# spray 101

laogao

2013-10-19

# 关于我

江湖人称老高

丰益咨询集团架构师

**2009** 年开始 **Scala** 之旅

译有《快学 **Scala**》

新浪微博：@程序员老高

# 议题

- spray 概览
- spray 架构
- spray 模块
- spray-routing
- directives
- 案例分析

# spray 概览

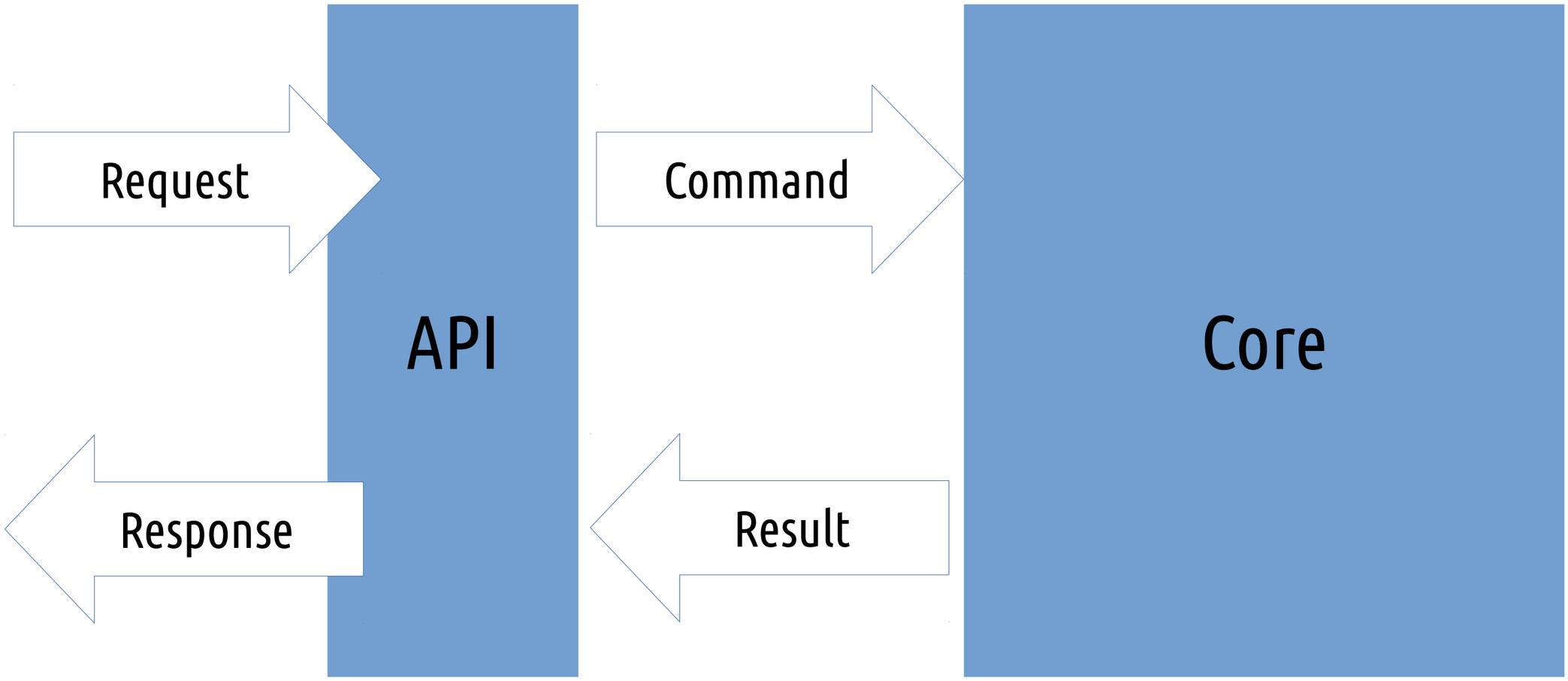
- 基于 Akka 构建和使用 RESTful 服务的工具包
- 基本原则： 轻、异步、非阻塞、模块化、低依赖、可测试
- 哲学 / 价值观： 类库，非框架

