micmiu – 软件开发+生活点滴

Thrift入门及Java实例演示

作者: Michael 日期: 2012 年 6 月 14 日 查看评论 发表评论 (29)

目录：

[一]、概述

Thrift是一个软件框架，用来进行可扩展且跨语言的服务的开发。它结合了功能强大的软件堆栈和代码生成引擎，以构建在 C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml 等等编程语言间无缝结合的、高效的服务。

Thrift最初由facebook开发，07年四月开放源码，08年5月进入apache孵化器。thrift允许你定义一个简单的定义文件中的数据类型和服务接口。以作为输入文件，编译器生成代码用来方便地生成RPC客户端和服务器通信的无缝跨编程语言。

官网地址：thrift.apache.org

推荐值得一看的文章：

- http://jnb.ociweb.com/jnb/jnbJun2009.html
- http://wiki.apache.org/thrift
- http://thrift.apache.org/static/files/thrift-20070401.pdf

[二]、下载配置

到官网下载最新版本，截止今日（2012-06-11）最新版本为0.8.0.

1. 如果是Maven构建项目的，直接在pom.xml 中添加如下内容：

```XHTML
<dependency>
  <groupId>org.apache.thrift</groupId>
  <artifactId>libthrift</artifactId>
  <version>0.8.0</version>
</dependency>
```

2.如果自己编译lib包，把下载的压缩包解压到X:盘，然后在X:\thrift-0.8.0\lib\java 目录下运行ant进行自动编译，会在X:\thrift-0.8.0\lib\java\build\ 目录下看到编译好的lib包：libthrift-0.8.0.jar

[三]、基本概念

1.数据类型

- 基本类型：
  - bool：布尔值，true 或 false，对应 Java 的 boolean
  - byte：8 位有符号整数，对应 Java 的 byte
  - i16：16 位有符号整数，对应 Java 的 short
  - i32：32 位有符号整数，对应 Java 的 int
  - i64：64 位有符号整数，对应 Java 的 long
  - double：64 位浮点数，对应 Java 的 double
  - string：utf-8编码的字符串，对应 Java 的 String
- 结构体类型：
  - struct：定义公共的对象，类似于 C 语言中的结构体定义，在 Java 中是一个 JavaBean
- 容器类型：
  - list：对应 Java 的 ArrayList
  - set：对应 Java 的 HashSet
  - map：对应 Java 的 HashMap
- 异常类型：
  - exception：对应 Java 的 Exception
- 服务类型：
  - service：对应服务的类

2.服务端编码基本步骤：

- 实现服务处理接口impl
- 创建TProcessor
- 创建TServerTransport
- 创建TProtocol
- 创建TServer
- 启动Server

3.客户端编码基本步骤：

- 创建Transport
- 创建TProtocol
- 基于TTransport和TProtocol创建 Client
- 调用Client的相应方法

4.数据传输协议

- TBinaryProtocol：二进制格式.
- TCompactProtocol：压缩格式
- TJSONProtocol：JSON格式
- TSimpleJSONProtocol：提供JSON只写协议, 生成的文件很容易通过脚本语言解析

tips:客户端和服务端的协议要一致

[四]、实例演示

1. thrift生成代码

创建Thrift文件：G:\test\thrift\demoHello.thrift ,内容如下：

```
namespace java com.micmiu.thrift.demo

service  HelloWorldService {
  string sayHello(1:string username)
}
```

目录结构如下：

文章归档

选择月份

标签云(3D)

```
G:\test\thrift&gt;tree /F
卷 other 的文件夹 PATH 列表
卷序列号为 D238-BE47
G:.
    demoHello.thrift
    demouser.thrift
    thrift-0.8.0.exe

没有子文件夹
```

thrift-0.8.0.exe 是官网提供的windows下编译工具，运用这个工具生成相关代码：

```
thrift-0.8.0.exe -r -gen java ./demoHello.thrift
```

生成后的目录结构如下：

```
G:\test\thrift&gt;tree /F
卷 other 的文件夹 PATH 列表
卷序列号为 D238-BE47
G:.
|  demoHello.thrift
|  demouser.thrift
|  thrift-0.8.0.exe
|
└─gen-java
    └─com
        └─micmiu
            └─thrift
                └─demo
                        HelloWorldService.java
```

将生成的HelloWorldService.java 文件copy到自己测试的工程中，我的工程是用maven构建的，故在pom.xml中增加如下内容：

```XHTML
<dependency>
    <groupId>org.apache.thrift</groupId>
    <artifactId>libthrift</artifactId>
    <version>0.8.0</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.5.8</version>
</dependency>
```

