## Thrift的技术栈

Thrift对**软件栈**的定义非常的清晰, 使得各个组件能够**松散的耦合**, 针对不同的应用场景, 选择不同是方式去搭建服务。

Thrift**软件栈**分层**从下向上**分别为：**传输层**(Transport Layer)、**协议层**(Protocol Layer)、**处理层**(Processor Layer)和**服务层**(Server Layer)。

**传输层**(Transport Layer)：传输层负责直接从网络中**读取**和**写入**数据，它定义了具体的**网络传输协议**；比如说TCP/IP传输等。

**协议层**(Protocol Layer)：协议层定义了**数据传输格式**，负责网络传输数据的**序列化**和**反序列化**；比如说JSON、XML、**二进制数据**等。

**处理层**(Processor Layer)：处理层是由具体的IDL（**接口描述语言**）生成的，封装了具体的**底层网络传输**和**序列化方式**，并委托给用户实现的Handler进行处理。

**服务层**(Server Layer)：整合上述组件，提供具体的**网络线程/IO服务模型**，形成最终的服务。

## Thrift的特性

### (一) 开发速度快

通过编写RPC接口Thrift IDL文件，利用**编译生成器**自动生成**服务端骨架**(Skeletons)和**客户端桩**(Stubs)。从而省去开发者**自定义**和**维护接口编解码**、**消息传输**、**服务器多线程模型**等基础工作。

* 服务端：只需要按照**服务骨架**即**接口**，编写好具体的**业务处理程序**(Handler)即**实现类**即可。
* 客户端：只需要拷贝IDL定义好的**客户端桩**和**服务对象**，然后就像调用本地对象的方法一样调用远端服务。

### (二) 接口维护简单

通过维护Thrift格式的IDL（**接口描述语言**）文件（注意写好注释），即可作为给Client使用的接口文档使用，也**自动生成**接口代码，始终保持代码和文档的一致性。且Thrift协议可灵活支持**接口**的**可扩展性**。

### (三) 学习成本低

因为其来自Google Protobuf开发团队，所以其IDL文件风格类似Google Protobuf，且更加**易读易懂**；特别是RPC**服务接口**的风格就像写一个**面向对象**的Class一样简单。

初学者只需参照：[http://thrift.apache.org/](https://link.jianshu.com?t=http://thrift.apache.org/" \t "_blank)，一个多小时就可以理解Thrift IDL文件的语法使用。

### (四) 多语言/跨语言支持

Thrift支持C++、 Java、Python、PHP、Ruby、Erlang、Perl、Haskell、C#、Cocoa、JavaScript、Node.js、Smalltalk等多种语言，即可生成上述语言的**服务器端**和**客户端程序**。

对于我们经常使用的Java、PHP、Python、C++支持良好，虽然对iOS环境的Objective-C(Cocoa)支持稍逊，但也完全满足我们的使用要求。

### (五) 稳定/广泛使用

Thrift在很多开源项目中已经被验证是**稳定**和**高效**的，例如Cassandra、Hadoop、HBase等；国外在Facebook中有广泛使用，国内包括百度、美团小米、和饿了么等公司。

## Thrift的数据类型

Thrift 脚本可定义的数据类型包括以下几种类型：

1. **基本类型：**  
   　　**bool**: 布尔值  
   　　**byte**: 8位有符号整数  
   　　**i16**: 16位有符号整数  
   　　**i32**: 32位有符号整数  
   　　**i64**: 64位有符号整数  
   　　**double**: 64位浮点数  
   　　**string**: UTF-8编码的字符串  
   　　**binary**: 二进制串
2. **结构体类型：**  
   　　**struct**: 定义的结构体对象
3. **容器类型：**  
   　　**list**: 有序元素列表  
   　　**set**: 无序无重复元素集合  
   　　**map**: 有序的key/value集合
4. **异常类型：**  
   　　**exception**: 异常类型
5. **服务类型：**  
   　　**service**: 具体对应服务的类

## Thrift的协议

Thrift可以让用户选择**客户端**与**服务端**之间**传输通信协议**的类别，在**传输协议**上总体划分为**文本**(text)和**二进制**(binary)传输协议。为**节约带宽**，**提高传输效率**，一般情况下使用**二进制**类型的传输协议为多数，有时还会使用基于**文本类型**的协议，这需要根据项目/产品中的实际需求。常用协议有以下几种：

* TBinaryProtocol：**二进制**编码格式进行数据传输
* TCompactProtocol：**高效率**的、**密集**的**二进制**编码格式进行数据传输
* TJSONProtocol： 使用JSON**文本**的数据编码协议进行数据传输
* TSimpleJSONProtocol：只提供JSON**只写**的协议，适用于通过**脚本语言解析**

## Thrift的传输层

常用的传输层有以下几种：

* TSocket：使用**阻塞式**I/O进行传输，是最常见的模式
* TNonblockingTransport：使用**非阻塞方式**，用于构建**异步客户端**
* TFramedTransport：使用**非阻塞方式**，按**块的大小**进行传输，类似于Java中的NIO

## Thrift的服务端类型

* TSimpleServer：**单线程**服务器端，使用标准的**阻塞式**I/O
* TThreadPoolServer：**多线程**服务器端，使用标准的**阻塞式**I/O
* TNonblockingServer：**单线程**服务器端，使用**非阻塞式**I/O
* THsHaServer：**半同步半异步**服务器端，基于**非阻塞式**IO读写和**多线程**工作任务处理
* TThreadedSelectorServer：**多线程选择器**服务器端，对THsHaServer在**异步**IO模型上进行增强