Apache Thrift

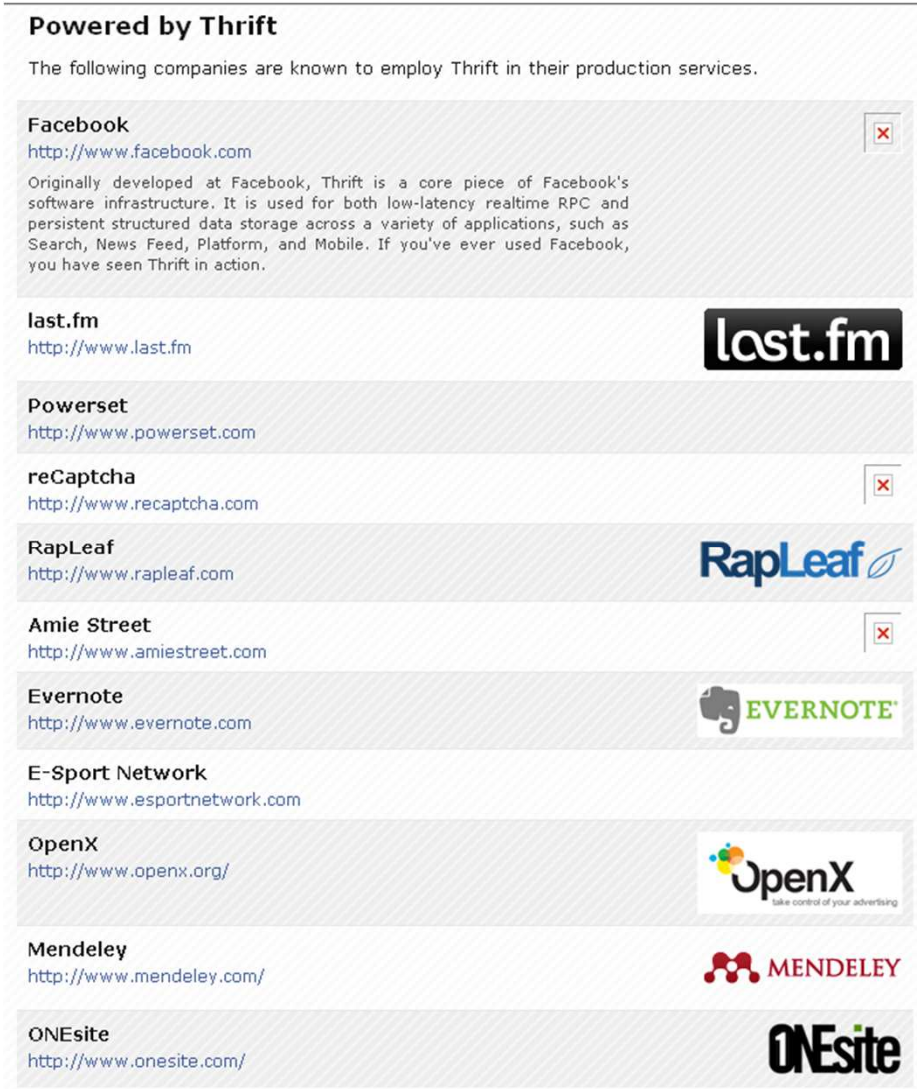
김용환
knight76.tistory.com

서론

Thrift

- a software framework for scalable cross language service development. It combines software stack with a code generation engine to build services that work efficiently and seamlessly
- Provides cross-language RPC and serialization library
- 지원 언어 :
  C++ ,C# ,Cocoa ,Erlang ,Haskell ,Java ,Ocaml ,Perl ,PHP ,Python ,Ruby ,Smalltalk
- Champion
  - Doug Cutting (하둡)
- Apache license

Powered by Thrift



Powered by Thrift

The following companies are known to employ Thrift in their production services.

Facebook
http://www.facebook.com

Originally developed at Facebook, Thrift is a core piece of Facebook's software infrastructure. It is used for both low-latency realtime RPC and persistent structured data storage across a variety of applications, such as Search, News Feed, Platform, and Mobile. If you've ever used Facebook, you have seen Thrift in action.

