SLEUTH + ZIPKIN

Kim Saabye Pedersen

SLEUTH – to track or follow

SLEUTH

Add (assuming spring-cloud BOM) to pom.xml:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
</dependency>
```

Now your logs look like this:

```
2016-06-06 09:27:34.341  INFO [appName,889d464470a393fe,889d464470a393fe,true]
```

- appName: the name of your app as defined by spring.application.name
- *889d464470a393fe*: the correlationId (aka traceId in Spring Guide)
- 889d464470a393fe: the id of the specific operation (spanId)

And so what….

In a distributed system you want this.

```
2016-06-06 09:27:34.341  INFO [appName,889d464470a393fe,889d464470a393fe,true]
```

In this case correlationId == spanId: Graph started by this “invocation”.

```
2016-06-06 10:19:07.841  INFO [appName,f6ad402cbcb55b2c,47c5cc5f1b4692,true]
```

If correlationId != spanId:

- traceId comes from other app or you created a new span.

And now you can correlate your logs.

DEMO

sleuthDemoApp: http://localhost:9987

someApp: http://localhost:9547

What Spring gives you

A lot of instrumentation to keep correlation ids across:

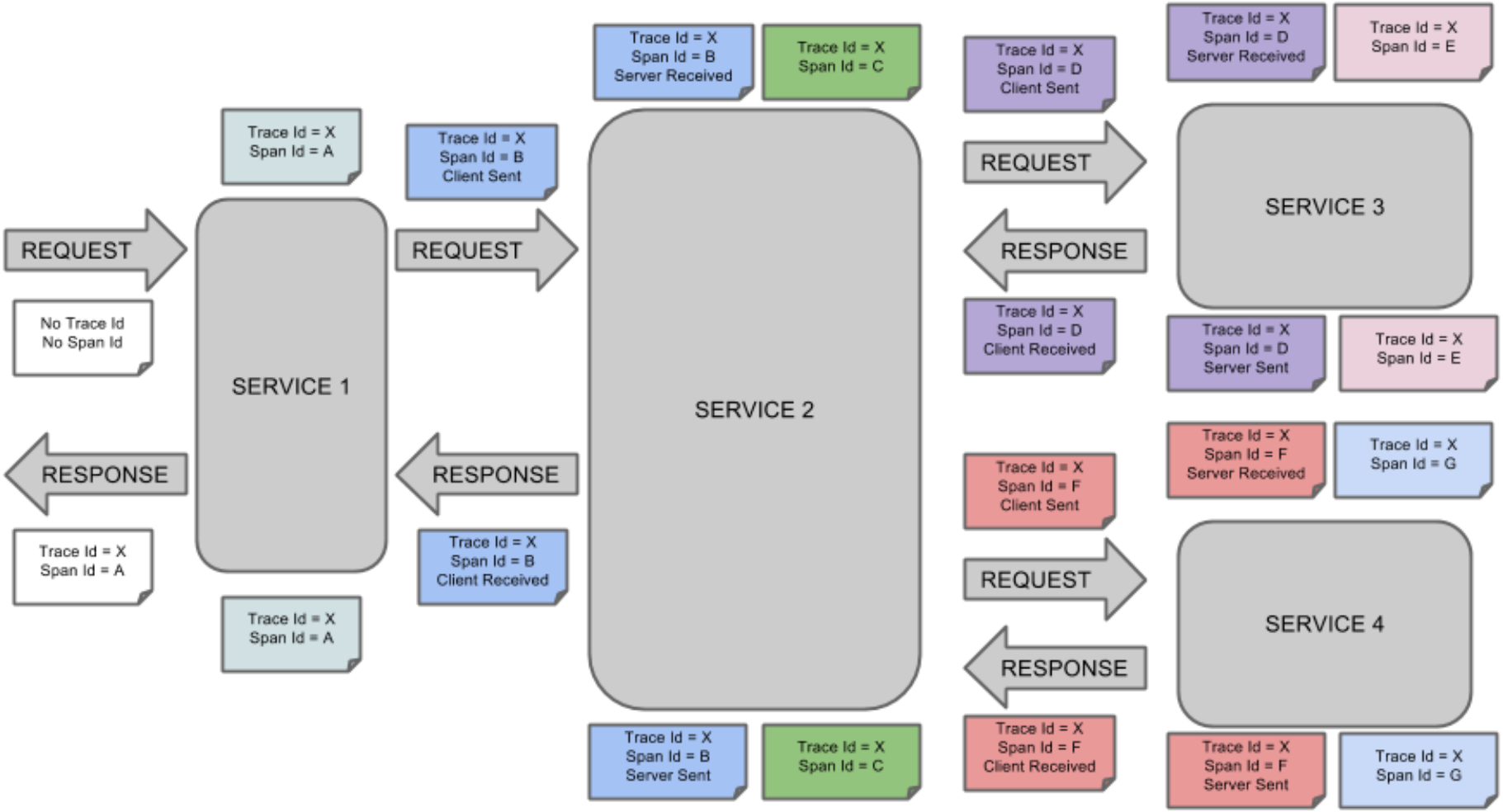
- Http & RestTemplate (incl. async)
- ExecutorService + RxJava
- Messaging
- Hystrix

+ a way to roll your own.

Under the hood

- **cs** - Client Sent - The client has made a request. This annotation depicts the start of the span.
- **sr** - Server Received - The server side got the request and will start processing it. If one subtracts the cs timestamp from this timestamp one will receive the network latency.
- **ss** - Server Sent - Annotated upon completion of request processing (when the response got sent back to the client). If one subtracts the sr timestamp from this timestamp one will receive the time needed by the server side to process the request.
- **cr** - Client Received - Signifies the end of the span. The client has successfully received the response from the server side. If one subtracts the cs timestamp from this timestamp one will receive the whole time needed by the client to receive the response from the server.

An example


REQUEST
No Trace Id
No Span Id
SERVICE 1
RESPONSE
Trace Id = X
Span Id = A
Trace Id = X
Span Id = A
Trace Id = X
Span Id = B
Client Sent
REQUEST
RESPONSE
Trace Id = X
Span Id = B
Client Received
Trace Id = X
Span Id = B
Server Received
Trace Id = X
Span Id = C
SERVICE 2
Trace Id = X
Span Id = B
Server Sent
Trace Id = X
Span Id = C
Trace Id = X
Span Id = D
Client Sent
REQUEST
RESPONSE
Trace Id = X
Span Id = D
Client Received
Trace Id = X
Span Id = D
Server Received
Trace Id = X
Span Id = E
SERVICE 3
Trace Id = X
Span Id = D
Server Sent
Trace Id = X
Span Id = E
Trace Id = X
Span Id = F
Client Sent
REQUEST
RESPONSE
Trace Id = X
Span Id = F
Client Received
Trace Id = X
Span Id = F
Server Received
Trace Id = X
Span Id = G
SERVICE 4
Trace Id = X
Span Id = F
Server Sent
Trace Id = X
Span Id = G

ZIPKIN

Zipkin – aggregation of spans (in-mem, db)

Zipkin UI – query, analyze, visualize spans

http://localhost:9966