

# 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 :

- 概述
- 下载配置
- 基本概念
  - 数据类型
  - 服务端编码基本步骤
  - 客户端编码基本步骤
  - 数据传输协议
- 实例演示 ( java )
  - thrift生成代码
  - 实现接口Iface
  - TSimpleServer服务模型
  - TThreadPoolServer 服务模型
  - TNonblockingServer 服务模型
  - THsHaServer服务模型
  - 异步客户端

### [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](http://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
1 <dependency>	
2 <groupId>org.apache.thrift</groupId>	
3 <artifactId>libthrift</artifactId>	
4 <version>0.8.0</version>	
5 </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

### 微博



孙吴空的爸爸 上海 浦东新区

加关注

#生态414硬件免费日#居然玩这么大？4月14日，@乐视商城 买会员0元送手机、电视、配件，再送生态礼包！还有最高414元优惠券等你抢哦！这些还不够？分享到微博，也有机会免费得超级电视哟！@赵一成\_ <http://t.cn/RqZoNwH>

4月5日 09:01

转发 | 评论

### 卧槽

互联网的那点事：爱奇艺通过软件强

### 近期文章

java.lang.NoSuchMethodError:  
org.eclipse.jdt.internal.compiler.CompilationResul  
Hadoop在Map阶段获取当前split的文件名  
hadoop多硬盘配置注意点  
Hadoop修改配置PID文件路径  
HBase安装配置snappy压缩算法  
Hadoop安装配置snappy压缩  
Happy New Year!  
HBase+Hadoop2 NN HA+Zookeeper独立安装的整合  
Hadoop2分布式及NN和RM实现HA的实验  
ansible requires a json module, none found

### 浏览

图文介绍IntelliJ IDEA 创建基于Maven构建的Web项目 - 56,089 views  
Thrift入门及Java实例演示 - 48,331 views  
SSO之CAS单点登录实例演示 - 35,331 views  
图文介绍openLDAP在windows上的安装配置 - 28,232 views  
java自带的MD5、SHA1算法演示 - 27,091 views  
java keytool证书工具使用小结 - 25,583 views  
Eclipse+Maven构建web项目及部署时Maven lib依赖问题的解决 - 22,316 views  
Maven构建web项目在Eclipse中部署的几种方法 - 21,060 views  
\$JAVA\_HOME环境变量在Mac OS X中设置的问题 - 17,883 views  
Openfire服务端源代码开发配置指南 - 17,569 views

### 分类目录

J2EE (42)  
Hibernate (12)  
Struts (5)  
Spring (11)

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：

```
1 namespace java com.micmiu.thrift.demo
2
3 service HelloWorldService {
4     string sayHello(1:string username)
5 }
```

目录结构如下：

http://www.micmiu.com/soa/rpc/thrift-sample/

JDBC	(6)
JTA	(3)
EJB	(4)
framework	(1)
海量数据	(36)
Hadoop	(17)
Hive	(8)
HBase	(7)
Spark	(2)
Sqoop	(2)
运维	(1)
ansible	(1)
SOA	(14)
webservice	(12)
RPC	(2)
OpenSource	(33)
openflashchart	(2)
生成文档	(5)
Openfire	(4)
Security	(1)
CORBA	(4)
Nutch	(6)
架构设计	(1)
缓存	(1)
企业应用	(24)
SNMP	(8)
SSO	(4)
服务器软件	(10)
编程语言	(37)
Java	(25)
javascript	(5)
Groovy	(4)
nodejs	(1)
Go	(2)
NoSQL	(12)
Berkeley	(6)
数据库	(10)
Oracle	(7)
mysql	(2)
Android	(4)
web UI	(1)
软件工具	(30)
常用软件	(13)
项目构建	(15)
测试工具	(2)
操作系统	(26)
linux	(19)
windows	(2)
Mac	(5)
小小	(9)
心底软绵绵	(3)
时间静悄悄	(6)
异常处理	(23)
杂谈	(2)

文章归档

选择月份

标签云(3D)

```
1 G:\test\thrift>tree /F
2 卷 other 的文件夹 PATH 列表
3 卷序列号为 D238-BE47
4 G:.
5     demoHello.thrift
6     demouser.thrift
7     thrift-0.8.0.exe
8
9 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：

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

生成后的目录结构如下：