2. 实现接口Iface

java代码：HelloWorldImpl.java

```
package com.micmiu.thrift.demo;

import org.apache.thrift.TException;

/**
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloWorldImpl implements HelloWorldService.Iface {

    public HelloWorldImpl() {
    }

    @Override
    public String sayHello(String username) throws TException {
        return "Hi," + username + " welcome to my blog www.micmiu.com";
    }

}
```

3.TSimpleServer服务端

简单的单线程服务模型，一般用于测试。

编写服务端server代码：HelloServerDemo.java

```
package com.micmiu.thrift.demo;

```

```
import org.apache.thrift.TProcessor;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.protocol.TCompactProtocol;
import org.apache.thrift.protocol.TJSONProtocol;
import org.apache.thrift.protocol.TSimpleJSONProtocol;
import org.apache.thrift.server.TServer;
import org.apache.thrift.server.TSimpleServer;
import org.apache.thrift.transport.TServerSocket;

/**
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloServerDemo {
    public static final int SERVER_PORT = 8090;

    public void startServer() {
        try {
            System.out.println("HelloWorld TSimpleServer start ....");

            TProcessor tprocessor = new HelloWorldService.Processor&lt;HelloWorldS
                    new HelloWorldImpl());
            // HelloWorldService.Processor&lt;HelloWorldService.Iface&gt; tprocess
            // new HelloWorldService.Processor&lt;HelloWorldService.Iface&gt;(
            // new HelloWorldImpl());

            // 简单的单线程服务模型，一般用于测试
            TServerSocket serverTransport = new TServerSocket(SERVER_PORT);
            TServer.Args tArgs = new TServer.Args(serverTransport);
            tArgs.processor(tprocessor);
            tArgs.protocolFactory(new TBinaryProtocol.Factory());
            // tArgs.protocolFactory(new TCompactProtocol.Factory());
            // tArgs.protocolFactory(new TJSONProtocol.Factory());
            TServer server = new TSimpleServer(tArgs);
            server.serve();

        } catch (Exception e) {
            System.out.println("Server start error!!!");
            e.printStackTrace();
        }
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        HelloServerDemo server = new HelloServerDemo();
        server.startServer();
    }

}
```

编写客户端Client代码：HelloClientDemo.java

```
package com.micmiu.thrift.demo;

import org.apache.thrift.TException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.protocol.TCompactProtocol;
import org.apache.thrift.protocol.TJSONProtocol;
import org.apache.thrift.protocol.TProtocol;
import org.apache.thrift.transport.TSocket;
import org.apache.thrift.transport.TTransport;
import org.apache.thrift.transport.TTransportException;

/**
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloClientDemo {

    public static final String SERVER_IP = "localhost";
    public static final int SERVER_PORT = 8090;
    public static final int TIMEOUT = 30000;

    /**
     *
     * @param userName
     */
    public void startClient(String userName) {
        TTransport transport = null;
        try {
            transport = new TSocket(SERVER_IP, SERVER_PORT, TIMEOUT);
            // 协议要和服务端一致
            TProtocol protocol = new TBinaryProtocol(transport);
            // TProtocol protocol = new TCompactProtocol(transport);
            // TProtocol protocol = new TJSONProtocol(transport);
```

```
36              HelloWorldService.Client client = new HelloWorldService.Client(
37                      protocol);
38              transport.open();
39              String result = client.sayHello(userName);
40              System.out.println("Thrify client result =: " + result);
41          } catch (TTransportException e) {
42              e.printStackTrace();
43          } catch (TException e) {
44              e.printStackTrace();
45          } finally {
46              if (null != transport) {
47                  transport.close();
48              }
49          }
50      }
51
52      /**
53       * @param args
54       */
55      public static void main(String[] args) {
56          HelloClientDemo client = new HelloClientDemo();
57          client.startClient("Michael");
58
59      }
60
61  }
```

先运行服务端程序，日志如下：

```
1  HelloWorld TSimpleServer start ....
```

再运行客户端调用程序，日志如下：

```
1  Thrify client result =: Hi,Michael welcome to my blog www.micmiu.com
```