last.fm
http://www.last.fm

Powerset
http://www.powerset.com

reCaptcha
http://www.recaptcha.com

RapLeaf
http://www.rapleaf.com

Amie Street
http://www.amiestreet.com

Evernote
http://www.evernote.com

E-Sport Network
http://www.esportnetwork.com

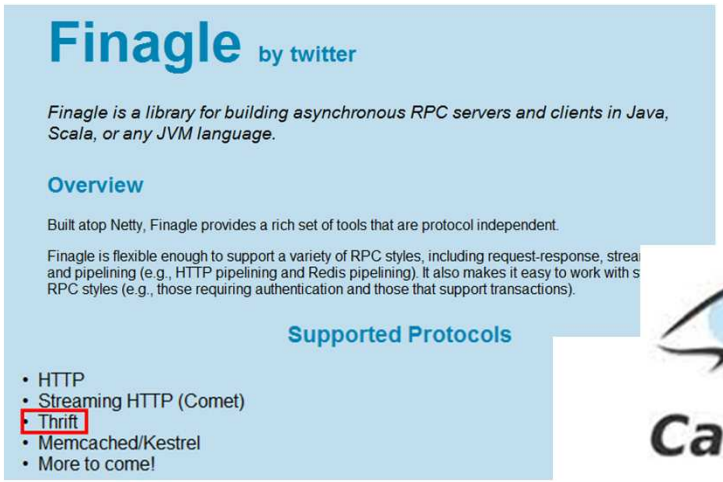
OpenX
http://www.openx.org/

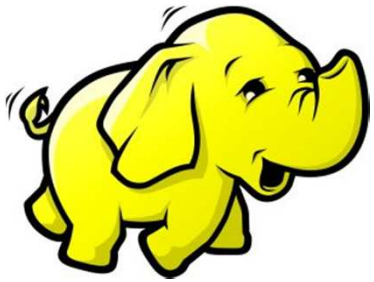
Mendeley
http://www.mendeley.com/

ONEsite
http://www.onesite.com/

Support Protocol

- 많은 곳에서 통신 규약으로 사용하는 중





장점

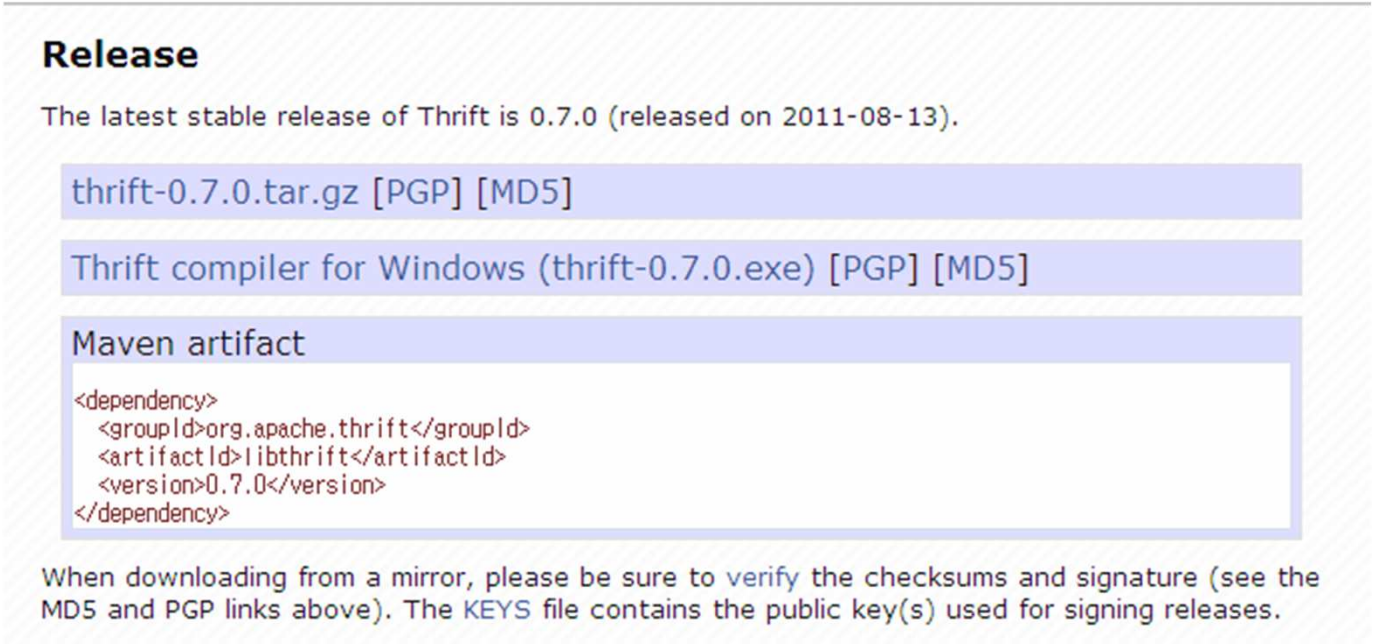
- 버전닝 지원
- 여러 언어에서 사용 가능하며 언어에 맞도록 소스가 생성되고, 언어 간의 Serialization 가능
- Sync, Async API 제공
- XML 설정이 필요 없음
- Layer에 맞는 Interface를 사용 및 Customizing 구축 가능
- RPC 기능 제공 (Google Protocol Buffer에는 없는 기능)
  - 서버 기능 좋음
    - 서블릿 제공(org.apache.thrift.server.TServlet)
    - 멀티쓰레드 지원 (org.apache.thrift.server.ThreadPoolServer : worker thread 지정)
    - Async 지원 (org.apache.thrift.server. TNonblockingServer : single threaded)
    - Multi-thread Half-Sync/Half-Async지원 : org.apache.thrift.server. THsHaServer
- Exception을 제공 (Google Protocol Buffer에는 없는 기능)
- Set, Map 지원 (Google Protocol Buffer에는 없는 기능)

단점 ?

- 자바로 코드가 생성될 때, Slf4j를 기본적으로 가지고 가며, 내부 코드는 모두 thrift lib가 필요함
- C++의 Server threading part는 Boost에 의존을 가짐
- 자바 쪽 이클립스 플러그인 없음
- Thrift lib의 api가 자주 바뀌어서, 버전 업마다 소스를 좀 보고 코딩해야 함. (인터넷 예제가 거의 없음)
- XML Serialization/Deserialization 기능 없음
- 문서가 확실히 적음
- 생성된 코드가 Google Protocol Buffer에 비해서 보기는 편하지는 않음 (특히 C++)

download

- http://thrift.apache.org/download/



Release

The latest stable release of Thrift is 0.7.0 (released on 2011-08-13).

thrift-0.7.0.tar.gz [PGP] [MD5]

Thrift compiler for Windows (thrift-0.7.0.exe) [PGP] [MD5]

Maven artifact

```
<dependency>
  <groupId>org.apache.thrift</groupId>
  <artifactId>libthrift</artifactId>
  <version>0.7.0</version>
</dependency>
```

When downloading from a mirror, please be sure to verify the checksums and signature (see the MD5 and PGP links above). The KEYS file contains the public key(s) used for signing releases.

설치 및 사용

- 원래는 configure/make를 필요 (리눅스/윈도우)
- 윈도우는 그냥 실행하면 됨 (여기서는 윈도우 버전으로만 테스트)

```
C:₩>thrift-0.7.0.exe
Usage: thrift [options] file
Options:
  -version    Print the compiler version
  -o dir      Set the output directory for gen-* packages
               (default: current directory)
  -out dir    Set the ouput location for generated files.
               (no gen-* folder will be created)
  -I dir      Add a directory to the list of directories
                searched for include directives
  -nowarn     Suppress all compiler warnings (BAD!)
  -strict     Strict compiler warnings on
  -v[erbose]  Verbose mode
  -r[ecurse]  Also generate included files
  -debug      Parse debug trace to stdout
  --gen STR   Generate code with a dynamically-registered generator.
                STR has the form language[:key1=val1[,key2,[key3=val3]]].
                Keys and values are options passed to the generator.
                Many options will not require values.

Available generators (and options):
  as3 (AS3):
    bindable:          Add [bindable] metadata to all the struct classes.
  c_glib (C, using GLib):
  cocoa (Cocoa):
    log_unexpected:  Log every time an unexpected field ID or type is encountere
d.
  cpp (C++):
    cob_style:       Generate "Continuation OBject"-style classes.
    no_client_completion:
                     Omit calls to completion__() in CobClient class.
    templates:       Generate templatized reader/writer methods.
    pure_enums:      Generate pure enums instead of wrapper classes.
    dense:           Generate type specifications for the dense protocol.
    include_prefix:  Use full include paths in generated files.
```

(소스 컴파일시)
Basic requirements

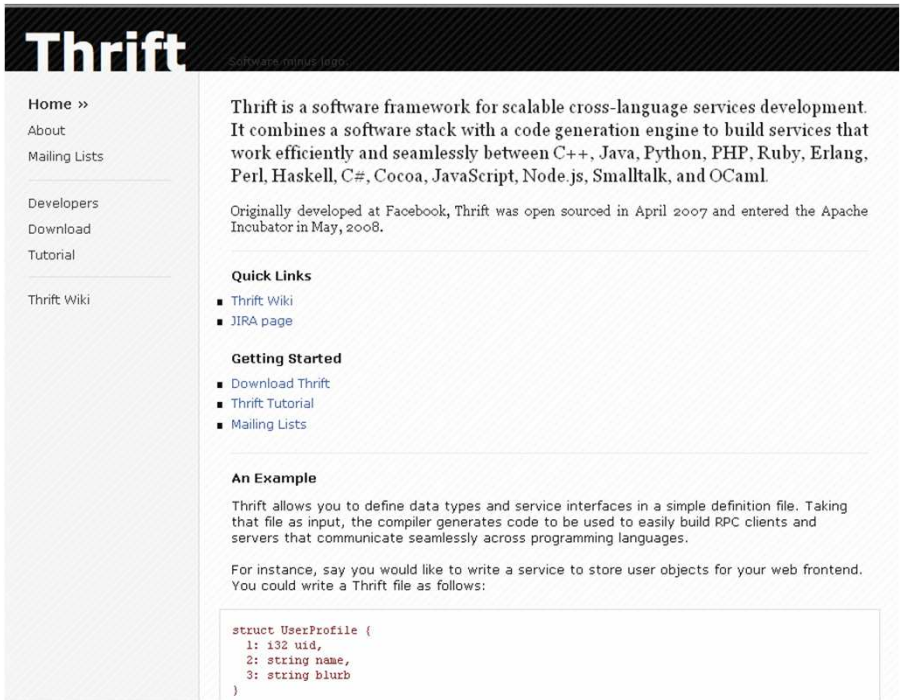
- A relatively POSIX-compliant *NIX system
  - Cygwin or MinGW can be used on Windows
- g++ 3.3.5+
- boost 1.33.1+ (1.34.0 for building all tests)
- Runtime libraries for lex and yacc might be needed for the compiler.

