[thrift－异步client&server使用例子](http://blog.csdn.net/whycold/article/details/11019967)

2013-09-03 21:42 6503人阅读 [评论](http://blog.csdn.net/whycold/article/details/11019967#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/whycold/article/details/11019967#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

thrift源码研究（15） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

         研究完异步client&server源码后，当然需要写个[**测试**](http://lib.csdn.net/base/softwaretest)例子来验证一下！

        同样采用test.thrift作为例子

/\* @file : test.thrift \*/

namespace cpp thrift.example

service Twitter {  
   string sendString(1:string data);  
}

     输入如下命令，自动生成实现异步client&server所需的类：

    thrift -r -strict  --gen cpp:cob\_style -o ./ test.thrift

    用户只要关心TwitterCobClient、TwitterCobSvIf和TwitterAsyncProcessor这三个类。

一、实现异步client

/\*@file: testclient.cpp\*/

#include <stdio.h>  
#include <getopt.h>  
#include <event.h>  
#include <string>  
#include <iostream>  
#include <boost/bind.hpp>  
#include <boost/function.hpp>  
#include <protocol/TBinaryProtocol.h>  
#include <transport/TSocket.h>  
#include <transport/TTransportUtils.h>  
#include <async/TEvhttpClientChannel.h>  
#include "Twitter.h"

using namespace std;  
using namespace ::apache::thrift;  
using namespace ::apache::thrift::protocol;  
using namespace ::apache::thrift::transport;  
using namespace ::apache::thrift::async;  
using namespace ::boost;  
using namespace ::thrift::example;  
using ::boost::bind;

class testClient : public TwitterCobClient  
{  
public:  
  testClient(boost::shared\_ptr< ::apache::thrift::async::TAsyncChannel> channel, TProtocolFactory\* protocolFactory)  
      : TwitterCobClient(channel, protocolFactory)  
  { };

  virtual void completed\_\_(bool success)  
  {  
    if (success)  
    {  
                printf("respone : %s \n", res.c\_str());   // 输出返回结果  
    }  
    else  
    {  
      printf("failed to respone\n");  
    }  
    fflush(0);  
  };

  string res;  
};

// callback function

static void my\_recv\_sendString(TwitterCobClient\* client)  
{  
    client->recv\_sendString(dynamic\_cast<testClient \*>(client)->res);  
};

static void sendString(  
         testClient& client  
         )     
{  
  printf("sendString start\n");  
  function<void(TwitterCobClient\* client)> cob = bind(&my\_recv\_sendString, \_1);  
  client.sendString(cob, "hello");   // 发送并注册回调函数  
  printf("sendString end\n");  
}

static void DoSimpleTest(  
  const std::string& host, int port  
  )  
{  
  printf( "running DoSimpleTest( %s, %d) ...\n",  
    host.c\_str(), port);

  event\_base\* base = event\_base\_new();

  boost::shared\_ptr< ::apache::thrift::async::TAsyncChannel>  channel1( new TEvhttpClientChannel( host, "/", host.c\_str(), port, base  ) );  
    
  testClient client1( channel1,  new TBinaryProtocolFactory() );

  sendString(client1);   // 发送第一个请求

  boost::shared\_ptr< ::apache::thrift::async::TAsyncChannel>  channel2( new TEvhttpClientChannel( host, "/", host.c\_str(), port, base  ) );

  testClient client2( channel2,  new TBinaryProtocolFactory() );

  sendString(client2);  // 发送第二个请求

  event\_base\_dispatch(base);

  event\_base\_free(base);

  printf( "done DoSimpleTest().\n" );  
}

int main( int argc, char\* argv[] )  
{  
  DoSimpleTest( "172.19.101.61", 14488 );  
  return 0;

}

   编译生成async\_client\_d。

二、异步server实现

/\*@file: testserver.cpp\*/

#include "Twitter.h"  
#include <protocol/TBinaryProtocol.h>  
#include <async/TEvhttpServer.h>  
#include <stdio.h>  
#include <getopt.h>  
#include <event.h>  
#include <string>  
#include <iostream>  
#include <boost/bind.hpp>  
#include <boost/function.hpp>  
#include <transport/TSocket.h>  
#include <transport/TTransportUtils.h>  
#include <async/TAsyncProtocolProcessor.h>

using namespace ::apache::thrift;  
using namespace ::apache::thrift::protocol;  
using namespace ::apache::thrift::transport;  
using namespace ::apache::thrift::async;

using boost::shared\_ptr;

using namespace thrift::example;

class TwitterAsyncHandler : public TwitterCobSvIf {  
 public:  
  TwitterAsyncHandler() {  
    // Your initialization goes here  
  }

  void sendString(std::tr1::function<void(std::string const& \_return)> cob, const std::string& data) {  
    printf("sendString rec:%s\n", data.c\_str());  // 输出收到的数据  
    std::string \_return = "world";   // 返回world字符串给客户端  
    return cob(\_return);  
  }

};

int main(int argc, char \*\*argv) {  
  shared\_ptr<TAsyncProcessor> underlying\_pro(new TwitterAsyncProcessor( shared\_ptr<TwitterCobSvIf>(new TwitterAsyncHandler()) ) );  
  shared\_ptr<TAsyncBufferProcessor> processor( new TAsyncProtocolProcessor( underlying\_pro, shared\_ptr<TProtocolFactory>(new TBinaryProtocolFactory()) ) );

  TEvhttpServer server(processor, 14488);  
  server.serve();  
  return 0;  
}

       编绎生成async\_server\_d。

三、运行async\_client\_d和async\_server\_d

（1）先运行server:

$./async\_server\_d

（2）然后运行async\_client\_d

$./async\_client\_d

 结果输出：

（1）server输出：

sendString rec:hello  
sendString rec:hello

收到两个请求，且内容为hello

（2）client输出：

running DoSimpleTest( 172.19.101.61, 14488) ...  
sendString start  
sendString end  
sendString start  
sendString end  
respone : world   
respone : world   
done DoSimpleTest()

发出两个请求，且回复都是world。

OK，发送数据和接收数据都正确。