测试成功，和预期的返回信息一致。

4.TThreadPoolServer 服务模型

线程池服务模型，使用标准的阻塞式IO，预先创建一组线程处理请求。

编写服务端代码：HelloServerDemo.java

```
1  package com.micmiu.thrift.demo;
2
3  import org.apache.thrift.TProcessor;
4  import org.apache.thrift.protocol.TBinaryProtocol;
5  import org.apache.thrift.server.TServer;
6  import org.apache.thrift.server.TThreadPoolServer;
7  import org.apache.thrift.transport.TServerSocket;
8
9  /**
10  * blog http://www.micmiu.com
11  *
12  * @author Michael
13  *
14  */
15 public class HelloServerDemo {
16      public static final int SERVER_PORT = 8090;
17
18      public void startServer() {
19          try {
20              System.out.println("HelloWorld TThreadPoolServer start ....");
21
22              TProcessor tprocessor = new HelloWorldService.Processor&lt;HelloWorldS
23                      new HelloWorldImpl());
24
25               TServerSocket serverTransport = new TServerSocket(SERVER_PORT);
26               TThreadPoolServer.Args ttpsArgs = new TThreadPoolServer.Args(
27               serverTransport);
28               ttpsArgs.processor(tprocessor);
29               ttpsArgs.protocolFactory(new TBinaryProtocol.Factory());
30
31              // 线程池服务模型，使用标准的阻塞式IO，预先创建一组线程处理请求。
32               TServer server = new TThreadPoolServer(ttpsArgs);
33               server.serve();
34
35          } catch (Exception e) {
36              System.out.println("Server start error!!!");
37              e.printStackTrace();
38          }
39      }
40
41      /**
```

```
     * @param args
     */
    public static void main(String[] args) {
        HelloServerDemo server = new HelloServerDemo();
        server.startServer();
    }

}
```

客户端Client代码和之前的一样，只要数据传输的协议一致即可，客户端测试成功，结果如下：

```
Thrify client result =: Hi,Michael welcome to my blog www.micmiu.com
```

5.TNonblockingServer 服务模型

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。

编写服务端代码：HelloServerDemo.java

```
package com.micmiu.thrift.demo;

import org.apache.thrift.TProcessor;
import org.apache.thrift.protocol.TCompactProtocol;
import org.apache.thrift.server.TNonblockingServer;
import org.apache.thrift.server.TServer;
import org.apache.thrift.transport.TFramedTransport;
import org.apache.thrift.transport.TNonblockingServerSocket;

/**
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloServerDemo {
    public static final int SERVER_PORT = 8090;

    public void startServer() {
        try {
            System.out.println("HelloWorld TNonblockingServer start ....");

            TProcessor tprocessor = new HelloWorldService.Processor&lt;HelloWorldS
                    new HelloWorldImpl());

            TNonblockingServerSocket tnbSocketTransport = new TNonblockingServerSo
                    SERVER_PORT);
            TNonblockingServer.Args tnbArgs = new TNonblockingServer.Args(
                    tnbSocketTransport);
            tnbArgs.processor(tprocessor);
            tnbArgs.transportFactory(new TFramedTransport.Factory());
            tnbArgs.protocolFactory(new TCompactProtocol.Factory());

            // 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
            TServer server = new TNonblockingServer(tnbArgs);
            server.serve();

        } catch (Exception e) {
            System.out.println("Server start error!!!");
            e.printStackTrace();
        }
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        HelloServerDemo server = new HelloServerDemo();
        server.startServer();
    }

}
```

编写客户端代码：HelloClientDemo.java

```
package com.micmiu.thrift.demo;

import org.apache.thrift.TException;
import org.apache.thrift.protocol.TCompactProtocol;
import org.apache.thrift.protocol.TProtocol;
import org.apache.thrift.transport.TFramedTransport;
import org.apache.thrift.transport.TSocket;
import org.apache.thrift.transport.TTransport;
import org.apache.thrift.transport.TTransportException;

/**
```

```
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloClientDemo {

    public static final String SERVER_IP = "localhost";
    public static final int SERVER_PORT = 8090;
    public static final int TIMEOUT = 30000;

    /**
     *
     * @param userName
     */
    public void startClient(String userName) {
        TTransport transport = null;
        try {
            transport = new TFramedTransport(new TSocket(SERVER_IP,
                    SERVER_PORT, TIMEOUT));
            // 协议要和服务端一致
            TProtocol protocol = new TCompactProtocol(transport);
            HelloWorldService.Client client = new HelloWorldService.Client(
                    protocol);
            transport.open();
            String result = client.sayHello(userName);
            System.out.println("Thrify client result =: " + result);
        } catch (TTransportException e) {
            e.printStackTrace();
        } catch (TException e) {
            e.printStackTrace();
        } finally {
            if (null != transport) {
                transport.close();
            }
        }
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        HelloClientDemo client = new HelloClientDemo();
        client.startClient("Michael");

    }

}
```