(소스 컴파일시)
Language requirements

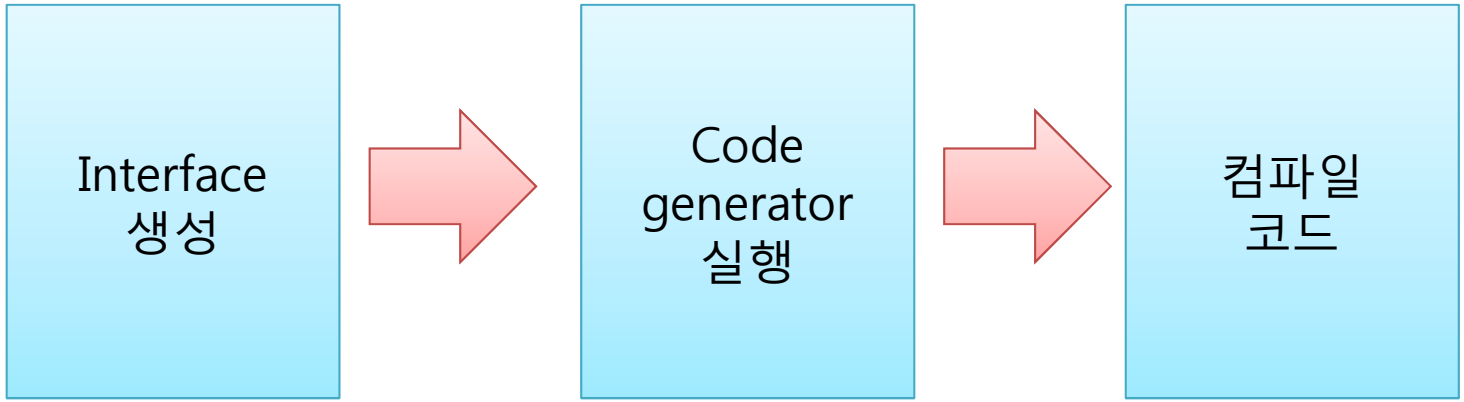
- C++
  - Boost 1.33.1+
  - libevent (optional, to build the nonblocking server)
  - zlib (optional)
- Java
  - Java 1.5+
  - Apache Ant
  - Apache Ivy (recommended)
  - Apache Commons Lang (recommended)
  - SLF4J
- C#: Mono 1.2.4+ (and pkg-config to detect it) or Visual Studio 2005+
- Python 2.4+ (including header files for extension modules)
- PHP 5.0+ (optionally including header files for extension modules)
- Ruby 1.8+ (including header files for extension modules)
- Erlang R12 (R11 works but not recommended)
- Perl 5
  - Bit::Vector
  - Class::Accessor

History

- Originally developed by Facebook
- Donated to apache
  apache incubator
- Now 상위 프로젝트
  http://thrift.apache.org/



Interface definition



.thrift 파일 작성

실행
thrift -gen java -gen py foo.thrift

예제

샘플 #1

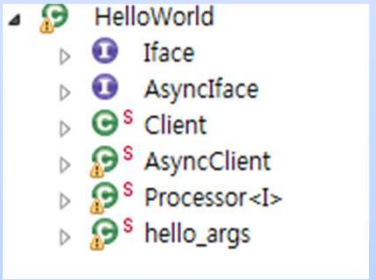
Thrift파일

```
service HelloWorld {
                oneway void hello(1:string name)
}
```

```
C:₩thrift>thrift-0.7.0.exe --gen java HelloWorld.thrift
```

Generated Code

```
public class HelloWorld {
   public interface Iface {
     public void hello(String name) throws org.apache.thrift.TException;
    }

  public interface AsyncIface {
    public void hello(String name, org.apache.thrift.async.AsyncMethodCallback<AsyncClient.hello_call> resultHandler) throws
org.apache.thrift.TException;
   }

  public static class Client extends org.apache.thrift.TServiceClient implements Iface {
   ...
   }

   public static class AsyncClient extends org.apache.thrift.async.TAsyncClient implements AsyncIface {
   ...
   }

  public static class Processor<I extends Iface> extends org.apache.thrift.TBaseProcessor implements org.apache.thrift.TProcessor    {
    }

  public static class hello_args implements org.apache.thrift.TBase<hello_args, hello_args._Fields>, java.io.Serializable, Cloneable    {
   ...
   }
```



Server Code : 사용자는 generated code를 상속하고.Thrift API를 이용해서 포트매핑

```
public class HelloWorldServer {

	public static void main(String[] argss) {
		try {
			Factory portFactory = new TBinaryProtocol.Factory(true, true);
			TServerSocket serverTransport = new TServerSocket(8000);
			HelloWorld.Processor<HelloWorld.Iface> processor =
				new HelloWorld.Processor<HelloWorld.Iface>(new
HelloWorldImpl());
			Args args = new Args(serverTransport);
			args.processor(processor);
			args.protocolFactory(portFactory);
			TServer server = new TThreadPoolServer(args);
			System.out.println("Starting server on port 8000 ...");
			server.serve();
		} catch (TTransportException e) {
			e.printStackTrace();
		}
	}
}

class HelloWorldImpl implements HelloWorld.Iface {
	@Override
	public void hello(String name) throws TException {
		long time = System.currentTimeMillis();
		System.out.println("Current time : " + time);
		System.out.println("Hello " + name);
	}
}
```

Client Code

```
public class HelloWorldClient {
        public static void main(String[] args) {
                TTransport transport;
                try {
                        transport = new TSocket("localhost", 8000);
                        TProtocol protocol = new TBinaryProtocol(transport);
                        HelloWorld.Client client = new HelloWorld.Client(protocol);
                        transport.open();
                        client.hello("World!!!");
                        transport.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

실행결과

```
Starting server on port 8000 ...
Current time : 1317014309989
Hello World!!!
```

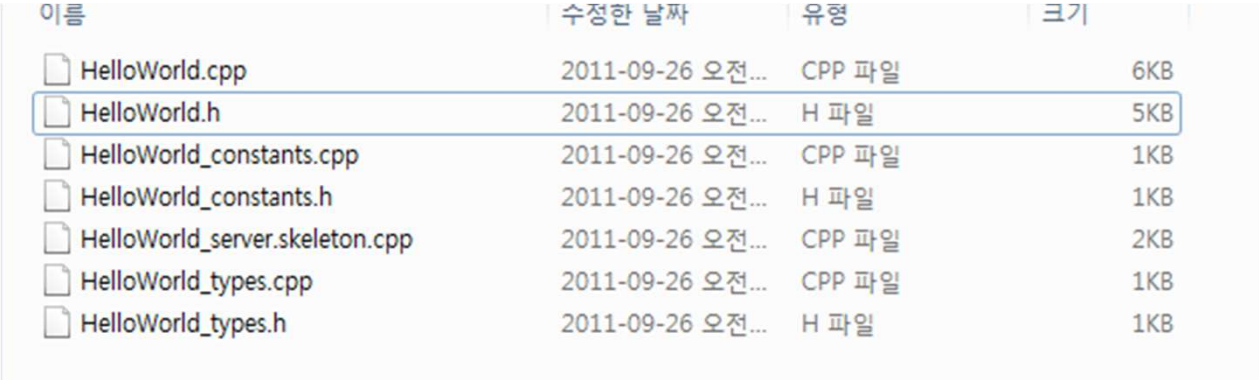
Client/Server 의 pom.xml

- Slf4j와 thrift library가 있어야 컴파일 됨

