# **ZkClient**

       ZkClient是由Datameer的工程师开发的开源客户端，对Zookeeper的原生API进行了包装，实现了超时重连、Watcher反复注册等功能。

在使用ZooKeeper的Java客户端时，经常需要处理几个问题：重复注册watcher、session失效重连、异常处理。

    目前已经运用到了很多项目中，知名的有Dubbo、Kafka、Helix。

 Maven依赖

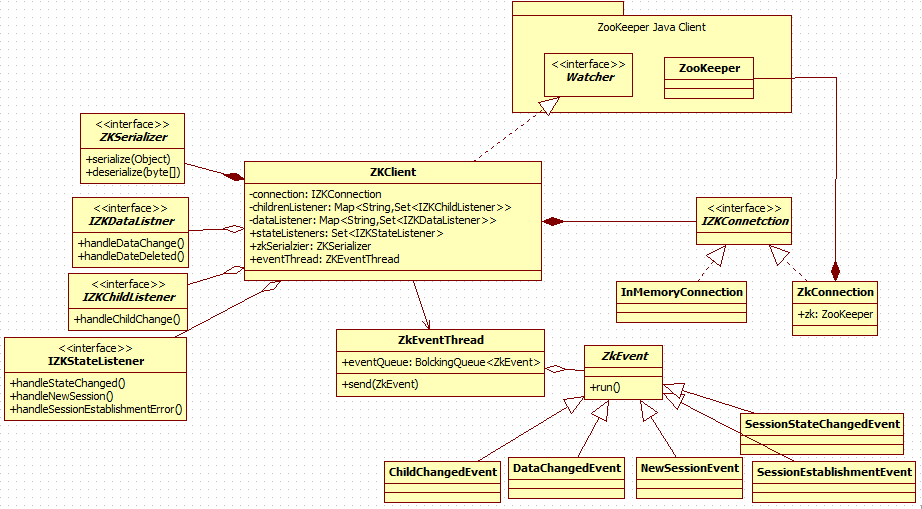
|  |
| --- |
| <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>0.10</version>  </dependency> |

或者

|  |
| --- |
| <dependency>  <groupId>com.github.sgroschupf</groupId>  <artifactId>zkclient</artifactId>  <version>0.1</version>  </dependency> |

Github：https://github.com/sgroschupf/zkclient

## **ZKClient的设计**



### **ZkClient的组件说明**

从上述结构上看，IZKConnection是一个ZkClient与ZooKeeper之间的一个适配器。在代码里直接使用的是ZKClient，其实质还是委托了zookeeper来处理了。

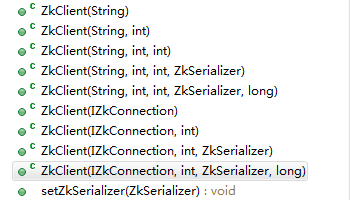
       前面有一篇文章中，已经说了，使用ZooKeeper客户端来注册watcher有几种方法：1、创建ZooKeeper对象时指定默认的Watcher，2、getData()，3、exists()，4、getchildren。其中getdata,exists注册的是某个节点的事件处理器（watcher），getchildren注册的是子节点的事件处理器（watcher）。而在ZKClient中，根据事件类型，分为了节点事件（数据事件）、子节点事件。对应的事件处理器则是IZKDataListener和IZKChildListener。另外加入了Session相关的事件和事件处理器。

       ZkEventThread是专门用来处理事件的线程。

### **重要处理流程说明**

#### **启动ZKClient**

在创建ZKClient对象时，就完成了到ZooKeeper服务器连接的建立。具体过程是这样的：



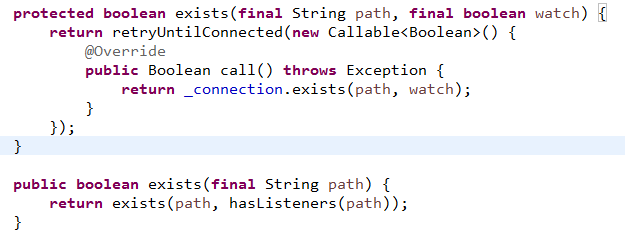
1、  启动时，指定好connection string，连接超时时间，序列化工具等。

2、  创建并启动eventThread，用于接收事件，并调度事件监听器Listener的执行。

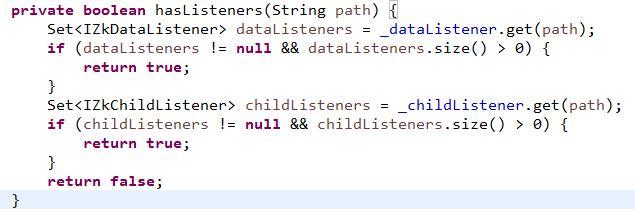
3、  连接到zookeeper服务器，同时将ZKClient自身作为默认的Watcher。

