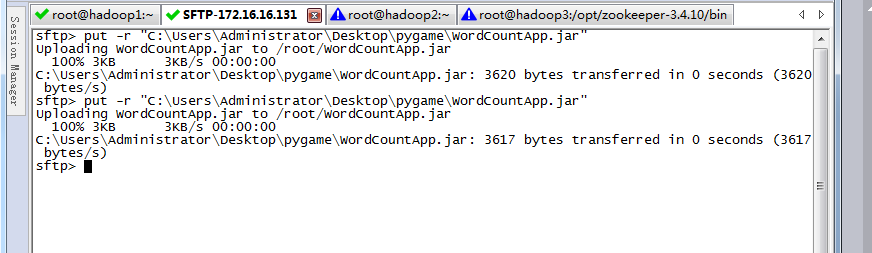
## 4.Hadoop运行打包好的jar文件

### 4.1 将打包好的jar上传到linux

#### 4.1.1 右键linux虚拟机，选择Connect SFTP Session 进入sftp

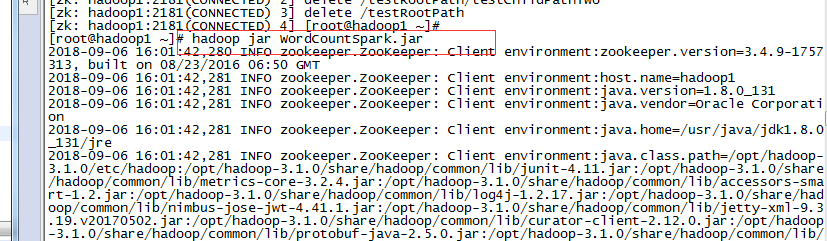


#### 4.1.2将jar直接拖到sftp上

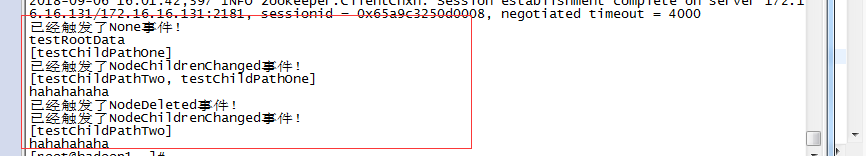


#### 4.1.3 然后执行下面命令

[root@hadoop1]# hadoop jar WordCountSpark.jar



#### 4.1.4 执行成功如图：



至此，本节实验结束！