LINE DEVELOPER DAY_2015

Armeria
LINE's next generation RPC Layer

Heui-seung Lee | LINE Server
LINE

CONTENTS

00

Wait, but why?

What we use today .. or facepalm

Thrift-over-HTTP/1 on top of Tomcat Servlet

Synchronous calls everywhere

Many threads and context switches

“Thread full? Let’s submit non-critical tasks in other thread pool.”

“Full again? Let’s create TP-A for task type A and TP-B for task type B, …”

It is false asynchrony, and tasks on critical paths are still synchronous.

01

What is Armeria?

What is Armeria?

(A)synchronous RPC library

.. built on top of:

Java 8	Lambda functions and nice service composition
Netty 4.1	High performance and full asynchrony
HTTP/1 & 2	Rich toolset and smooth upgrade path
Thrift	Our universal payload format

Armeria + Netty 4.1

Is your RPC really async?

When async is (not) async:

Connection pooling - `pool.acquire()`

Domain name lookups - `InetAddress.getByName()`

HTTP/2 support - head-of-line blocking

Armeria + HTTP

Unmatchable backward compatibility – HTTP/1

`cURL`, `wget`, and even `telnet`

`THttpClient` – the official Thrift-over-HTTP client

Future is now – HTTP/2

Your web browser

Armeria client

Others are catching up – e.g. `cURL`

02

Writing a Thrift service

Your first Thrift service

```
// HelloService.* are generated by the official Thrift compiler.
HelloService.AsyncIface helloHandler = new HelloService.AsyncIface() {
    public void hello(String name, AsyncMethodCallback cb) {
        cb.onComplete("Hello, " + name + '!');
    }
};
// or if the service has a single operation:
helloHandler = (name, cb) -> cb.onComplete("Hello, " + name + '!');
// or synchronously:
HelloService.Iface helloHandler = name -> "Hello, " + name + '!';
```

Did you notice?

Compatibility with Apache Thrift

Uses the code generated by the official Thrift compiler

Switching from Apache Thrift or Nifty should be a breeze (and vice versa.)

Fully compatible with the official `THttpClient`

Bootstrapping an Armeria server

```
ServerBuilder sb = new ServerBuilder();
sb.port(8080, SessionProtocol.HTTP);
sb.defaultVirtualHost(new VirtualHostBuilder().serviceAt(
        "/hello",
        new ThriftService(helloHandler, ThriftProtocolFactories.BINARY)
                .decorate(LoggingHandler::new)).build());

Server server = sb.build();
server.start();
```

Client side?

```
HelloService.Iface helloService = Clients.newClient(
        "tbinary+http://example.com:8080/hello",
        HelloService.Iface.class); // or AsyncIface.class

String greeting = helloService.hello("Armerian World");
assert greeting.equals("Hello, Armerian World!");
```

