thrift 在 go 语言环境的使用测试

作者：450370050
原文链接：https://ld246.com/article/1524558127251
来源网站：链滴

Thrift是通用型rpc框架，但是功能比较单一，只是实现了点对点(End-to-End)的通讯框架，缺乏服务理的功能，比如服务注册和发现、 负载均衡、容灾、服务监控等功能。

Idl介绍

传输格式

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。

TDebugProtocol – 使用易懂的可读的文本格式，以便于debug

传输方式

TSocket -阻塞式socker

TFramedTransport – 以frame为单位进行传输，非阻塞式服务中使用。

TFileTransport – 以文件形式进行传输。

TMemoryTransport – 将内存用于I/O. java实现时内部实际使用了简单的ByteArrayOutputStream。

TZlibTransport – 使用zlib进行压缩， 与其他传输方式联合使用。当前无java实现。

服务模型

TSimpleServer – 简单的单线程服务模型，常用于测试

TThreadPoolServer – 多线程服务模型，使用标准的阻塞式IO。

TNonblockingServer – 多线程服务模型，使用非阻塞式IO（需使用TFramedTransport数据传输方）

基本类型

bool: 布尔值

byte: 有符号字节

i16: 16位有符号整型 int16

i32: 32位有符号整型 int32

i64: 64位有符号整型 int64

double: 64位浮点型

string: 字符串

binary: Blob 类型　byte数组

list<t>: 数组

set<t>: 集合

map<t, t>: 字典

结构体

1.struct不能继承，但是可以嵌套，不能嵌套自己。

2.其成员都是有明确类型

3.成员是被正整数编号过的，其中的编号使不能重复的，这个是为了在传输过程中编码使用。

4.成员分割符可以是逗号（,）或是分号（;），而且可以混用

5.字段会有optional和required之分和protobuf一样，但是如果不指定则为无类型–可以不填充该值但是

在序列化传输的时候也会序列化进去，optional是不填充则部序列化，required是必须填充也必须序化。

6.每个字段可以设置默认值

7.同一文件可以定义多个struct，也可以定义在不同的文件，进行include引入。

```
struct User{
  1: required string name, //改字段必须填写
  2: optional i32 age = 0; //默认值
  3: bool gender //默认字段类型为optional
}
```

常量定义

使用方法：在变量前面加上const

```
const i32 const_int = 1;
```

类型定义

Thrift支持C/C++类型定义

typedef i32 myint

末尾没有逗号

Exception (异常)

异常在语法和功能上类似于结构体，差别是异常使用关键字exception，而且异常是继承每种语言的

础异常类。

```
exception MyException {
    1: i32 errorCode,
    2: string message
}
```

Service (服务定义类型)

服务的定义方法在语义上等同于面向对象语言中的接口。

```
service HelloService {
    i32 sayInt(1:i32 param)
    string sayString(1:string param)
    bool sayBoolean(1:bool param)
    void sayVoid()
}
```

NameSpace

Thrift中的命名空间类似于C++中的namespace和java中的package，它们提供了一种组织（隔离）码的简便方式。名字空间也可以用于解决类型定义中的名字冲突。

Include

便于管理、重用和提高模块性/组织性，我们常常分割Thrift定义在不同的文件中。包含文件搜索方式c++一样。Thrift允许文件包含其它thrift文件，用户需要使用thrift文件名作为前缀访问被包含的对象

测试

1、安装下载

下载thrift代码编译生成工具，可直接生成脚手架代码。

http://thrift.apache.org/download

github下载框架代码，一定要根据代码生成工具的版本选择对应的代码版本，否则生成的代码会有异常

https://github.com/apache/thrift

2、IDL编写

```
vi   my_article_service.thrift
namespace go my_article_service
struct Article{
    1: i32      id,
    2: string   title,
    3: string   content,
}

service articleService{
```

```
    list<Article> ArticleList(),
    Article     ArticleInfo(1:i32 id),
}
```

在gen-go文件夹下生成脚手架代码

```
thrift -r --gen go my_article_service.thrif
```

3、编写server

业务处理handler

```
vi handler/article_handler.go
```

```
package handler


import (
   "thirft_test/self/gen-go/my_article_service"
)

type  ArticleService   struct{
}
func(ah ArticleService) ArticleList() ([]*my_article_service.Article, error) {
   var al []*my_article_service.Article
   ai := new(my_article_service.Article)
   ai.ID = 123
   ai.Title = "dasdas"
   ai.Content = "dsadasdsadsdsa"
   al = append(al, ai)
   return al,nil
}

func(ah ArticleService) ArticleInfo(id int32) (*my_article_service.Article, error) {
   ai := new(my_article_service.Article)
   ai.ID = id
   ai.Title = "dasdas"
   ai.Content = "dsadasdsadsdsa"
   return ai,nil
}
```

```
vi man.go
```

```
package main


import (
   "git.apache.org/thrift.git/lib/go/thrift"
   "thirft_test/self/gen-go/my_article_service"
   "thirft_test/self/handler"
)

func main() {
   errc := make(chan error)
```

```
    go func() {
        var as my_article_service.ArticleService
        as = handler.ArticleService{}

        processor := my_article_service.NewArticleServiceProcessor(as)
        transportFactory := thrift.NewTFramedTransportFactory(thrift.NewTTransportFactory())
        protocolFactory := thrift.NewTBinaryProtocolFactoryDefault()
        serverTransport, _ := thrift.NewTServerSocket("127.0.0.1:9090")
        server := thrift.NewTSimpleServer4(processor, serverTransport, transportFactory, protocol
actory)
        errc<- server.Serve()
    }()
    <-errc
}
```

4、编写client

vi client/main.go

```
package main


import (
    "git.apache.org/thrift.git/lib/go/thrift"
    "thirft_test/self/gen-go/my_article_service"
    "fmt"
)

func main() {
    transportFactory := thrift.NewTFramedTransportFactory(thrift.NewTTransportFactory())
    protocoleFactory := thrift.NewTBinaryProtocolFactoryDefault()
    socket,err := thrift.NewTSocket("127.0.0.1:9090")
    if err != nil {

    }
    transport := transportFactory.GetTransport(socket)
    client := my_article_service.NewArticleServiceClientFactory(transport, protocoleFactory)
    transport.Open()
    article,_ := client.ArticleInfo(123)
    fmt.Println(article)
    al,_ := client.ArticleList()
    fmt.Println(al)
    transport.Close()
}
```

5、运行

```
Article({ID:123 Title:dasdas Content:dsadasdsadsdsa})
[Article({ID:123 Title:dasdas Content:dsadasdsadsdsa})]
```