客户端的测试成功，结果如下：

```
Thrify client result =: Hi,Michael welcome to my blog www.micmiu.com
```

6.THsHaServer服务模型

半同步半异步的服务端模型，需要指定为： TFramedTransport 数据传输的方式。

编写服务端代码：HelloServerDemo.java

```
package com.micmiu.thrift.demo;

import org.apache.thrift.TProcessor;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.protocol.TCompactProtocol;
import org.apache.thrift.server.THsHaServer;
import org.apache.thrift.server.TNonblockingServer;
import org.apache.thrift.server.TServer;
import org.apache.thrift.server.TSimpleServer;
import org.apache.thrift.server.TThreadPoolServer;
import org.apache.thrift.transport.TFramedTransport;
import org.apache.thrift.transport.TNonblockingServerSocket;
import org.apache.thrift.transport.TServerSocket;

/**
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloServerDemo {
    public static final int SERVER_PORT = 8090;

    public void startServer() {
        try {
            System.out.println("HelloWorld THsHaServer start ....");

            TProcessor tprocessor = new HelloWorldService.Processor&lt;HelloWorldS
```

```
                    new HelloWorldImpl());

            TNonblockingServerSocket tnbSocketTransport = new TNonblockingServerSo
                    SERVER_PORT);
            THsHaServer.Args thhsArgs = new THsHaServer.Args(tnbSocketTransport);
            thhsArgs.processor(tprocessor);
            thhsArgs.transportFactory(new TFramedTransport.Factory());
            thhsArgs.protocolFactory(new TBinaryProtocol.Factory());

            //半同步半异步的服务模型
            TServer server = new THsHaServer(thhsArgs);
            server.serve();

        } catch (Exception e) {
            System.out.println("Server start error!!!");
            e.printStackTrace();
        }
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        HelloServerDemo server = new HelloServerDemo();
        server.startServer();
    }

}
```

客户端代码和上面 4 中的类似，只要注意传输协议一致以及指定传输方式为TFramedTransport。

7.异步客户端

编写服务端代码：HelloServerDemo.java

```
package com.micmiu.thrift.demo;

import org.apache.thrift.TProcessor;
import org.apache.thrift.protocol.TCompactProtocol;
import org.apache.thrift.server.TNonblockingServer;
import org.apache.thrift.server.TServer;
import org.apache.thrift.transport.TFramedTransport;
import org.apache.thrift.transport.TNonblockingServerSocket;

/**
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloServerDemo {
    public static final int SERVER_PORT = 8090;

    public void startServer() {
        try {
            System.out.println("HelloWorld TNonblockingServer start ....");

            TProcessor tprocessor = new HelloWorldService.Processor&lt;HelloWorldS
                    new HelloWorldImpl());

            TNonblockingServerSocket tnbSocketTransport = new TNonblockingServerSo
                    SERVER_PORT);
            TNonblockingServer.Args tnbArgs = new TNonblockingServer.Args(
                    tnbSocketTransport);
            tnbArgs.processor(tprocessor);
            tnbArgs.transportFactory(new TFramedTransport.Factory());
            tnbArgs.protocolFactory(new TCompactProtocol.Factory());

            // 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
            TServer server = new TNonblockingServer(tnbArgs);
            server.serve();

        } catch (Exception e) {
            System.out.println("Server start error!!!");
            e.printStackTrace();
        }
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        HelloServerDemo server = new HelloServerDemo();
        server.startServer();
    }

}
```

