1. ZooKeeper Command

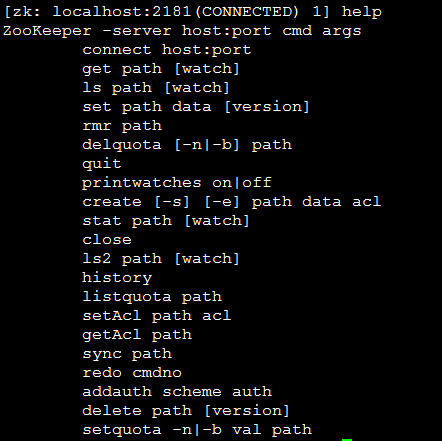
连接ZooKeeper执行命令：

/apply/zookeeper/bin/zkCli.sh -server 192.168.200.125:2181

/apply/zookeeper ==> ZooKeeper的安装目录

192.168.200.125 ==> ZooKeeper Server IP

执行上面命令成功后，就可以管理ZooKeeper数据了。输入 help 命令会得到帮助信息，如图：



常用命令：

* get path - 获取指定path的数据
* ls path - 列出指定path下的znode
* set path data - 设置指定path的值
* create [-s] [-e] path data acl - 创建znode
* delete path - 删除znode

1. ZooKeeper Client
2. 依赖jar包

<dependency>

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

<artifactId>zookeeper</artifactId>

<version>3.4.8</version>

</dependency>

1. 配置

<!-- =============ZooKeeper====================== -->

<!-- zookeeper工厂 -->

<bean id=*"zooKeeperFactory"* class=*"com.banksteel.bonesvn.util.zookeeper.ZooKeeperFactory"*

p:connectString=*"192.168.200.125:2181,192.168.200.126:2181,192.168.200.104:2181****/bone/svn****"*

p:sessionTimeout=*"6000"* />

<!-- 节点名制造器 -->

<bean id=*"zkLatentIpNodeNameMaker"* class=*"com.banksteel.bonesvn.util.zookeeper.ZkLatentIpNodeNameMaker"*

p:zk-ref=*"zooKeeperFactory"* p:defaultIpDataNode=*"DEV"* p:child=*"**app\_param.properties"* />

<!-- zookeeper数据反序列化 -->

<bean id=*"zkPropertiesDeserializer"* class=*"com.banksteel.bonesvn.util.zookeeper.ZkPropertiesDeserializer"* />

<!-- 节点 -->

<bean id=*"cfgNode"* class=*"com.banksteel.bonesvn.util.zookeeper.ZkNode"*

p:debug=*"true"* p:zk-ref=*"zooKeeperFactory"* p:maker-ref=*"zkLatentIpNodeNameMaker"*

p:deserializer-ref=*"zkPropertiesDeserializer"* />

<!-- 获取节点数据 -->

<bean id=*"cfgProperties"* class=*"org.springframework.beans.factory.config.MethodInvokingFactoryBean"*

p:targetObject-ref=*"cfgNode"* p:targetMethod=*"load"* />

此配置需注意两点：

***/bone/svn***指定 为此项目使用ZooKeeper的根路径，项目中使用的path都基于此根路径。

*app\_param.properties* 为项目的配置文件名

**上面的配置为定义ZooKeeper，下面说下怎样把ZooKeeper的配置集成如Spring中：**

<bean id=*"propertyConfigurer"* class=*"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"*>

<property name=*"ignoreUnresolvablePlaceholders"* value=*"true"*/>

<property name=*"properties"* ref=*"cfgProperties"* />

<property name=*"locations"*>

<list>

<value>classpath:rabbitmq.properties</value>

</list>

</property>

</bean>

请注意上述标绿的配置，此配置让ZooKeeper的配置项托管于Spring中。

1. 写配置文件到ZooKeeper

ZooKeeperTest 类中写了一个使用案例，main函数逻辑即为把项目中的配置文件上传到ZooKeeper中：

String connectString = "192.168.200.125:2181,192.168.200.126:2181,192.168.200.104:2181/bone/bone-app";

**final** CountDownLatch count = **new** CountDownLatch(1);

ZooKeeper zk = **null**;

**try** {

Watcher newWatcher = **new** Watcher() {

@Override

**public** **void** process(WatchedEvent event) {

System.***out***.println(event);

**if** (event.getType() == Event.EventType.***None***

&& event.getState() == KeeperState.***SyncConnected***) {

count.countDown();

}

}

};

zk = **new** ZooKeeper(connectString, 6000, newWatcher);

count.await();

ZooKeeperTest test = **new** ZooKeeperTest(zk);

test.addCfgFile("", ProjectEnv.***DEV***, **new** String[] {

"app\_param.properties"

});

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (zk != **null**)

**try** {

zk.close();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

注：上面三块标绿是需要注意的：第一个为连接地址；第二个为需要添加的配置的环境；第三个为配置文件名，需要本地存在此文件。

环境变量枚举如下：

**static** **enum** ProjectEnv {

***DEV***("DEV", "开发环境") {

@Override

**public** String[] getIps() {

/\*

//如下IP在项目启动时会自动获取ZooKeeper配置项

return new String[] {

"192.168.1.101",

"192.168.1.102",

"192.168.1.103",

};

\*/

**return** **new** String[] {

};

}

},

***TEST***("TEST", "测试环境") {

@Override

**public** String[] getIps() {

**return** **new** String[] {

"192.168.200.104"

};

}

},

***PRE***("PRE", "预发环境") {

@Override

**public** String[] getIps() {

**return** **new** String[] {

};

}

},

***PROD***("PROD", "生产环境") {

@Override

**public** String[] getIps() {

**return** **new** String[] {

};

}

};

**private** **final** String value;

**private** **final** String remark;

**private** ProjectEnv(String value, String remark) {

**this**.value = value;

**this**.remark = remark;

}

**public** **abstract** String[] getIps();

**public** String getValue() {

**return** **this**.value;

}

**public** String getRemark() {

**return** **this**.remark;

}

}

注：getIps() 方法为当前环境部署的服务器IP地址。