03

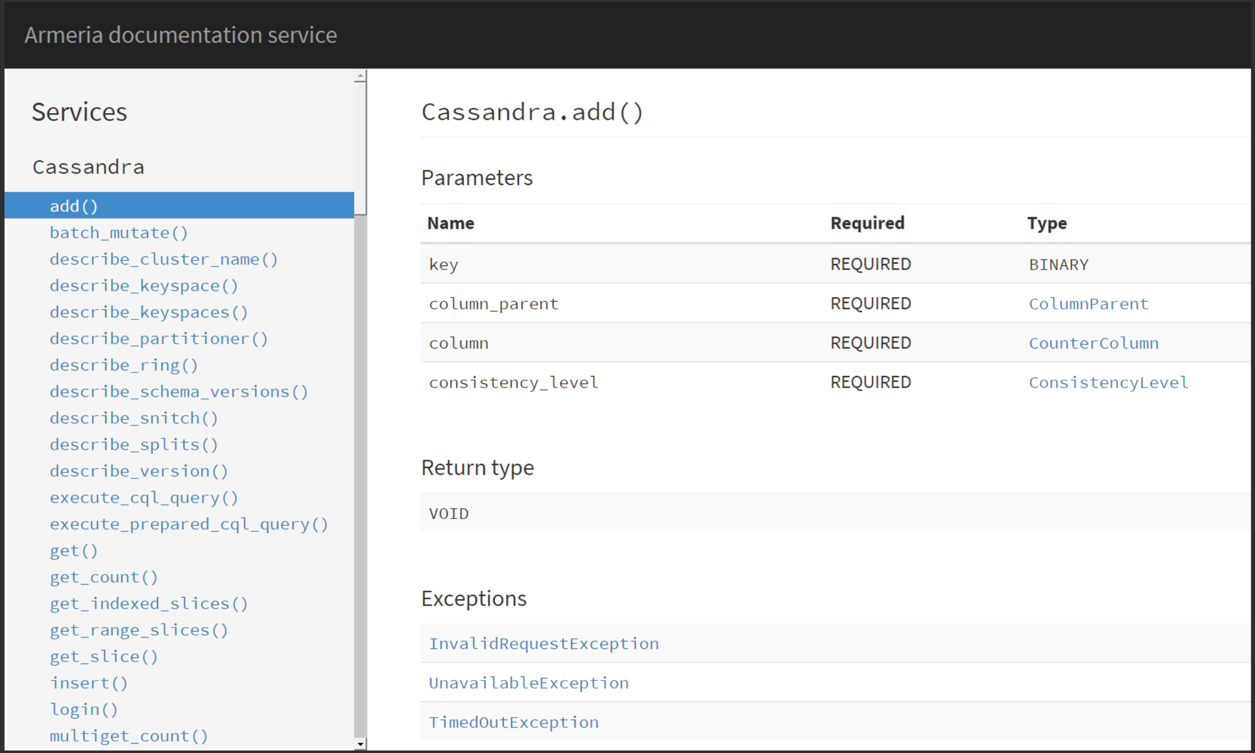
Browsing Thrift services

Armeria documentation service

Auto-discovery & metadata construction

```
VirtualHostBuilder vhb = new VirtualHostBuilder();
vhb.serviceAt("/foo/", new ThriftService(…))
   .serviceAt("/bar/", new ThriftService(…));
   .serviceUnder("/docs/", new DocService());
```

All from your web browser



Armeria documentation service

Services

Cassandra

- add()
- batch_mutate()
- describe_cluster_name()
- describe_keyspace()
- describe_keyspaces()
- describe_partitioner()
- describe_ring()
- describe_schema_versions()
- describe_snitch()
- describe_splits()
- describe_version()
- execute_cql_query()
- execute_prepared_cql_query()
- get()
- get_count()
- get_indexed_slices()
- get_range_slices()
- get_slice()
- insert()
- login()
- multiget_count()

Cassandra.add()

Parameters

Name	Required	Type
key	REQUIRED	BINARY
column_parent	REQUIRED	ColumnParent
column	REQUIRED	CounterColumn
consistency_level	REQUIRED	ConsistencyLevel