```
<dependencies>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.6.2</version>
</dependency>

<dependency>
    <groupId>org.apache.thrift</groupId>
    <artifactId>libthrift</artifactId>
    <version>0.7.0</version>
</dependency>

</dependencies>
```

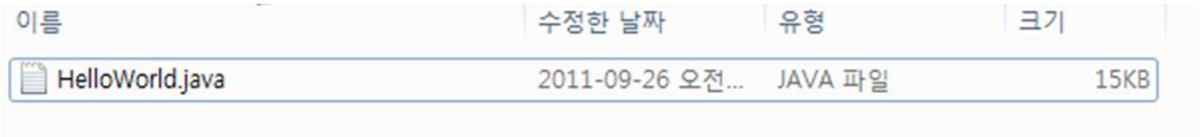
생성된 소스

- C++



이름	수정한 날짜	유형	크기
HelloWorld.cpp	2011-09-26 오전...	CPP 파일	6KB
HelloWorld.h	2011-09-26 오전...	H 파일	5KB
HelloWorld_constants.cpp	2011-09-26 오전...	CPP 파일	1KB
HelloWorld_constants.h	2011-09-26 오전...	H 파일	1KB
HelloWorld_server.skeleton.cpp	2011-09-26 오전...	CPP 파일	2KB
HelloWorld_types.cpp	2011-09-26 오전...	CPP 파일	1KB
HelloWorld_types.h	2011-09-26 오전...	H 파일	1KB

- Java



이름	수정한 날짜	유형	크기
HelloWorld.java	2011-09-26 오전...	JAVA 파일	15KB

샘플 #2

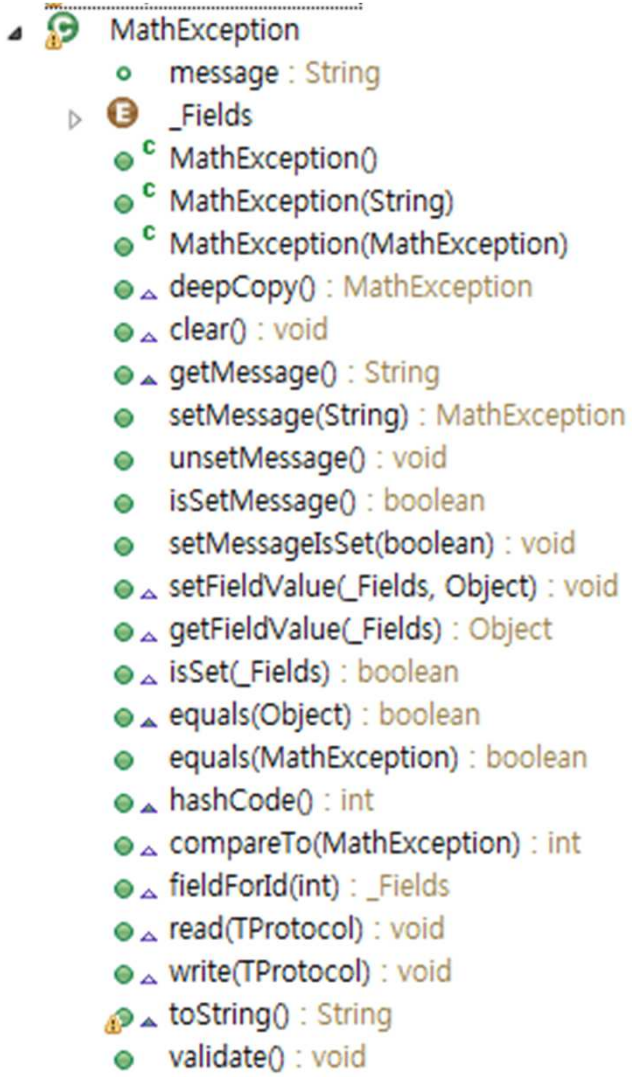
```
exception MathException {
   1 : string message
}

service CalculatorServicef {
   i32 add (1 : i32  a , 2 : i32 b )
   i32 subtract (1 : i32 a , 2 : i32 b )
   double divide (1 : double a , 2 : double b ) throws (1 : MathException me)
}
```

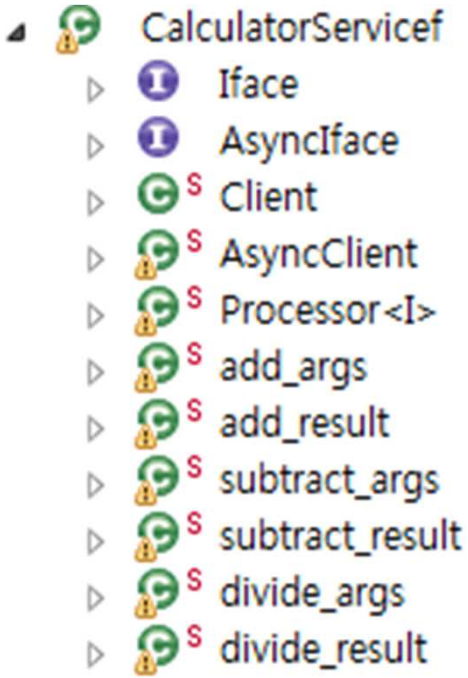
생성된 소스

자바


공유 대상
굽기
새 폴더
이름
수정한 날짜
유형
크기
CalculatorServicef.java
2011-09-26 오후...
JAVA 파일
78KB
MathException.java
2011-09-26 오후...
JAVA 파일
9KB

클래스

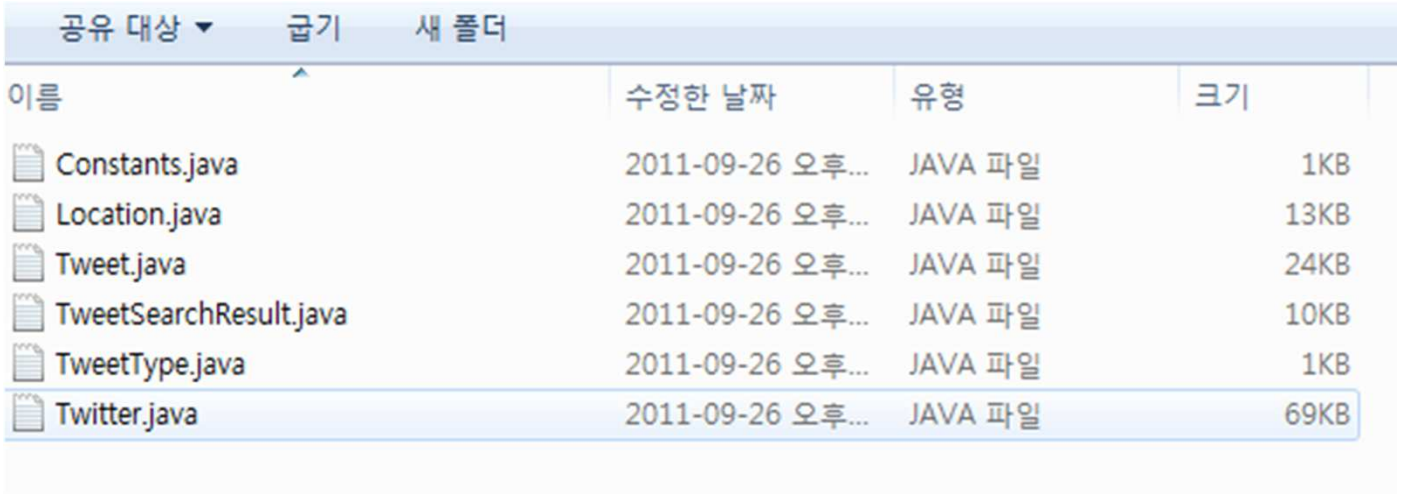

MathException
message : String
_Fields
MathException()
MathException(String)
MathException(MathException)
deepCopy() : MathException
clear() : void
getMessage() : String
setMessage(String) : MathException
unsetMessage() : void
isSetMessage() : boolean
setMessageIsSet(boolean) : void
setFieldValue(_Fields, Object) : void
getFieldValue(_Fields) : Object
isSet(_Fields) : boolean
equals(Object) : boolean
equals(MathException) : boolean
hashCode() : int
compareTo(MathException) : int
fieldForId(int) : _Fields
read(TProtocol) : void
write(TProtocol) : void
toString() : String
validate() : void


CalculatorServicef
Iface
AsyncIface
Client
AsyncClient
Processor<I>
add_args
add_result
subtract_args
subtract_result
divide_args
divide_result

샘플 #3