#### **为节点注册Watcher**

       ZooKeeper的三个方法：getData、getChildren、exists，ZKClient都提供了相应的代理方法。就拿exists来看：



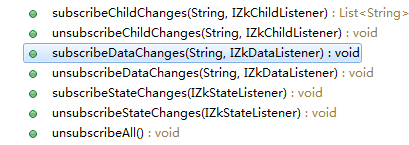
可以看到，是否注册watcher，由hasListeners(path)来决定的。



hasListeners就是看有没有与该数据节点绑定的listener。

所以呢，默认情况下，都会自动的为指定的path注册watcher，并且是默认的watcher（ZKClient）。怎么才能让hasListeners判定值为true呢，也就是怎么才能为path绑定Listener呢？

ZKClient提供了订阅功能：

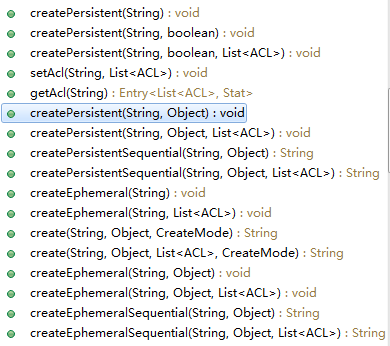


一个新建的会话，只需要在取得响应的数据节点后，调用subscribteXxx就可以订阅上相应的事件了。

#### **ZooKeeper的变更操作**

Zookeeper中提供的变更操作有：节点的创建、删除，节点数据的修改。

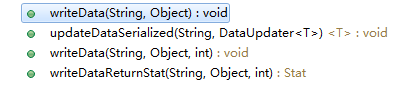
创建操作，数据节点分为四种，ZKClient分别为他们提供了相应的代理：



删除节点的操作：

IMG_262

修改节点数据的操作：



writeDataReturnStat（）：写数据并返回数据的状态。

updateDataSerialized（）：修改已序列化的数据。执行过程是：先读取数据，然后使用DataUpdater对数据修改，最后调用writeData将修改后的数据发送给服务端。

#### **客户端处理变更**

       前面已经知道，ZKClient是默认的Watcher，并且在为各个数据节点注册的Watcher都是这个默认的Watcher。那么该是如何将各种事件通知给相应的Listener呢？

处理过程大致可以概括为下面的步骤：

1、判断变更类型：变更类型分为State变更、ChildNode变更（创建子节点、删除子节点、修改子节点数据）、NodeData变更（创建指定node，删除节点，节点数据变更）。

2、取出与path关联的Listeners，并为每一个Listener创建一个ZKEvent，将ZkEvent交给ZkEventThread处理。

3、ZkEventThread线程，拿到ZkEvent后，只需要调用ZkEvent的run方法进行处理。

从这里也可以知道，具体的怎么如何调用Listener，还要依赖于ZkEvent的run()实现了。

#### **序列化处理**

ZooKeeper中，会涉及到序列化、反序列化的操作有两种：getData、setData。在ZKClient中，分别用readData、writeData来替代了。

对于readData：先调用zookeeper的getData，然后进行使用ZKSerializer进行反序列化工作。

对于writeData：先使用ZKSerializer将对象序列化后，再调用zookeeper的setData。

#### 注册监听

在ZkClient中客户端可以通过注册相关的事件监听来实现对Zookeeper服务端时间的订阅。其中ZkClient提供的监听事件接口有以下几种：

| 接口类 | 注册监听方法 | 解除监听方法 |
| --- | --- | --- |
| IZkChildListener | ZkClient的subscribeChildChanges方法 | ZkClient的unsubscribeChildChanges方法 |
| IZkDataListener | ZkClient的subscribeDataChanges方法 | ZkClient的subscribeChildChanges方法 |
| IZkStateListener | ZkClient的subscribeStateChanges方法 | ZkClient的unsubscribeStateChanges方法 |

其中ZkClient还提供了一个unsubscribeAll方法，来解除所有监听。

## **ZkClient如何解决使用ZooKeeper客户端遇到的问题的呢？**

Watcher自动重注册：这个要是依赖于hasListeners（）的判断，来决定是否再次注册。如果对此有不清晰的，可以看上面的流程处理的说明

       Session失效重连：如果发现会话过期，就先关闭已有连接，再重新建立连接。

       异常处理：对比ZooKeeper和ZKClient，就可以发现ZooKeeper的所有操作都是抛异常的，而ZKClient的所有操作，都不会抛异常的。在发生异常时，它或做日志，或返回空，或做相应的Listener调用。

相比于ZooKeeper官方客户端，使用ZKClient时，只需要关注实际的Listener实现即可。所以这个客户端，还是推荐大家使用的。