一．Server 端和 client 端编码基本步骤

1.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

2.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

二．代码

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)
}
```

运用 thrift-0.8.0.exe 生成相关代码：

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

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：

```
<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;

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;

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;HelloWorldService.Iface&gt;(
                                        new HelloWorldImpl());
                        // HelloWorldService.Processor&lt;HelloWorldService.Iface&gt;
```

```
tprocessor =
                    // 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;

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);
                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("LONG LI ZHANG");
            }
}
```

结果

服务端

HelloWorld TSimpleServer start ....

客户端

Thrify client result =: Hi LONG LI ZHANG, welcome to my blog

www.micmiu.com

```
44
45              // 服务端开启服务s
46              server.serve();
47
48          } catch (TTransportException e) {
49              e.printStackTrace();
50          }
```

Console | Problems | Progress | Debug Shell | Servers

HelloServiceServer [Java Application] C:\Program Files\Java\jdk1.7.0_80\jre\bin\javaw.exe (2020年10月12日 下午11:37:12)

```
Hello World TSimpleServer start
```

```
41              // 关闭
```

Console | Problems | Progress | Debug Shell | Servers

<terminated> HelloClient [Java Application] C:\Program Files\Java\jdk1.7.0_80\jre\bin\javaw.exe (2020年10月12日 下午11:34:36)

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

总结报告

1. Thrift 类介绍

Thrift 代码包（位于 thrift-0.6.1/lib/cpp/src）有以下几个目录：

concurrency：并发和时钟管理方面的库

processor：Processor 相关类

protocal：Protocal 相关类

transport：transport 相关类

server：server 相关类

1.1 Transport 类

负责数据传输，有以下几个可用类：

TFileTransport：文件（日志）传输类，允许 client 将文件传给 server，允许 server 将收到的数据写到文件中；

THttpTransport：采用 Http 传输协议进行数据传输；

TSocket：采用 TCP Socket 进行数据传输；

TZlibTransport：压缩后对数据进行传输，或者将收到的数据解压。

下面几个类主要是对上面几个类地装饰（采用了装饰模式），以提高传输效率。

TBufferedTransport：对某个 Transport 对象操作的数据进行 buffer，即从 buffer 中读取数据进行传输，或者将数据直接写入 buffer；

TFramedTransport：同 TBufferedTransport 类似，也会对相关数据进行 buffer，同时，它支持定长数据发送和接收；

TMemoryBuffer：从一个缓冲区中读写数据。

1.2 Protocol 类

负责数据编码，主要有以下几个可用类：

TBinaryProtocol：二进制编码

TJSONProtocol：JSON 编码

TCompactProtocol：密集二进制编码

TDebugProtocol：以用户易读的方式组织数据

1.3 Server 类

TSimpleServer：简单的单线程服务器，主要用于测试

TThreadPoolServer：使用标准阻塞式 IO 的多线程服务器

TNonblockingServer：使用非阻塞式 IO 的多线程服务器，TFramedTransport 必须使用该类型的 server

1.5 对象序列化和反序列化

Thrift 中的 Protocol 负责对数据进行编码，因而可使用 Protocol 相关对象进行序列化和反序列化。

2.编写 client 和 server

2.1 client 端代码编写

Client 编写的方法分为以下几个步骤：

（1） 定义 TTransport，为你的 client 设置传输方式（如 socket， http 等）。

（2） 定义 Protocal，使用装饰模式（Decorator 设计模式）封装 TTransport，为你的数据设置编码格式（如二进制格式，JSON 格式等）

（3） 实例化 client 对象，调用服务接口。

说明：如果用户在 thrift 文件中定义了一个叫${server_name}的 service，则会生成一个叫${server_name}Client 的对象。

2.2 Server 端代码编写

（1） 定义一个 TProcess，这个是 thrift 根据用户定义的 thrift 文件自动生成的类

（2） 使用 TServerTransport 获得一个 TTransport

（3） 使用 TTransportFactory，可选地将原始传输转换为一个适合的应用传输（典型的是使用 TBufferedTransportFactory）

（4） 使用 TProtocolFactory，为 TTransport 创建一个输入和输出

（5） 创建 TServer 对象（单线程，可以使用 TSimpleServer；对于多线程，用户可使用 TThreadPoolServer 或者 TNonblockingServer），调用它的 server()函数。

说明：thrift 会为每一个带 service 的 thrift 文件生成一个简单的 server 代码（桩），