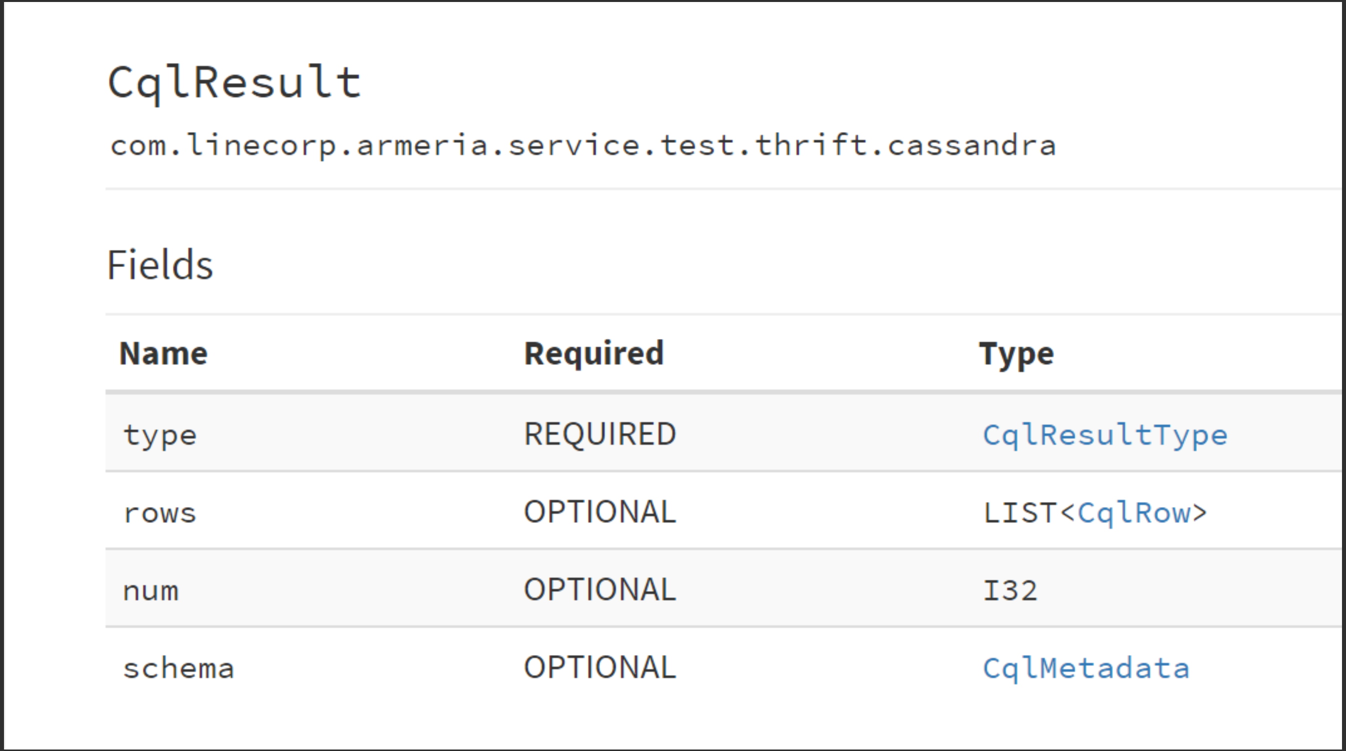
Return type

VOID

Exceptions

- InvalidRequestException
- UnavailableException
- TimedOutException

All from your web browser



CqlResult

com.linecorp.armeria.service.test.thrift.cassandra

Fields

Name	Required	Type
type	REQUIRED	CqlResultType
rows	OPTIONAL	LIST<CqlRow>
num	OPTIONAL	I32
schema	OPTIONAL	CqlMetadata

04

Advanced stuff

Embedding Apache Tomcat

Enjoy the best of the two worlds :-)

```
virtualHostBuilder.serviceUnder(
        "/myapp/", TomcatService.forCurrentClassPath());
```

Powered by custom connector

Tomcat opens 'zero' sockets.

Tomcat gets HTTP/2 support for free!

Let Armeria serve your RPC calls at its best performance
    while serving your rich webapp in the same port/JVM.

Using as an HTTP/1 & 2 client

Not just an RPC call

```
SimpleHttpClient client = Clients.newClient(
        "none+http://example.com", SimpleHttpClient.class);

SimpleHttpRequest req =
        SimpleHttpRequestBuilder.forGet("/foo/bar.json")
                                .header("Accept", "application/json")
                                .build();

Future<SimpleHttpResponse> future = client.execute(req);
SimpleHttpResponse response = future.get();

assert res.status().code() == 200;
```

05

Food for thought

Food for thought

The yaks we didn't shave (yet)

Metrics	- Evolve DocService into something great
Thrift compiler	- Better code generation
Large requests	- Reactive stream?
Session protocols	- MUX?
Serialization formats	- Protobuf?
Integration	- JMX, Swagger, Authn/z framework, …
Experience	- Spring Boot-like experience

