Zookeeper 分布式锁

参考网址：http://blog.csdn.net/desilting/article/details/41280869

编写者：何永安

时间：2016.07.15

目　录

1：核心类 DistributedLock

```
package com.iqcloud.common.distributedlock;

import org.apache.zookeeper.*;
import org.apache.zookeeper.data.Stat;

import java.util.List;
import java.io.IOException;
import java.util.Collections;

public class DistributedLock implements Watcher{
	private ZooKeeper zk = null;

	/*
	 * 锁的根目录
	 */
	private final String LOCK_ROOT_PATH = "/LOCKS_ROOT";

	/*
	 * zookeeper 连接地址
	 */
	private String zookeeperConnection;

	private String selfPath;
	private String waitPath;
	private String parentPath;
	private String subPath;
	private boolean isDistributed;
	private boolean canRun = false;
	private Action action = null;
	private Boolean isHasFinish = false;                    // 任务是否已经执行完成

	public DistributedLock(Action theAction, boolean theDistributed, String theZookeeperConnection, String parentNode, String theSubNode) {
		action = theAction;
		isDistributed = theDistributed;
		zookeeperConnection = theZookeeperConnection;
		try {
			zk = new ZooKeeper(zookeeperConnection, 6000, this);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		parentPath = LOCK_ROOT_PATH + "/" + parentNode;
		subPath = parentPath + "/" + theSubNode;
	}

	/**
	 * 获取锁
	 * @return
	 */
	public boolean getLock() {
		// 创建相关节点
		try {
			if(null == zk.exists(LOCK_ROOT_PATH, false)){
				zk.create(LOCK_ROOT_PATH, null, ZooDefs.Ids.OPEN_ACL_UNSAFE, CreateMode.PERSISTENT);
			}

			if(null == zk.exists(parentPath, false)){
			zk.create(parentPath, null, ZooDefs.Ids.OPEN_ACL_UNSAFE, CreateMode.PERSISTENT);
			}

			if(null == zk.exists(subPath, false)){
			selfPath = zk.create(subPath, null, ZooDefs.Ids.OPEN_ACL_UNSAFE, CreateMode.EPHEMERAL_SEQUENTIAL);
			}

			System.out.println("创建锁路径:"+selfPath);

			if (isDistributed){// 同步执行
				boolean bok = checkMinPath();
				if (bok){
					canRun = true;
				}
				while (true) {
					if (canRun){
						boolean bok1 = checkMinPath();
					if (bok1){// 获取所成功
```

```
                    getLockSuccess();
                    return bok;
                   }
                   break;
            }else{
                    System.out.println("同步等待...");
                    Thread.sleep(1000);
            }
        }
        return true;
}else{// 异步执行
        // 开个线程跑
        Thread theThread = new Thread(){
                @Override
                public void run() {
                        // TODO Auto-generated method stub
                        boolean bok;
                        try {
                                bok = checkMinPath();
                                if (bok){// 获取所成功
                                        synchronized (isHasFinish) {
                                                if (!isHasFinish){
                                                        isHasFinish = true;
                                                        getLockSuccess();
                                                }
                                        }
                                } catch (KeeperException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                } catch (InterruptedException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
                };

                 theThread.start();

                return true;
            }
        } catch (KeeperException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
                return false;
        } catch (InterruptedException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
                return false;
        }
}

/**
 * 获取锁成功
 */
public void getLockSuccess() throws KeeperException, InterruptedException {
        if(zk.exists(this.selfPath,false) == null){
                System.out.println("本节点已不在了...");
                releaseConnection();
                return;
        }

        System.out.println("获取锁成功，赶紧干活！");
        try {
                if (null != action){
                        action.action();
                }
        } finally {
                System.out.println("删除本节点："+selfPath);
                zk.delete(this.selfPath, -1);
                releaseConnection();
        }
}

/**
 * 关闭 ZK 连接
 */
public void releaseConnection() {
        if (this.zk != null) {
                try {
                        this.zk.close();
```

```
            } catch ( InterruptedException e ) {}
        }
        System.out.println("释放连接");
}

/**
 * 检查自己是不是最小的节点
 * @return
 */
public boolean checkMinPath() throws KeeperException, InterruptedException {
        System.out.println("checkMinPath");
        List<String> subNodes = zk.getChildren(parentPath, false);
        Collections.sort(subNodes);
        int index = subNodes.indexOf(selfPath.substring(parentPath.length()+1));
        switch (index){
                case -1:{
                        System.out.println("本节点已不在了..." + selfPath);
                        return false;
                }
                case 0:{
                        System.out.println("子节点中，我果然是老大" + selfPath);
                        return true;
                }
                default:{
                        this.waitPath = parentPath + "/" + subNodes.get(index - 1);
                        System.out.println("获取子节点中，排在我前面的" + waitPath);
                        try{
                                zk.getData(waitPath, true, new Stat());
                                return false;
                        }catch(KeeperException e){
                                if(zk.exists(waitPath,false) == null){
                                        System.out.println("子节点中，排在我前面的" + waitPath + "已失踪，幸福来得太突然?");
                                        return checkMinPath();
                                }else{
                                        throw e;
                                }
                        }
                }
        }
}

@Override
public void process(WatchedEvent event) {
        if(event == null){
                return;
        }

        Event.KeeperState keeperState = event.getState();
        Event.EventType eventType = event.getType();
        if (Event.KeeperState.SyncConnected == keeperState) {
                if (Event.EventType.None == eventType ) {
                        System.out.println("成功连接上 ZK 服务器" );
                }else if (event.getType() == Event.EventType.NodeDeleted && event.getPath().equals(waitPath)) {
                        System.out.println("收到情报，排我前面的家伙已挂，我是不是可以出山了？ ");
                        if (isDistributed){//  同步执行
                                canRun = true;
                        }else{
                                try {
                                        if(checkMinPath()){
                                                synchronized (isHasFinish) {
                                                        if (!isHasFinish){
                                                                isHasFinish = true;
                                                                getLockSuccess();
                                                        }
                                                }
                                        }
                                } catch (KeeperException e) {
                                        e.printStackTrace();
                                } catch (InterruptedException e) {
                                        e.printStackTrace();
                                }
                        }
                }
        } else if ( Event.KeeperState.Disconnected == keeperState ) {
                System.out.println("与 ZK 服务器断开连接" );
        } else if ( Event.KeeperState.AuthFailed == keeperState ) {
                System.out.println("权限检查失败" );
        } else if ( Event.KeeperState.Expired == keeperState ) {
                System.out.println("会话失效" );
        }
}
```

```
}
```

