Akka HTTP: The Reactive Web Toolkit

Dr. Roland Kuhn
@rolandkuhn — Akka Tech Lead
Live Demo
API Design

• goals:
  • no magic
  • supreme compositionality
  • exhaustive model of bounded stream processing

• consequences:
  • immutable and reusable stream blueprints
  • explicit materialization step

Materialization

• Akka Streams separate the *what* from the *how*
  • declarative Source/Flow/Sink DSL to create blueprint
  • FlowMaterializer turns this into running Actors
• this allows alternative materialization strategies
  • optimization
  • verification / validation
  • cluster deployment
• only Akka Actors for now, but more to come!
Akka HTTP
Why do we add an HTTP module?

• Akka is about building distributed applications
• distribution implies integration
  • between internal (sub)systems
    ➔ akka-cluster based on akka-remote
  • with external systems
    ➔ akka-http (lingua franca of the internet)
HTTP Stream Topology
The Application Stack

user-level

Akka HTTP

Akka HTTP Core

Akka IO

Java NIO (JDK)

O/S-level network stack
Akka HTTP Core

• implements HTTP/1.1 according to the RFCs
• based upon the TCP facilities of Akka IO
• exposed as freely reusable stream transformations (BidiFlows & Flows)
• low-level and extensible implementation of HTTP model and spec
Stream Pipelines

TCP

Optional (not yet*)

SSL

HTTP

Responses

App

Requests

* It was literally sooooo close, but didn’t make it for ScalaDays.
A Quick View of the HTTP Model

case class HttpRequest(
    method: HttpMethod = HttpMethods.GET,
    uri: Uri = Uri./,
    headers: immutable.Seq[HttpHeader] = Nil,
    entity: RequestEntity = HttpEntity.Empty,
    protocol: HttpProtocol = HttpProtocols.`HTTP/1.1`)
) extends HttpMessage

case class HttpResponse(
    status: StatusCode = StatusCodes.OK,
    headers: immutable.Seq[HttpHeader] = Nil,
    entity: ResponseEntity = HttpEntity.Empty,
    protocol: HttpProtocol = HttpProtocols.`HTTP/1.1`)
) extends HttpMessage
Akka HTTP

• builds upon Akka HTTP Core
• adds (un)marshaling for HttpEntities
• adds (de)compression (gzip / deflate)
• high-level server-side routing DSL
  • type-safe and flexible
  • highly compositional building blocks (Directives)
  • includes common behaviors pre-canned
Live Demo
When can we have it?

- currently pre-release versions:
  - reactive-streams 1.0.0-RC3
  - Akka Streams & HTTP 1.0-M4

- still missing:
  - Akka HTTP client features
  - SSL integration
  - websockets

- Akka Streams & HTTP 1.0 expected end of Feb’15
Resources

• https://github.com/akka/akka/tree/release-2.3-dev
• akka-user mailing list