```
1 G:\test\thrift>tree /F
2 卷 other 的文件夹 PATH 列表
3 卷序列号为 D238-BE47
4 G:.
5 | demoHello.thrift
6 | demouser.thrift
7 | thrift-0.8.0.exe
8 |
9 |_gen-java
10 |   |_com
11 |     |_micmiu
12 |       |_thrift
13 |         |_demo
14 |           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

```
1 <dependency>
2   <groupId>org.apache.thrift</groupId>
3   <artifactId>libthrift</artifactId>
4   <version>0.8.0</version>
5 </dependency>
6 <dependency>
7   <groupId>org.slf4j</groupId>
8   <artifactId>slf4j-log4j12</artifactId>
9   <version>1.5.8</version>
10 </dependency>
```

2. 实现接口Iface

java代码：HelloWorldImpl.java

```
1 package com.micmiu.thrift.demo;
2
3 import org.apache.thrift.TException;
4
5 /**
6  * blog http://www.micmiu.com
7  *
8  * @author Michael
9  *
10 */
11 public class HelloWorldImpl implements HelloWorldService.Iface {
12
13     public HelloWorldImpl() {
14     }
15
16     @Override
17     public String sayHello(String username) throws TException {
18         return "Hi," + username + " welcome to my blog www.micmiu.com";
19     }
20
21 }
```

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

```
1 package com.micmiu.thrift.demo;
2
```

技术链接

- 10→me@iteye
- 21→nosqlfan
- 22→mysqlslops
- 23→OpenCloudDB
- 31→CrazyJvm
- 32→封神无度
- 33→董的博客

功能

- 登录
- 文章RSS
- 评论RSS
- WordPress.org

```

3 import org.apache.thrift.TProcessor;
4 import org.apache.thrift.protocol.TBinaryProtocol;
5 import org.apache.thrift.protocol.TCompactProtocol;
6 import org.apache.thrift.protocol.TJSONProtocol;
7 import org.apache.thrift.protocol.TSimpleJSONProtocol;
8 import org.apache.thrift.server.TServer;
9 import org.apache.thrift.server.TSimpleServer;
10 import org.apache.thrift.transport.TServerSocket;
11
12 /**
13  * blog http://www.micmiu.com
14  *
15  * @author Michael
16  *
17  */
18 public class HelloServerDemo {
19     public static final int SERVER_PORT = 8090;
20
21     public void startServer() {
22         try {
23             System.out.println("HelloWorld TSimpleServer start ...");
24
25             TProcessor tprocessor = new HelloWorldService.Processor<HelloWorldService.Iface>(
26                 new HelloWorldImpl());
27             // HelloWorldService.Processor<HelloWorldService.Iface> tprocessor =
28             // new HelloWorldService.Processor<HelloWorldService.Iface>(
29             // new HelloWorldImpl());
30
31             // 简单的单线程服务模型，一般用于测试
32             TServerSocket serverTransport = new TServerSocket(SERVER_PORT);
33             TServer.Args tArgs = new TServer.Args(serverTransport);
34             tArgs.processor(tprocessor);
35             tArgs.protocolFactory(new TBinaryProtocol.Factory());
36             // tArgs.protocolFactory(new TCompactProtocol.Factory());
37             // tArgs.protocolFactory(new TJSONProtocol.Factory());
38             TServer server = new TSimpleServer(tArgs);
39             server.serve();
40
41         } catch (Exception e) {
42             System.out.println("Server start error!!!");
43             e.printStackTrace();
44         }
45     }
46
47     /**
48      * @param args
49      */
50     public static void main(String[] args) {
51         HelloServerDemo server = new HelloServerDemo();
52         server.startServer();
53     }
54
55 }