# 议题

- spray 概览
- spray 架构
- spray 模块
- spray-routing
- directives
- 案例分析

# spray 架构

符合 spray 价值观的典型应用架构



# spray 架构

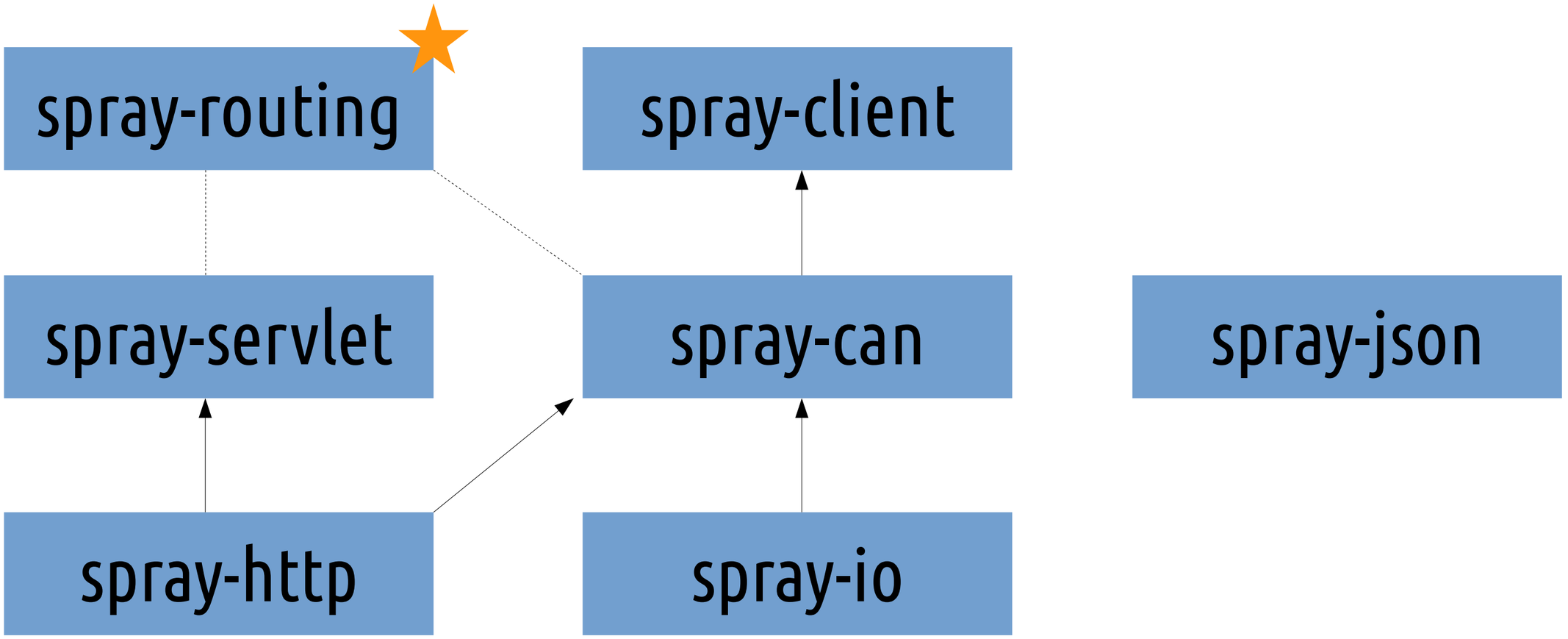
## API 层的职能

- 请求响应 / 路由
- (un)marshalling
- 认证 / 授权
- 缓存
- 错误处理
- ...