```
namespace cpp thrift.example
namespace java thrift.example

enum TweetType {
TWEET,
RETWEET = 2,
DM = 0xa,
REPLY
}
struct Location {
1: required double latitude;
2: required double longitude;
}
struct Tweet {
1: required i32 userId;
2: required string userName;
3: required string text;
4: optional Location loc;
5: optional TweetType tweetType = TweetType.TWEET;
16: optional string language = "english";
}
typedef list<Tweet> TweetList
struct TweetSearchResult {
1: TweetList tweets;
}
const i32 MAX_RESULTS = 100;
service Twitter {
void ping(),
bool postTweet(1:Tweet tweet);
TweetSearchResult searchTweets(1:string query);
oneway void zip()
}
```

생성 파일



- 상수는 Constants클래스로.
- Enum은 org.apache.thrift.TEnum 을 상속 받은 java enum으로 변경

Type

Types

- C/C++ style typedefs.

```
typedef i32 MyInteger
Typedef StrutType SStruct
```

- Enum

```
enum TweetType {
   TWEET,
   RETWEET = 2,
   DM = 0xa,
   REPLY
}

struct Tweet {
   1: required i32 userId;
   2: optional TweetType tweetType = TweetType.TWEET
}
```

기타

- comments
  - /* */
  - //
- namespace

```
namespace java com.example.project
```

- Include

```
include "tweet.thrift"
...
struct TweetSearchResult {
1: list<tweet.Tweet> tweets;
}
```

기타

- Constants

