

# 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>
```

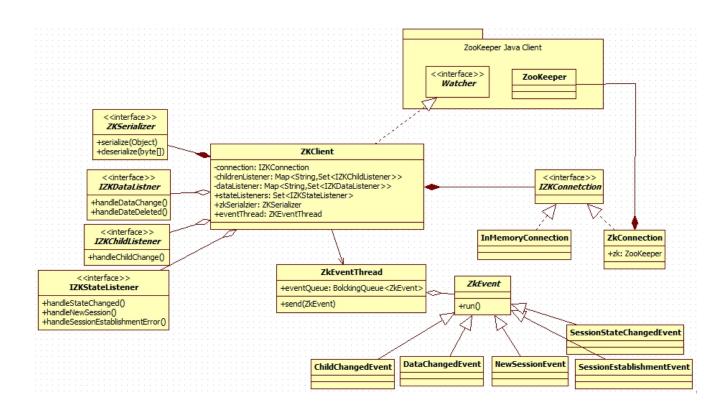
#### 或者

```
<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:

- C ZkClient(String)
- C ZkClient(String, int)
- C ZkClient(String, int, int)
- <sup>c</sup> ZkClient(String, int, int, ZkSerializer)
- <sup>c</sup> ZkClient(String, int, int, ZkSerializer, long)
- C ZkClient(IZkConnection)
- C ZkClient(IZkConnection, int)
- C ZkClient(IZkConnection, int, ZkSerializer)
- C ZkClient(IZkConnection, int, ZkSerializer, long)
- setZkSerializer(ZkSerializer) : void
- 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:



```
protected boolean exists(final String path, final boolean watch) {
     return retryUntilConnected(new Callable<Boolean>() {
         @Override
         public Boolean call() throws Exception {
             return connection.exists(path, watch);
     });
}
 public boolean exists(final String path) {
     return exists(path, hasListeners(path));
 }
可以看到,是否注册 watcher,由 hasListeners(path)来决定的。
 private boolean hasListeners(String path) {
     Set<IZkDataListener> dataListeners = dataListener.get(path);
     if (dataListeners != null && dataListeners.size() > 0) {
         return true;
     Set<IZkChildListener> childListeners = _childListener.get(path);
     if (childListeners != null && childListeners.size() > 0) {
         return true;
     return false;
}
```

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:

#### 架构师之家 www.jiagouedu.com

#### 图灵技术交流网站 http://www.jiagouedu.com

- subscribeChildChanges(String, IZkChildListener) : List < String >
- unsubscribeChildChanges(String, IZkChildListener): void
- subscribeDataChanges(String, IZkDataListener): void
- unsubscribeDataChanges(String, IZkDataListener): void
- subscribeStateChanges(IZkStateListener) : void
- unsubscribeStateChanges(IZkStateListener) : void
- unsubscribeAll(): void

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。

#### 客户端处理变更(watcher 通知)

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。

#### 注册监听 watcher

接口类	注册监听方法	解除监听方法



#### 图灵技术交流网站 http://www.jiagouedu.com

IZkChildListener	ZkClient的 subscribeChildChanges 方法	ZkClient 的 unsubscribeChildChanges 方法
(子节点)		
IZkDataListener	ZkClient 的 subscribeDataChanges 方法	ZkClient 的 unsubscribeDataChanges 方法
(数据)		
IZkStateListener	ZkClient 的 subscribeStateChanges 方	ZkClient 的 unsubscribeStateChanges 方法
(客户端状	法	
态)		

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:

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

#### ZooKeeper 的 CRUD

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:



#### 图灵技术交流网站 http://www.jiagouedu.com

- createPersistent(String) : void
- createPersistent(String, boolean): void
- o createPersistent(String, boolean, List<ACL>): void
- setAcl(String, List<ACL>): void
- getAcl(String): Entry < List < ACL > , Stat >
- createPersistent(String, Object): void
- createPersistent(String, Object, List<ACL>): void
- createPersistentSequential(String, Object): String
- createPersistentSequential(String, Object, List<ACL>): String
- createEphemeral(String): void
- o createEphemeral(String, List<ACL>): void
- create(String, Object, CreateMode) : String
- create(String, Object, List<ACL>, CreateMode) : String
- createEphemeral(String, Object) : void
- createEphemeral(String, Object, List<ACL>): void
- createEphemeralSequential(String, Object): String
- createEphemeralSequential(String, Object, List<ACL>): String

#### 删除节点的操作:

- delete(String): boolean
- delete(String, int) : boolean

#### 修改节点数据的操作:

- writeData(String, Object): void
- updateDataSerialized(String, DataUpdater<T>) <T> : void
- writeData(String, Object, int): void
- writeDataReturnStat(String, Object, int): Stat

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。

#### 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。

### 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。