```

编写客户端Client代码：HelloClientDemo.java

```

1 package com.micmiu.thrift.demo;
2
3 import org.apache.thrift.TException;
4 import org.apache.thrift.protocol.TBinaryProtocol;
5 import org.apache.thrift.protocol.TCompactProtocol;
6 import org.apache.thrift.protocol.TJSONProtocol;
7 import org.apache.thrift.protocol.TProtocol;
8 import org.apache.thrift.transport.TSocket;
9 import org.apache.thrift.transport.TTransport;
10 import org.apache.thrift.transport.TTransportException;
11
12 /**
13  * blog http://www.micmiu.com
14  *
15  * @author Michael
16  *
17  */
18 public class HelloClientDemo {
19
20     public static final String SERVER_IP = "localhost";
21     public static final int SERVER_PORT = 8090;
22     public static final int TIMEOUT = 30000;
23
24     /**
25      *
26      * @param userName
27      */
28     public void startClient(String userName) {
29         TTransport transport = null;
30         try {
31             transport = new TSocket(SERVER_IP, SERVER_PORT, TIMEOUT);
32             // 协议要和服务端一致
33             TProtocol protocol = new TBinaryProtocol(transport);
34             // TProtocol protocol = new TCompactProtocol(transport);
35             // 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 (Exception 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<>(new HelloWorldImpl());
23
24              TServerSocket serverTransport = new TServerSocket(SERVER_PORT);
25              TThreadPoolServer.Args ttpsArgs = new TThreadPoolServer.Args(
26                  serverTransport);
27              ttpsArgs.processor(tprocessor);
28              ttpsArgs.protocolFactory(new TBinaryProtocol.Factory());
29
30              // 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。
31              TServer server = new TThreadPoolServer(ttpsArgs);
32              server.serve();
33
34          } catch (Exception e) {
35              System.out.println("Server start error!!!");
36              e.printStackTrace();
37          }
38      }
39
40
41  /**

```

```
42     * @param args
43     */
44     public static void main(String[] args) {
45         HelloServerDemo server = new HelloServerDemo();
46         server.startServer();
47     }
48
49 }
```

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：

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

## 5.NonblockingServer 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**

```
1 package com.micmiu.thrift.demo;
2
3 import org.apache.thrift.TProcessor;
4 import org.apache.thrift.protocol.TCompactProtocol;
5 import org.apache.thrift.server.TNonblockingServer;
6 import org.apache.thrift.server.TServer;
7 import org.apache.thrift.transport.TFramedTransport;
8 import org.apache.thrift.transport.TNonblockingServerSocket;
9
10 /**
11  * blog http://www.micmiu.com
12  *
13  * @author Michael
14  */
15
16 public class HelloServerDemo {
17     public static final int SERVER_PORT = 8090;
18
19     public void startServer() {
20         try {
21             System.out.println("HelloWorld TNonblockingServer start ...");
22
23             TProcessor tprocessor = new HelloWorldService.Processor<HelloWorldService>
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());
33
34             // 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
35             TServer server = new TNonblockingServer(tnbArgs);
36             server.serve();
37
38         } catch (Exception e) {
39             System.out.println("Server start error!!!");
40             e.printStackTrace();
41         }
42     }
43
44     /**
45     * @param args
46     */
47     public static void main(String[] args) {
48         HelloServerDemo server = new HelloServerDemo();
49         server.startServer();
50     }
51
52 }
```

编写客户端代码：**HelloClientDemo.java**

```
1 package com.micmiu.thrift.demo;
2
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;
9 import org.apache.thrift.transport.TTransportException;
10
11 /**
```

```

12  * blog http://www.micmiu.com
13  *
14  * @author Michael
15  *
16  */
17  public class HelloClientDemo {
18
19      public static final String SERVER_IP = "localhost";
20      public static final int SERVER_PORT = 8090;
21      public static final int TIMEOUT = 30000;
22
23      /**
24       *
25       * @param userName
26       */
27      public void startClient(String userName) {
28          TTransport transport = null;
29          try {
30              transport = new TFramedTransport(new TSocket(SERVER_IP,
31                  SERVER_PORT, TIMEOUT));
32              // 协议要和服务端一致
33              TProtocol protocol = new TCompactProtocol(transport);
34              HelloWorldService.Client client = new HelloWorldService.Client(
35                  protocol);
36              transport.open();
37              String result = client.sayHello(userName);
38              System.out.println("Thrify client result =: " + result);
39          } catch (TTransportException e) {
40              e.printStackTrace();
41          } catch (TException e) {
42              e.printStackTrace();
43          } finally {
44              if (null != transport) {
45                  transport.close();
46              }
47          }
48      }
49
50      /**
51       * @param args
52       */
53      public static void main(String[] args) {
54          HelloClientDemo client = new HelloClientDemo();
55          client.startClient("Michael");
56      }
57
58
59 }