```
const i32 INT_CONST = 1234;
const map<string,string> MAP_CONST = {"hello": "world",
"goodnight": "moon"}
```

- 안되는 것
  - cyclic structs
  - Polymorphism
  - Overloading
  - null return
  - heterogeneous containers : items in a container must be of the same type (자바 generic을 생각)

IDL>Java

- bool: boolean
- byte: byte
- i16: short
- i32: int
- i64: long
- double: double
- string: String
- list<type>: List<type>
- set<type>: Set<type>
- Map<key, value>: Map<key, value>
- Typedef a b : just B

IDL->C++

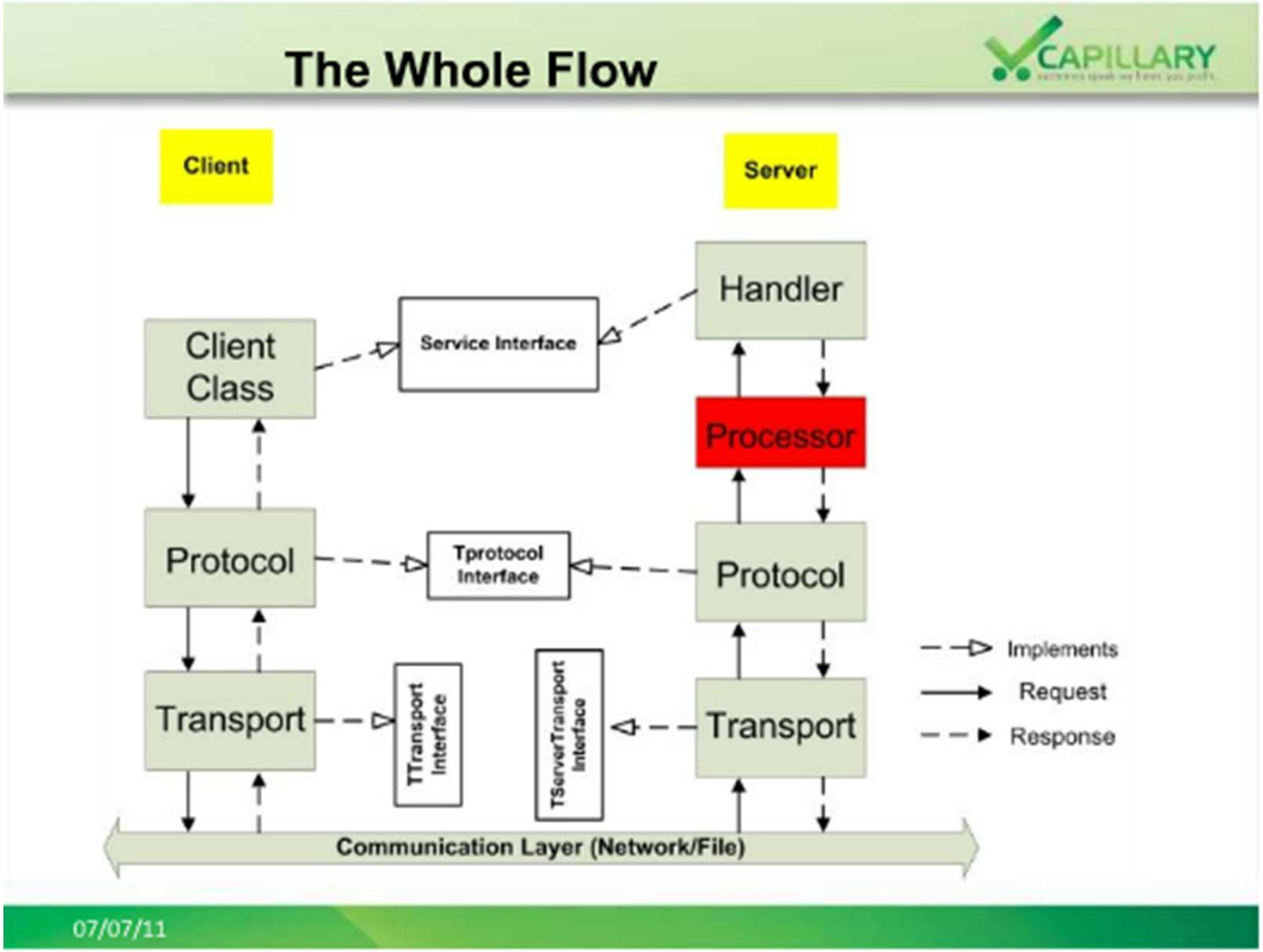
- bool: bool
- byte: int8_t
- i16: int16_t
- i32: int32_t
- i64: int64_t
- double: double
- string: std::string
- list<t1>: std::vector<t1>
- set<t1>: std::set<t1>
- map<t1,t2>: std::map<T1, T2>

Types

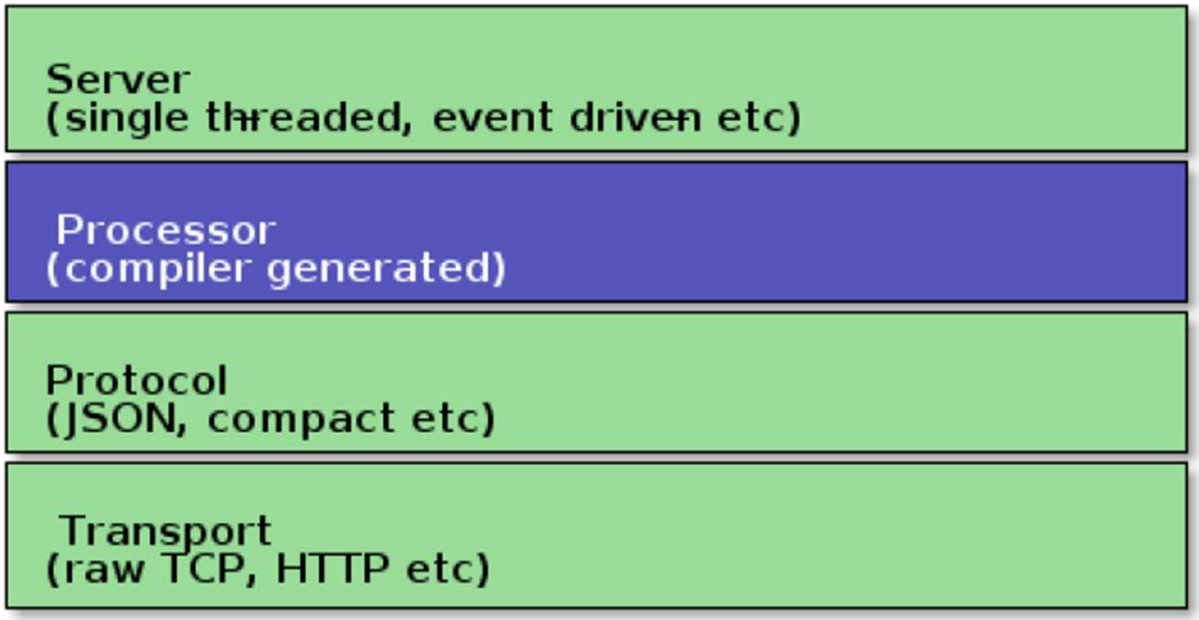
- Basic type
  - bool, byte, int(i16, i32, i64), double, string
- **Containers**
  - list<type>
  - Set<type>
  - Map<keyType, valueType>
- Struct (c와 비슷)
  - Struct안에 struct과 enum 사용가능
  - But, 선언은 {} 밖에서 해야 함. 안에서 선언금지
  - 상속 안됨
- Exception
  - Structs
- Service
  - Services are defined using Thrift types

Stacks

Flow


The Whole Flow
CAPILLARY
Client
Server
Handler
Client Class
Service Interface
Processor
Protocol
Tprotocol Interface
Protocol
Transport
TTransport Interface
TServerTransport Interface
Transport
Implements
Request
Response
Communication Layer (Network/File)
07/07/11

Thrift Network Stack



LanguageSupport

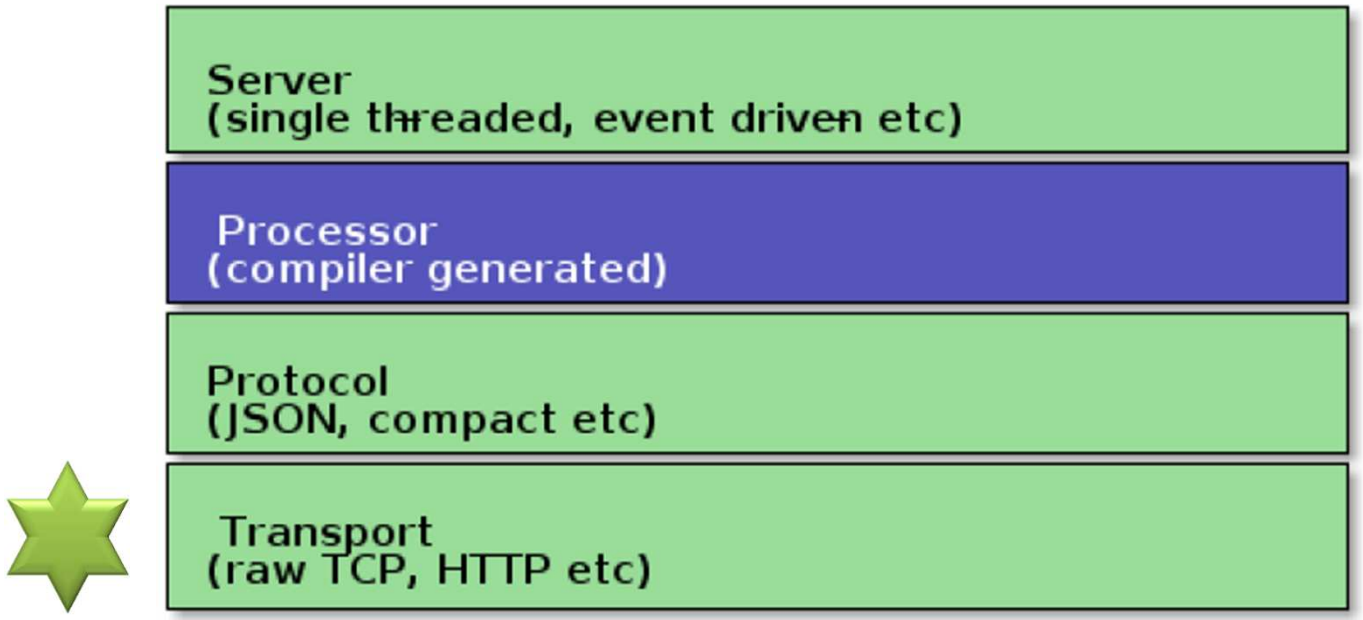
Legend	
Color	Description
green	available since version X.X
orange	patch available (JIRA-Ticket)
yellow	planned for version X.X
white	unknown
red	won't fix (JIRA-Ticket)

- http://wiki.apache.org/thrift/LibraryFeatures?action=show&redirect=LanguageSupport

(2011.9.25)

	Lang Features	Protocol Support				Transports		Servers			Clients		OS Support		
Language	Unions	Binary	Dense	Compact	JSON	Framed	SSL	Basic	Non-blocking	HTTP	Basic	HTTP	Win	OSX	Linux
Action Script 3 (as3)															
C Glib (c_glib)		0.6				0.6		0.6			0.6				0.6
C++(cpp)							0.7			0.4			THRIFT-757		
C# (csharp)					0.5		THRIFT-181			THRIFT-322					
Erlang (erl)															
Haskell (hs)															
Java (java)		0.2		0.2	0.2	0.2	0.5	0.2	0.2	0.4	0.2				
JavaScript (js)					0.3		0.3					0.3			
Node.js (js:node)		0.6				0.6			0.6		0.6				
Objective C (cocoa)															
OCaml (ocaml)															
Perl (perl)															
PHP (php)															
Python (py)							0.7								
Ruby (rb)															
Smalltalk(st)															

Thrift Network Stack



Transport Layer

- TFileTransport – This transport writes to a file.
- TFramedTransport – This transport is required when using a non-blocking server. It sends data in frames, where each frame is preceded by a length information.
- TMemoryTransport – Uses memory for I/O. The Java implementation uses a simple ByteArrayOutputStream internally.
- TSocket – Uses blocking socket I/O for transport.
- TZlibTransport – Performs compression using zlib. Used in conjunction with another transport. Not available in the Java implementation.

Transport interface

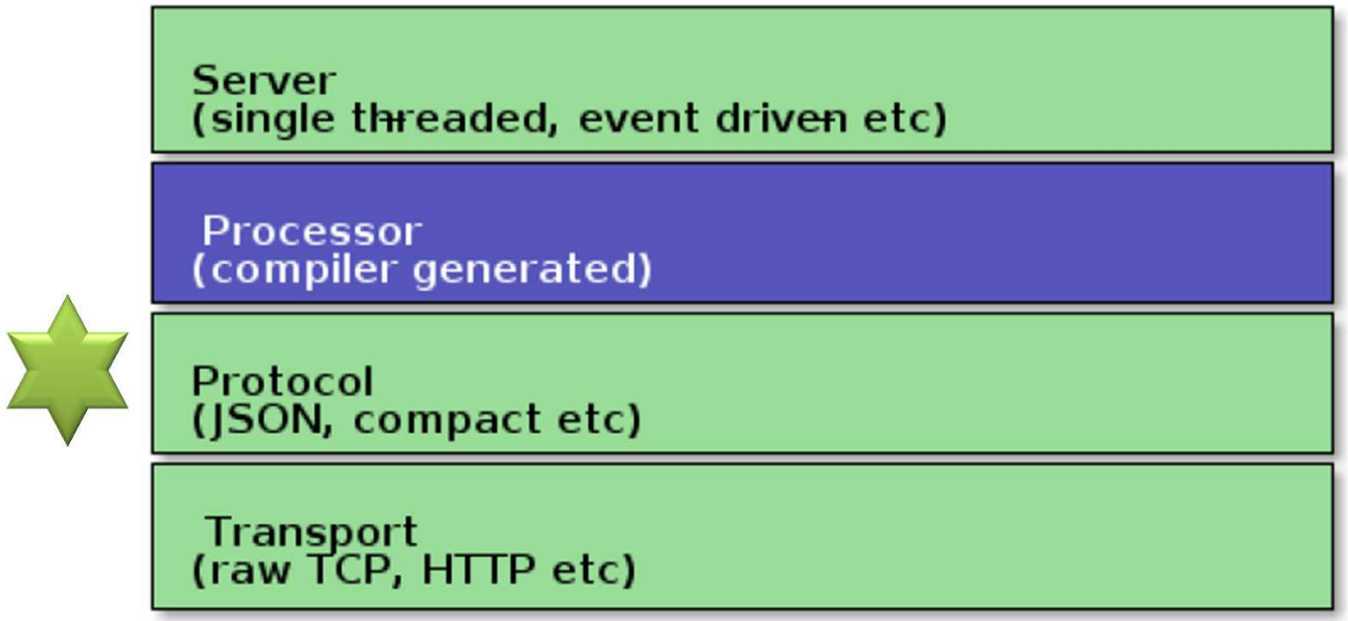
- Open
- Close
- Read
- Write
- Flush

Server Transport interface

- Open
- Listen
- Accept
- Close

- file: read/write to/from a file on disk
- http: as the name suggests

Thrift Network Stack



Protocol layer

- TBinaryProtocol – A straight-forward binary format encoding numeric values as binary. It is faster than the text protocol but more difficult to debug.
- TCompactProtocol – Very efficient, dense encoding of data.
- TDebugProtocol – Uses a human-readable text format to aid in debugging.
- TDenseProtocol – Similar to TCompactProtocol, striping off the meta information from what is transmitted.
- TJSONProtocol – Uses JSON for encoding of data.
- TSimpleJSONProtocol – A write-only protocol using JSON. Suitable for parsing by scripting languages.

Protocol interface

writeMessageBegin(name, type, seq)
writeMessageEnd()
writeStructBegin(name)
writeStructEnd()
writeFieldBegin(name, type, id)
writeFieldEnd()
writeFieldStop()
writeMapBegin(ktype, vtype, size)
writeMapEnd()
writeListBegin(etype, size)
writeListEnd()
writeSetBegin(etype, size)
writeSetEnd()
writeBool(bool)
writeByte(byte)
writeI16(i16)
writeI32(i32)
…
…
..

Json Serialization

Thrift