06

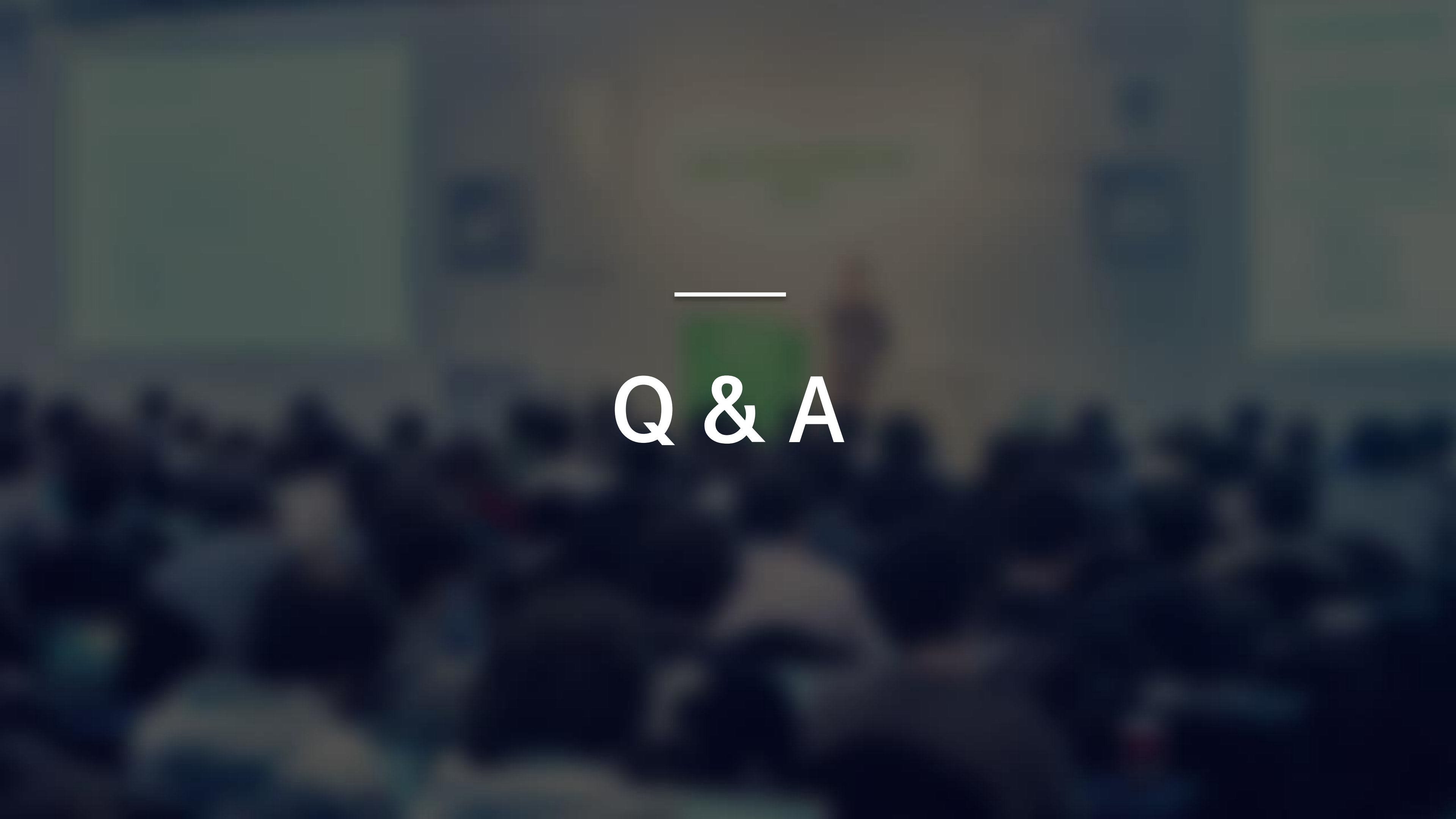
.. and maybe you could help us!?

We're open-sourcing Armeria

Try it and let us know what you think!

Did we mention we're using it in production already? ;-)

`line.github.io/armeria`



Q & A

THANK YOU
FOR
LISTENING.