编写客户端Client代码：HelloAsynClientDemo.java

```java
package com.micmiu.thrift.demo;

import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;

import org.apache.thrift.TException;
import org.apache.thrift.async.AsyncMethodCallback;
import org.apache.thrift.async.TAsyncClientManager;
import org.apache.thrift.protocol.TCompactProtocol;
import org.apache.thrift.protocol.TProtocolFactory;
import org.apache.thrift.transport.TNonblockingSocket;
import org.apache.thrift.transport.TNonblockingTransport;

import com.micmiu.thrift.demo.HelloWorldService.AsyncClient.sayHello_call;

/**
 * blog http://www.micmiu.com
 *
 * @author Michael
 *
 */
public class HelloAsynClientDemo {

    public static final String SERVER_IP = "localhost";
    public static final int SERVER_PORT = 8090;
    public static final int TIMEOUT = 30000;

    /**
     *
     * @param userName
     */
    public void startClient(String userName) {
        try {
            TAsyncClientManager clientManager = new TAsyncClientManager();
            TNonblockingTransport transport = new TNonblockingSocket(SERVER_IP,
                    SERVER_PORT, TIMEOUT);

            TProtocolFactory tprotocol = new TCompactProtocol.Factory();
            HelloWorldService.AsyncClient asyncClient = new HelloWorldService.Asyn
                    tprotocol, clientManager, transport);
            System.out.println("Client start .....");

            CountDownLatch latch = new CountDownLatch(1);
            AsynCallback callBack = new AsynCallback(latch);
            System.out.println("call method sayHello start ...");
            asyncClient.sayHello(userName, callBack);
            System.out.println("call method sayHello .... end");
            boolean wait = latch.await(30, TimeUnit.SECONDS);
            System.out.println("latch.await =:" + wait);
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("startClient end.");
    }

    public class AsynCallback implements AsyncMethodCallback&lt;sayHello_call&gt;
        private CountDownLatch latch;

        public AsynCallback(CountDownLatch latch) {
            this.latch = latch;
        }

        @Override
        public void onComplete(sayHello_call response) {
            System.out.println("onComplete");
            try {
                // Thread.sleep(1000L * 1);
                System.out.println("AsynCall result =:"
                        + response.getResult().toString());
            } catch (TException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                latch.countDown();
            }
        }

        @Override
        public void onError(Exception exception) {
            System.out.println("onError :" + exception.getMessage());
            latch.countDown();
        }
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        HelloAsynClientDemo client = new HelloAsynClientDemo();
```

```
91          client.startClient("Michael");
92
93      }
94
95  }
```

先运行服务程序，再运行客户端程序，测试结果如下：

```
1  Client start .....
2  call method sayHello start ...
3  call method sayHello .... end
4  onComplete
5  AsynCall result =:Hi,Michael welcome to my blog www.micmiu.com
6  latch.await =:true
7  startClient end.
```

18

RPC apache, RPC, Thrift

← Apache Ivy入门 JRobin绘制指定时间段的流量图 →

发表评论？ 29 条评论。

while.for 2015 年 11 月 5 日 在 下午 5:56 回复

对nifty有研究么？几乎木有什么资料。。。抓狂ing...

回复

Michael 2015 年 11 月 6 日 在 下午 4:57

还个还真没有接触过

东辉 2015 年 9 月 27 日 在 下午 8:05 回复

非常感谢

东辉 2015 年 9 月 27 日 在 下午 8:00 回复

非常好，最近项目中应用到Thrift。学习了，程序也可以跑起来！

kou 2015 年 8 月 20 日 在 下午 5:27 回复

谢谢啦

水煮鱼 2015 年 8 月 17 日 在 下午 12:00 回复

非常不错,刚刚接触这个东西。看了博主的文章懂了些基础的东西。感谢

abc 2015 年 5 月 8 日 在 上午 10:56 回复

楼主有qq吗

leo 2015 年 4 月 30 日 在 上午 11:57 回复

使用 thrift 工具生成的java 文件加载到项目中提示出现大量的：
Error:(383, 20) java: service.demo.Hello.Processor.helloString不是抽象的, 并且未覆盖org.apache.thrift.ProcessFunction中的抽象方法isOneway()
错误信息。 请教楼主怎么破？

zhxh007 2015 年 4 月 25 日 在 下午 2:02 回复

楼主用的eclispe么，什么主题啊，挺好看的

颜颜 2014 年 12 月 1 日 在 下午 11:53 回复

请教下博主 用半异步半同步的时候 我开启了多个服务每个都指定了一个端口号 然后 客户端链接的时候 一到执行实现服务接口的方法时就客户端保持了。。既不往下运行也不停止。。。。 可能是什么原因造成的呢。。谢谢啦！

回复

Michael 2014 年 12 月 9 日 在 下午 11:15

既不继续运行也不停止，这只是个表面现象，要找到真正原因（死锁、挂起 还是。。。）才能解决

原 2014 年 11 月 6 日 在 上午 11:52

回复

博主你好，7的代码和5的代码有什么区别呢？

回复

Michael 2014 年 11 月 24 日 在 下午 4:32

没有实际对比过，估计差异不大，最好看看官方这两个版本的说明

glutton 2014 年 4 月 4 日 在 上午 11:49

回复

博主你好，文章非常有用
就是在最后一个例子，我的异步客户端输出如下：
Client start …..
call method sayHello start …
call method sayHello …. end
latch.await =:false
startClient end.