```
struct Job {
  1: i32 num1 = 0,
  2: i32 num2,
}
```

Generated Java Code by Compiler

```
public class Job implements org.apache.thrift.TBase<Job, Job._Fields>, java.io.Serializable,
Cloneable {
.....
}
```

Java code

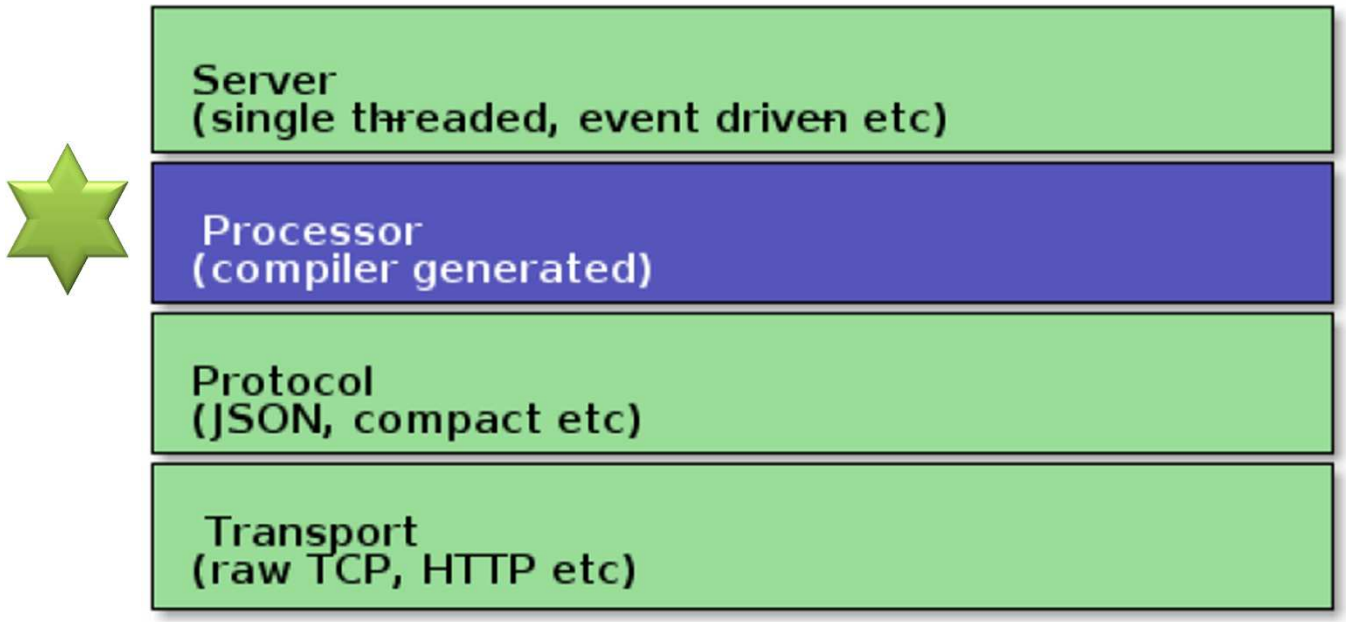
```
Job job;
....

