**Thrift 非阻塞式IO服务模型-TNonblockingServer**

原创 2015年03月01日 14:56:11

* 标签：
* [RPC](http://so.csdn.net/so/search/s.do?q=RPC&t=blog) /
* [thrift](http://so.csdn.net/so/search/s.do?q=thrift&t=blog)
* 3886

使用非阻塞式IO，服务端和客户端需要指定 TFramedTransport 数据传输的方式。

服务端HelloTNonblockingServer.java：

**[java]** [view plain](http://blog.csdn.net/zhu_tianwei/article/details/44002929) [copy](http://blog.csdn.net/zhu_tianwei/article/details/44002929)

1. **package** cn.slimsmart.thrift.demo.helloworld;
3. **import** org.apache.thrift.TException;
4. **import** org.apache.thrift.TProcessor;
5. **import** org.apache.thrift.protocol.TCompactProtocol;
6. **import** org.apache.thrift.server.TNonblockingServer;
7. **import** org.apache.thrift.server.TServer;
8. **import** org.apache.thrift.transport.TFramedTransport;
9. **import** org.apache.thrift.transport.TNonblockingServerSocket;
11. /\*\*
12. \* 注册服务端
13. \*  使用非阻塞式IO，服务端和客户端需要指定 TFramedTransport 数据传输的方式。 TNonblockingServer
14. \*/
15. **public** **class** HelloTNonblockingServer {
16. // 注册端口
17. **public** **static** **final** **int** SERVER\_PORT = 8080;
19. **public** **static** **void** main(String[] args) **throws** TException {
20. //处理器
21. TProcessor tprocessor = **new** HelloWorld.Processor<HelloWorld.Iface>(**new** HelloWorldImpl());
22. // 传输通道 - 非阻塞方式
23. TNonblockingServerSocket serverTransport = **new** TNonblockingServerSocket(SERVER\_PORT);
24. //异步IO，需要使用TFramedTransport，它将分块缓存读取。
25. TNonblockingServer.Args tArgs = **new** TNonblockingServer.Args(serverTransport);
26. tArgs.processor(tprocessor);
27. tArgs.transportFactory(**new** TFramedTransport.Factory());
28. //使用高密度二进制协议
29. tArgs.protocolFactory(**new** TCompactProtocol.Factory());
30. // 使用非阻塞式IO，服务端和客户端需要指定TFramedTransport数据传输的方式
31. TServer server = **new** TNonblockingServer(tArgs);
32. System.out.println("HelloTNonblockingServer start....");
33. server.serve(); // 启动服务
34. }
35. }

客户端HelloNonblockingClient.java：

**[java]** [view plain](http://blog.csdn.net/zhu_tianwei/article/details/44002929) [copy](http://blog.csdn.net/zhu_tianwei/article/details/44002929)

1. **package** cn.slimsmart.thrift.demo.helloworld;
3. **import** org.apache.thrift.TException;
4. **import** org.apache.thrift.protocol.TCompactProtocol;
5. **import** org.apache.thrift.protocol.TProtocol;
6. **import** org.apache.thrift.transport.TFramedTransport;
7. **import** org.apache.thrift.transport.TSocket;
8. **import** org.apache.thrift.transport.TTransport;
10. /\*\*
11. \* 客户端调用HelloTNonblockingServer,HelloTHsHaServer
12. \* 非阻塞
13. \*/
14. **public** **class** HelloNonblockingClient {
15. **public** **static** **final** String SERVER\_IP = "127.0.0.1";
16. **public** **static** **final** **int** SERVER\_PORT = 8080;
17. **public** **static** **final** **int** TIMEOUT = 30000;
19. **public** **static** **void** main(String[] args) **throws** TException {
20. //设置传输通道，对于非阻塞服务，需要使用TFramedTransport，它将数据分块发送
21. TTransport transport = **new** TFramedTransport(**new** TSocket(SERVER\_IP, SERVER\_PORT, TIMEOUT));
22. // 协议要和服务端一致
23. //HelloTNonblockingServer
24. ////使用高密度二进制协议
25. TProtocol protocol = **new** TCompactProtocol(transport);
26. //HelloTHsHaServer
27. ////使用二进制协议
28. //TProtocol protocol = new TBinaryProtocol(transport);
29. HelloWorld.Client client = **new** HelloWorld.Client(protocol);
30. transport.open();
31. String result = client.sayHello("jack");
32. System.out.println("result : " + result);
33. //关闭资源
34. transport.close();
35. }
36. }

版权声明：

* 本文已收录于以下专栏：
* [RPC之Thrift学习实战](http://blog.csdn.net/column/details/slimina-thrift.html)