没有获得和您一样的结果，大约是什么原因呢？感谢

glutton 2014 年 4 月 4 日 在 上午 11:48

回复

博主你好，文章非常有用
就是在最后一个例子，我的异步客户端输出如下：

清风 2014 年 3 月 31 日 在 下午 6:08

回复

请问 Thrift的数据 是如何进行传递的
和普通的WebService的区别仅在于 WS传递的是XML 而Thrift传递的是thrift包？
thrift的封包解包和WS的有什么区别
为什么thrift比用CXF实现的WS要快（PS.我自己做了个实验，前者的耗时大概是后者的1/3到1/10）

回复

Michael 2014 年 3 月 31 日 在 下午 11:00

thrift的传输数据采用二进制格式，相比xml、json 体积更小，对于高并发、大数据量更有优势

milan111 2013 年 5 月 28 日 在 上午 10:22

回复

非常感谢！受益匪浅！
您给出的例子都是java通信的，那如果我想用java调用nodejs或者nodejs调用java该怎么写呢？有类似的例子吗？

回复

Michael 2013 年 5 月 30 日 在 上午 10:07

这个目前没有例子的

qq663550185 2013 年 1 月 14 日 在 下午 7:00

回复

请问在服务端有遇到过这种情况么？
Thrift error occurred during processing of message.
org.apache.thrift.TException: Message length exceeded: 2
at org.apache.thrift.protocol.TBinaryProtocol.checkReadLength(TBinaryProtocol.java:393)
at org.apache.thrift.protocol.TBinaryProtocol.readAll(TBinaryProtocol.java:377)
at org.apache.thrift.protocol.TBinaryProtocol.readI16(TBinaryProtocol.java:278)
at org.apache.thrift.protocol.TBinaryProtocol.readFieldBegin(TBinaryProtocol.java:229)

回复

Michael 2013 年 1 月 14 日 在 下午 10:53

没有遇到过的，看错误信息 好像是消息读取时有问题

jose 2012 年 10 月 16 日 在 下午 5:24

回复

请教一个问题
假如有2个service 如
HelloWorldService1
HelloWorldService2.

那么server端的code 如何写
System.out.println("HelloWorld TNonblockingServer start ….");
22

```
23
TProcessor tprocessor = new HelloWorldService.Processor(
24
new HelloWorldImpl());
25

26
TNonblockingServerSocket tnbSocketTransport = new TNonblockingServerSocket(
27
SERVER_PORT);
28
TNonblockingServer.Args tnbArgs = new TNonblockingServer.Args(
29
tnbSocketTransport);
30
tnbArgs.processor(tprocessor);
31
tnbArgs.transportFactory(new TFramedTransport.Factory());
32
tnbArgs.protocolFactory(new TCompactProtocol.Factory());
TServer server = new THsHaServer(thhsArgs);
40
server.serve();
```

这段代码是把 HelloWorldService 启动了 ，如何还想启动一个service 怎么办？

谢谢赐教！

回复

Michael 2012 年 10 月 16 日 在 下午 6:09

如果你想在一个port上起多个好像不行（我没有深入研究过），实在不行可以把两个service的接口合并成一个

jose 2012 年 10 月 16 日 在 下午 5:14

回复

写的不错

dhpasa 2012 年 9 月 13 日 在 下午 2:51

回复

可以啦接着看下去，呵呵。thank u

dhpasa 2012 年 9 月 13 日 在 下午 2:37

回复

生成代码具体是怎么实现的啊？看不明白

回复

Michael 2012 年 9 月 13 日 在 下午 2:48

那一步没有看懂？我好想写得已经很详细了

程序娃娃 2012 年 6 月 27 日 在 下午 5:33

回复

非常感谢

回复

Michael 2012 年 6 月 28 日 在 上午 9:51

以后可以多多交流

发表评论

昵称

邮箱

网址

注意 - 你可以用以下 HTML tags and attributes:

```
<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite>
<code class="" title="" data-url=""> <del datetime=""> <em> <i> <q cite=""> <strike>
<strong> <pre class="" title="" data-url=""> <span class="" title="" data-url="">
```

提交

↑ 回到顶部