TSerializer serializer = new TSerializer(new TSimpleJSONProtocol.Factory());
String json = serializer.toString(job);
```

Binary Serialization

Java code

```
// serialization code
TSerializer serializer = new TSerializer(new TBinaryProtocol.Factory());
byte[] bytes = serializer.serialize(new Job());

// deserialization code
TDeserializer deserializer = new TDeserializer(new TBinaryProtocol.Factory());
Job job= new Job();
deserializer.deserialize(job, bytes);
```

Thrift Network Stack

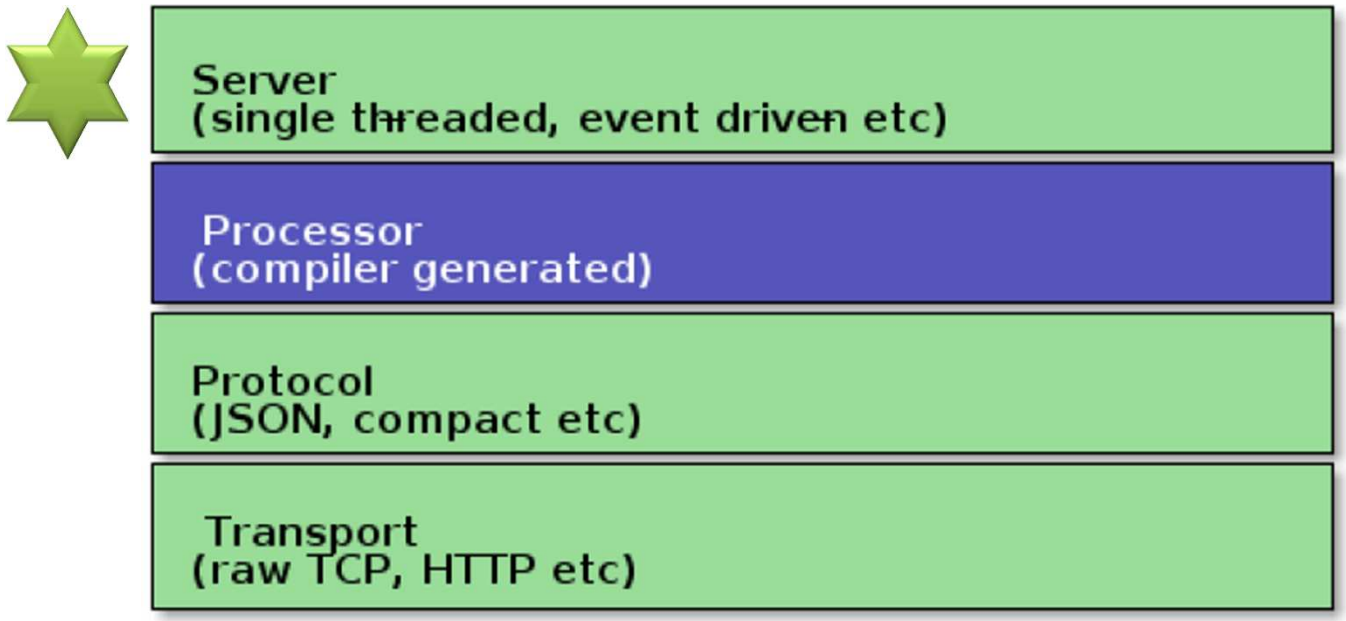


processor

- Thrift 컴파일러에 의해서 interface에 맞는 processor가 생성됨
- Processor는 input/ouput stream을 관장하는 클래스(encapsulates the ability to read data from input streams and write to output streams.)

```
interface TProcessor {
bool process(TProtocol in, TProtocol out) throws TException
}
```

Thrift Network Stack



Server Layer

- TNonblockingServer – A multi-threaded server using non-blocking io (Java implementation uses NIO channels). TFramedTransport must be used with this server.
- TSimpleServer – A single-threaded server using std blocking io. Useful for testing.
- TThreadPoolServer – A multi-threaded server using std blocking io.

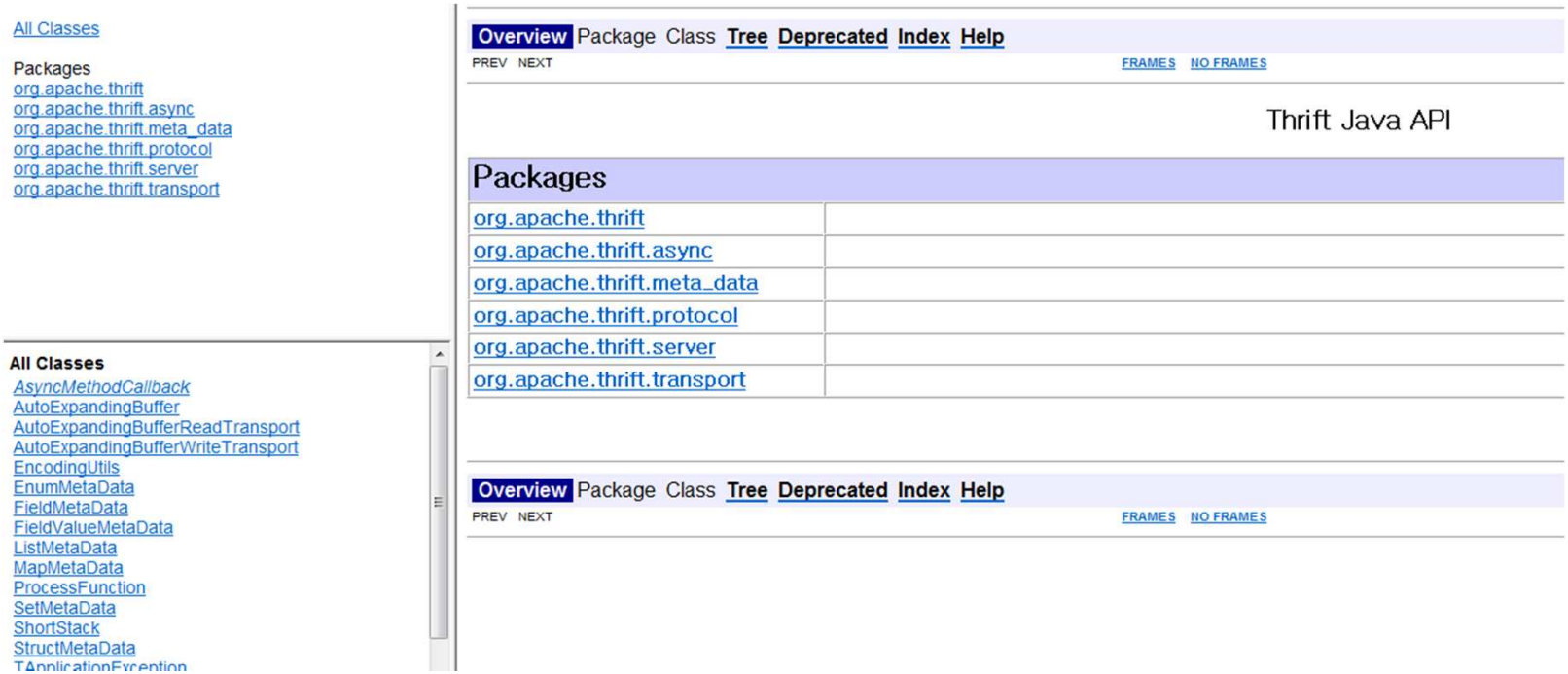
기타

기존 IDL를 변경할 때..

- 기존 IDL를 바꿀때마다 고생하지 않아도 되는 방법
  - Don’t change the numeric tags for any existing fields.
  - Any new fields that you add should be optional.
  - Non-required fields can be removed, as long as the tag number is not used again in your updated message type
    "OBSOLETE_“으로 이름 변경하기
  - Changing a default value is generally OK, as long as you remember that default values are never sent over the wire.

Thrift API

- http://people.apache.org/~jfarrell/thrift/0.7.0/javadoc/



끝