2：接口 action

```
package com.iqcloud.distributedlock;

public interface Action {
    public void action();
}
```

3：Demo：DistributedLockController

```
package com.kdmc.ruida.controller;

import java.io.IOException;

import javax.servlet.http.HttpSession;

import org.apache.tools.ant.taskdefs.Sleep;
import org.apache.xmlbeans.impl.xb.xsdschema.WhiteSpaceDocument.WhiteSpace.Value;
import org.apache.zookeeper.KeeperException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import com.alibaba.fastjson.JSON;
import com.iqcloud.auth.dto.IQCloudInfoDto;
import com.iqcloud.auth.facade.IIqUsersFacade;
import com.iqcloud.distributedlock.Action;
import com.iqcloud.distributedlock.DistributedLock;

/*
 * 用户信息控制器
 */

@Controller
@RequestMapping(value = "/distributedLockController")
public class DistributedLockController {
	@Autowired
	private IIqUsersFacade iIqUsersFacade;

	private Integer value = 0;

	/*
	 * 登录
	 */
	@RequestMapping(value = "/login")
	public @ResponseBody IQCloudInfoDto login(@RequestBody IQCloudInfoDto iqCloudInfoDto, HttpSession session){
		IQCloudInfoDto iqCloudInfoDto2 = new IQCloudInfoDto();
		iqCloudInfoDto2.setJsonBody("OK");
		final String jsonStr = JSON.toJSONString(iqCloudInfoDto);
		/*
		for (int i = 0; i < 10; i++) {
			try {
				System.out.println(jsonStr + "---> " + i);
				Thread.sleep(1000);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}



		synchronized (value) {
			for (int i = 0; i < 10; i++) {
				try {
					System.out.println(jsonStr + "---> " + i);
					Thread.sleep(1000);
				} catch (InterruptedException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
		}
```

```
        */

        Action action = new Action() {
            @Override
            public void action() {
                // TODO Auto-generated method stub
                for (int i = 0; i < 25; i++) {
                    try {
                        System.out.println(jsonStr + "---> " + i);
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
        };

        DistributedLock distributedLock = new DistributedLock(action, true, "127.0.0.1:2181", "TEST_ACTION_A", "TEST");
        distributedLock.getLock();

        return iqCloudInfoDto2;
    }

    /*
     * 登录
     */
    @RequestMapping(value = "/login1")
    public @ResponseBody IQCloudInfoDto login1(@RequestBody IQCloudInfoDto iqCloudInfoDto, HttpSession session){
        IQCloudInfoDto iqCloudInfoDto2 = new IQCloudInfoDto();
        iqCloudInfoDto2.setJsonBody("OK");

        final String jsonStr = JSON.toJSONString(iqCloudInfoDto);
        /*
        for (int i = 0; i < 10; i++) {
            try {
                System.out.println(jsonStr + "---> " + i);
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }



        synchronized (value) {
            for (int i = 0; i < 10; i++) {
                try {
                    System.out.println(jsonStr + "---> " + i);
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
        */

        Action action = new Action() {
            @Override
            public void action() {
                // TODO Auto-generated method stub
                for (int i = 0; i < 25; i++) {
                    try {
                        System.out.println(jsonStr + "---> " + i);
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
        };

        DistributedLock distributedLock = new DistributedLock(action, true, "127.0.0.1:2181", "TEST_ACTION_B", "TEST");
        distributedLock.getLock();

        return iqCloudInfoDto2;
    }
}
```