LogStage: Zero-cost Structural Logging for Scala

Septimal Mind Ltd

team@7mind.io

July 24, 2018

What's wrong with logging frameworks?

- Do we need structured logs? Yes, obviously,
- Logging frameworks are convenient for a programmer *xor* for a machine, not for both,
- Logging frameworks love to break SOLID,
- Magic rituals required!

`fluentd` logging API example

```
1  object Example {
2    // We love rituals! Is it SOLID, hmm?..
3    val LOG = FluentLoggerFactory
4      .getLogger("fluentd.test")
5
6    // ...
7
8    val data =
9        new HashMap[String, String]()
10   data.put("from", "userA")
11   data.put("to", "userB")
12   LOG.log("follow", data)
13 }
```

`scala-logging` API example

```scala
class Example
  extends LazyLogging { // Let's break SOLID!
  //...

  // Renders as "Received message from JohnDoe"
  // Structure lost
  logger.trace(s"Received message from $user")
}
```

The code...

```
1 val user = "JohnDoe"
2 logger.debug(s"Received a message from $user")
```

. . . is structured

```
Expr(Apply(Select(
  Apply(
    Select(Select(Ident("scala"), scala.StringContext),
      TermName("apply"))
      , List(Literal(Constant("Received a message from "))
          , Literal(Constant(""))
        )
  ),
  TermName("s")
  )
, List(Ident(TermName("user")))
))
```

The code is always structured

- We have argument names, types and order defined in code,
- As well we have some static information about the context – file, line, etc,
- We have static part of our message – interpolation context or *message template*,
- We may process our string interpolations with a macro, recover structure and pass it to a logger.

LOGSTAGE

First-class logging framework for Scala

Quick overview

- Almost no dependencies,
- Compile-time structure and context extraction,
- Console and file sinks out of the box, log rotation supported,
- Asynchronous sink out of the box (single worker thread at the moment),
- String and JSON rendering out of the box,
- Automatic structure identifiers for JSON policy,
- Modular – you may implement your own sink, router, etc,
- DI-ready, no singletons or classpath scanners,
- **Method**-level granularity,
- User-provided logging contexts,
- Slf4J backend – LogStage is a drop-in replacement for Logback, route your legacy logs,
- Location hyperlinks for IntelliJ console.

An example

```
1  class ExampleService(log: IzLogger) {
2    val justAnArg = "example"
3    val justAList = List[Any](10, "green", "bottles")
4    val msec = Random.nextInt(1000)
5    log.trace(s"Argument: $justAnArg, another arg: $justAList")
6    log.info(s"Expr: ${Random.nextInt() -> "number"}")
7    log.warn(s"Hidden: ${Random.nextInt() -> "number" -> null}")
8    val ctxLog = log("userId" -> "user@google.com"
9      , "company" -> "acme")
10   ctxLog.info(s"Processing time: $msec")
11 }
```

Followed by a cute screenshot of course:

```
I 2018-07-23T23:07:14.946+01:00[Europe/Dublin] ...leSinkTest:1  (LoggingAsyncSinkTest.scala:20) Argument: ⤢
↘justAnArg=example, another arg: justAList=List(10, green, bottles)
I 2018-07-23T23:07:15.056+01:00[Europe/Dublin] ...leSinkTest:1  (LoggingAsyncSinkTest.scala:21) Expr: number=1601070974
W 2018-07-23T23:07:15.062+01:00[Europe/Dublin] ...leSinkTest:1  (LoggingAsyncSinkTest.scala:22) Hidden: -539747744
```

Something nice for our robots


```
{
"just_a_list":[10,"green","bottles"],
"just_an_arg":"example",
"@event":{
  "class":"f48ebb70",
  "logger": "...logstage.api.routing.ExampleService.start",
  "file":"LoggingAsyncSinkTest.scala", "line":20,
  "thread":{ "id":1, "name":"ScalaTest-run-running-LoggingJson4sTest" },
  "level":"trace",
  "timestamp":1532384023837,
  "datetime":"2018-07-23T22:13:43.837Z[UTC]"
},
"@template":
"Argument: ${just_an_arg}, another arg: ${just_a_list}",
"@message":
"Argument: justAnArg=example, another arg: justAList=List(10, green, bottles)"
}
```

Status and things to do

LogStage is.

- ready to use,
- in real production for 4 months.

Our plans:

- Declarative router config,
- Start using log events to collect metrics (keep in mind, we have derived structure identifiers),
- Cleanups and refactorings.

Thank you for attention

https://izumi.7mind.io/logstage/

We're looking for clients, contributors, adopters and colleagues ;)

About the author:

- coding for 18 years, 10 years of hands-on commercial engineering experience,
- has been leading a cluster orchestration team in Yandex, "the Russian Google",
- implemented "*Interstellar Spaceship*" – an orchestration solution to manage 50K+ physical machines across 6 datacenters,
- Owns an Irish R&D company, https://7mind.io,
- Contacts: team@7mind.io,
- Github: https://github.com/pshirshov