```

客户端的测试成功，结果如下：

```

1 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**

```

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.protocol.TCompactProtocol;
6  import org.apache.thrift.server.THsHaServer;
7  import org.apache.thrift.server.TNonblockingServer;
8  import org.apache.thrift.server.TServer;
9  import org.apache.thrift.server.TSimpleServer;
10 import org.apache.thrift.server.TThreadPoolServer;
11 import org.apache.thrift.transport.TFramedTransport;
12 import org.apache.thrift.transport.TNonblockingServerSocket;
13 import org.apache.thrift.transport.TServerSocket;
14
15 /**
16  * blog http://www.micmiu.com
17  *
18  * @author Michael
19  *
20  */
21 public class HelloServerDemo {
22     public static final int SERVER_PORT = 8090;
23
24     public void startServer() {
25         try {
26             System.out.println("HelloWorld THsHaServer start ....");
27
28             TProcessor tprocessor = new HelloWorldService.Processor<>();

```

```

29         new HelloWorldImpl());
30
31         TNonblockingServerSocket tnbSocketTransport = new TNonblockingServerSocket(
32             SERVER_PORT);
33         THsHaServer.Args thhsArgs = new THsHaServer.Args(tnbSocketTransport);
34         thhsArgs.processor(tprocessor);
35         thhsArgs.transportFactory(new TFramedTransport.Factory());
36         thhsArgs.protocolFactory(new TBinaryProtocol.Factory());
37
38         //半同步半异步的服务模型
39         TServer server = new THsHaServer(thhsArgs);
40         server.serve();
41
42     } catch (Exception e) {
43         System.out.println("Server start error!!!");
44         e.printStackTrace();
45     }
46 }
47
48 /**
49  * @param args
50  */
51 public static void main(String[] args) {
52     HelloServerDemo server = new HelloServerDemo();
53     server.startServer();
54 }
55
56 }

```

客户端代码和上面 4 中的类似，只需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

```

1  package com.micmiu.thrift.demo;
2
3  import org.apache.thrift.TProcessor;
4  import org.apache.thrift.protocol.TCompactProtocol;
5  import org.apache.thrift.server.TNonblockingServer;
6  import org.apache.thrift.server.TServer;
7  import org.apache.thrift.transport.TFramedTransport;
8  import org.apache.thrift.transport.TNonblockingServerSocket;
9
10 /**
11  * blog http://www.micmiu.com
12  *
13  * @author Michael
14  */
15
16 public class HelloServerDemo {
17     public static final int SERVER_PORT = 8090;
18
19     public void startServer() {
20         try {
21             System.out.println("HelloWorld TNonblockingServer start ...");
22
23             TProcessor tprocessor = new HelloWorldService.Processor<>(new HelloWorldImpl());
24
25             TNonblockingServerSocket tnbSocketTransport = new TNonblockingServerSocket(
26                 SERVER_PORT);
27             TNonblockingServer.Args tnbArgs = new TNonblockingServer.Args(
28                 tnbSocketTransport);
29             tnbArgs.processor(tprocessor);
30             tnbArgs.transportFactory(new TFramedTransport.Factory());
31             tnbArgs.protocolFactory(new TCompactProtocol.Factory());
32
33             // 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
34             TServer server = new TNonblockingServer(tnbArgs);
35             server.serve();
36
37         } catch (Exception e) {
38             System.out.println("Server start error!!!");
39             e.printStackTrace();
40         }
41     }
42
43     /**
44      * @param args
45      */
46     public static void main(String[] args) {
47         HelloServerDemo server = new HelloServerDemo();
48         server.startServer();
49     }
50
51 }
52 }

```

编写客户端Client代码：HelloAsynClientDemo.java



