LogStage: Zero-cost Structural Logging for Scala

Septimal Mind Ltd

team@7mind.io

July 26, 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!

`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")
}
```

`fluentd` logging API example

```scala
object Example {
  // We love rituals! Is it SOLID, hmm?..
  val LOG = FluentLoggerFactory
    .getLogger("fluentd.test")

  // ...

  val data =
      new HashMap[String, String]()
  data.put("from", "userA")
  data.put("to", "userB")
  LOG.log("follow", data)
}
```

The code...

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

. . . is structured

```
1   Expr(Apply(Select(
2     Apply(
3       Select(Select(Ident("scala"), scala.StringContext),
4         TermName("apply"))
5         , List(Literal(Constant("Received a message from "))
6             , Literal(Constant("")))
7           )
8     ),
9     TermName("s")
10    )
11  , List(Ident(TermName("user"))))
12  ))
```

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.

```scala
def trace(message: String): Unit = macro doTrace

def doTrace(c: blackbox.Context)(message: c.Expr[String]): c.Expr[Unit] = {
  ???
}
```

L★GSTAGE

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

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

Followed by a cute screenshot of course:

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 your 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