# 议题

- spray 概览
- spray 架构
- spray 模块
- spray-routing
- directives
- 案例分析

# spray 模块


spray-routing
spray-client
spray-servlet
spray-can
spray-json
spray-http
spray-io

# 议题

- spray 概览
- spray 架构
- spray 模块
- spray-routing
- directives
- 案例分析

# spray-routing

- 运行于 spray-can 之上，也可以通过 spray-servlet 运行在 servlet 容器中
- 用于构建 API 层的各种工具类
- 用于定义 RESTful API/ 路由的 DSL
- 重点是 RESTful API ，而非基于网页的 Web 应用

# 议题

- spray 概览
- spray 架构
- spray 模块
- spray-routing
- directives
- 案例分析

# directives

- 对传入的请求进行数据转换 ( 抽取参数、表单信息、数据加工等 )
- 根据某种逻辑拒绝特定请求
- ...
- directive 也可以被重新组合，并且是类型安全的

# 议题

- spray 概览
- spray 架构
- spray 模块
- spray-routing
- directives
- 案例分析

# 案例分析

## RESTful Data Exchange Platform (JSON)

- 内部项目、典型的企业环境应用
- 用于 SAP 和应用系统之间的订单数据交换
- SAP 端的接口是 .NET 实现
- 应用系统是 Play(Java) 实现
- 从需求到上线仅 1 人 x2 天
- 目前已稳定运行数月 0 故障
- 团队技术背景为 Java EE 6
- 之前仅使用过 Scala 和 Akka ，并无任何 spray 经验

# 案例分析

- Scala 2.10.1
- Spray 1.1-M7
- Argonaut 1.0
- Akka microkernel 2.1.4
- MongoDB 2.4.4

# 案例分析

```
libraryDependencies ++= Seq(
  "com.typesafe.akka" %% "akka-actor" % "2.1.4"
, "com.typesafe.akka" %% "akka-kernel" % "2.1.4"
, "com.typesafe.akka" %% "akka-slf4j" % "2.1.4"
, "com.typesafe.akka" %% "akka-remote" % "2.1.4"
, "com.typesafe.akka" %% "akka-testkit" % "2.1.4"
, "io.spray" % "spray-http" % "1.1-M7"
, "io.spray" % "spray-can" % "1.1-M7"
, "io.spray" % "spray-routing" % "1.1-M7"
, "org.mongodb" %% "casbah" % "2.6.1"
, "io.argonaut" %% "argonaut" % "6.0"
, "ch.qos.logback" % "logback-classic" % "1.0.13")
```

# 案例分析

```
import spray.http.{ContentType, HttpBody}
import spray.http.MediaTypes._
import spray.httpx.marshalling._
import spray.httpx.unmarshalling._

implicit val MaterialMarshaller =
  Marshaller.of[Material](`application/json`) {
    (value, contentType, ctx) =>
      ctx.marshalTo(HttpBody(contentType, value.asJson.toString))
  }
implicit val MaterialUnmarshaller =
  Unmarshaller.delegate[String, Material](`application/json`) { str =>
    str.decodeOption[Material].get
  }
```

# 案例分析

```
val route = {
  authenticate(BasicAuth()) { user =>
    val owner = user.username match {
      case x if x.startsWith("O_") => x.substring(2)
      case _ => ""
    }
    respondWithMediaType(`application/json`) {
      get {
        path("order/byid" / PathElement) { orderId =>
          complete { (orderService ? GetOrderById(owner, orderId)).mapTo[Option[Order]] }
        } ~
        ...
      } ~
      put {
        ...
      }
    }
  }
}
```

# 案例分析

```
class Bootstrapper extends Bootable with SprayCanHttpServerApp {
  val config = ConfigFactory.load()
  def startup = {
    system.actorOf(Props[OrderService], "order-service")
    // ...
    val server = system.actorOf(Props[RestActor], "server")
    newHttpServer(server) ! Bind(
      interface = config.getString("rest.listening"),
      port = config.getInt("rest.port")
    )
  }
  def shutdown = {
    system.shutdown()
  }
}
```

# 问答

# 推荐阅读

https://leanpub.com/theneophytesguidetoscala