```
1 package com.micmiu.thrift.demo;
2
3 import java.util.concurrent.CountDownLatch;
4 import java.util.concurrent.TimeUnit;
5
6 import org.apache.thrift.TException;
7 import org.apache.thrift.async.AsyncMethodCallback;
8 import org.apache.thrift.async.TAsyncClientManager;
9 import org.apache.thrift.protocol.TCompactProtocol;
10 import org.apache.thrift.protocol.TProtocolFactory;
11 import org.apache.thrift.transport.TNonblockingSocket;
12 import org.apache.thrift.transport.TNonblockingTransport;
13
14 import com.micmiu.thrift.demo.HelloWorldService.AsyncClient.sayHello_call;
15
16 /**
17  * blog http://www.micmiu.com
18  *
19  * @author Michael
20  *
21  */
22 public class HelloAsyncClientDemo {
23
24     public static final String SERVER_IP = "localhost";
25     public static final int SERVER_PORT = 8090;
26     public static final int TIMEOUT = 30000;
27
28     /**
29      *
30      * @param userName
31      */
32     public void startClient(String userName) {
33         try {
34             TAsyncClientManager clientManager = new TAsyncClientManager();
35             TNonblockingTransport transport = new TNonblockingSocket(SERVER_IP,
36                 SERVER_PORT, TIMEOUT);
37
38             TProtocolFactory tprotocol = new TCompactProtocol.Factory();
39             HelloWorldService.AsyncClient asyncClient = new HelloWorldService.Async
40                 tprotocol, clientManager, transport);
41             System.out.println("Client start .....");
42
43             CountDownLatch latch = new CountDownLatch(1);
44             AsyncCallback callBack = new AsyncCallback(latch);
45             System.out.println("call method sayHello start ...");
46             asyncClient.sayHello(userName, callBack);
47             System.out.println("call method sayHello .... end");
48             boolean wait = latch.await(30, TimeUnit.SECONDS);
49             System.out.println("latch.await =: " + wait);
50         } catch (Exception e) {
51             e.printStackTrace();
52         }
53         System.out.println("startClient end.");
54     }
55
56     public class AsyncCallback implements AsyncMethodCallback<sayHello_call> {
57         private CountDownLatch latch;
58
59         public AsyncCallback(CountDownLatch latch) {
60             this.latch = latch;
61         }
62
63         @Override
64         public void onComplete(sayHello_call response) {
65             System.out.println("onComplete");
66             try {
67                 // Thread.sleep(1000L * 1);
68                 System.out.println("AsyncCall result =:"
69                     + response.getResult().toString());
70             } catch (TException e) {
71                 e.printStackTrace();
72             } catch (Exception e) {
73                 e.printStackTrace();
74             } finally {
75                 latch.countDown();
76             }
77         }
78
79         @Override
80         public void onError(Exception exception) {
81             System.out.println("onError : " + exception.getMessage());
82             latch.countDown();
83         }
84     }
85
86     /**
87      * @param args
88      */
89     public static void main(String[] args) {
90         HelloAsyncClientDemo client = new HelloAsyncClientDemo();
```

```
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 AsyncCall result =:Hi,Michael welcome to my blog www.micmiu.com
6 latch.await =:true
7 startClient end.
```

原创文章，转载请注明：转载自micmiu – 软件开发+生活点滴[ <http://www.micmiu.com/> ]

本文链接地址: <http://www.micmiu.com/soa/rpc/thrift-sample/>

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...

**Michael** 2015 年 11 月 6 日 在 下午 4:57 回复

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eclipse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！

**Michael** 2014 年 12 月 9 日 在 下午 11:15

回复

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？

**Michael** 2014 年 11 月 24 日 在 下午 4:32

回复

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的封装解包和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）

**Michael** 2014 年 3 月 31 日 在 下午 11:00

回复

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？

**Michael** 2013 年 5 月 30 日 在 上午 10:07

回复

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)

**Michael** 2013 年 1 月 14 日 在 下午 10:53

回复

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());
33 TServer server = new THsHaServer(thhsArgs);
34 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！

**Michael** 2012 年 10 月 16 日 在 下午 6:09

回复

如果你想在一個port上起多個好像不行（我沒有深入研究過），實在不行可以把兩個service的接口合併成一個

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 😞

**Michael** 2012 年 9 月 13 日 在 下午 2:48

回复

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

**Michael** 2012 年 6 月 28 日 在 上午 9:51

回复

😊 以后可以多多交流

## 发表评论

<input type="text"/>	昵称
<input type="text"/>	邮箱
<input type="text"/>	网址



注意 - 你可以用以下 HTML tags and attributes:

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

