LEGO

let’s build everything by scala

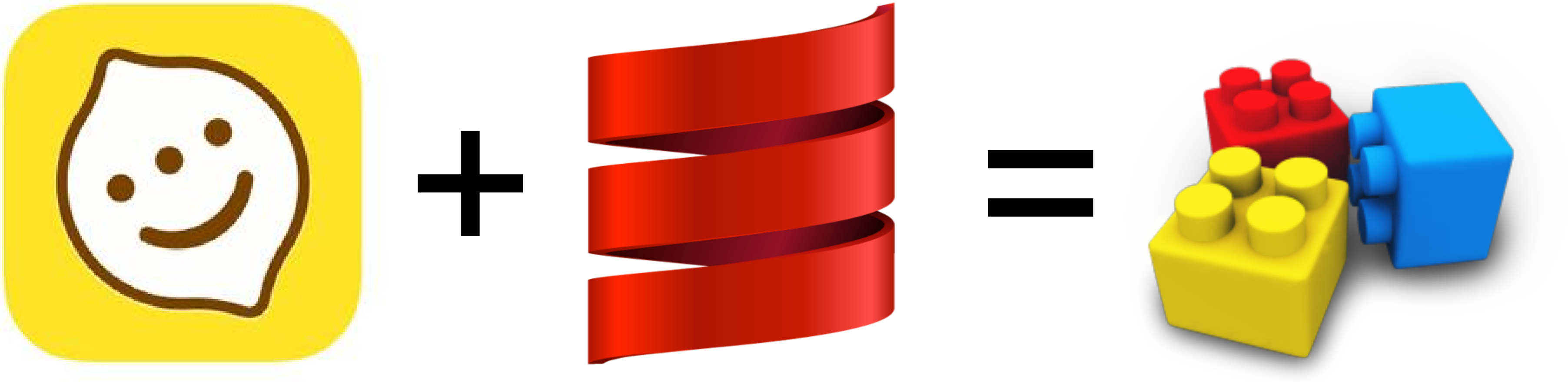
聚石 @ alibaba-inc.com

关于我

- Scala 爱好者, CSUG 成员, HouseMD的作者
- 来往后端基础服务
- 淘宝中间件
- github.com/zhongl

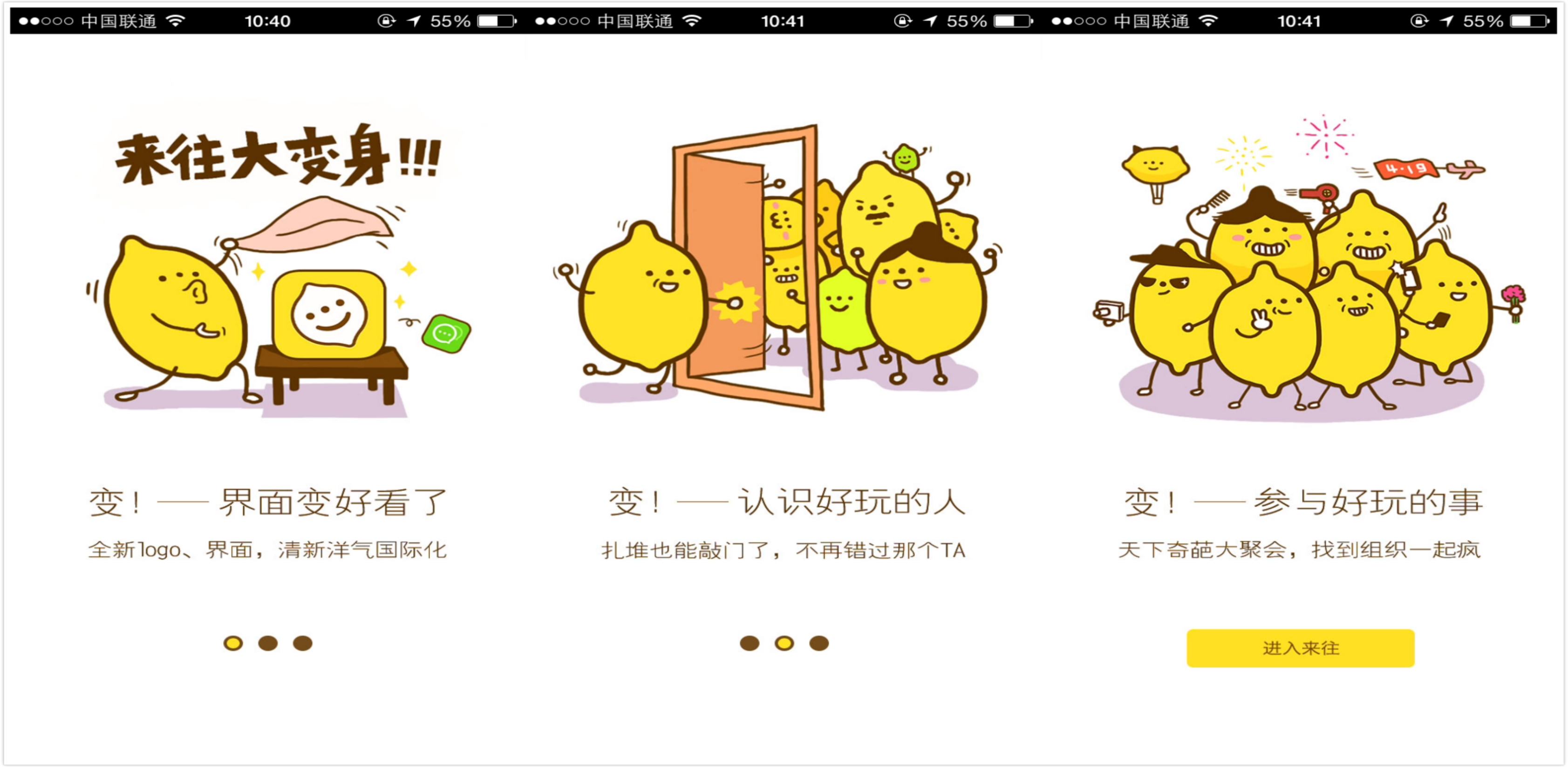
大纲

- LEGO 由来和理念
- Scala 在应用中的探索
- Scala 在布道中的反思



LEGO 的由来和理念

扎堆来往 http://laiwang.com



变！—— 界面变好看了

全新logo、界面，清新洋气国际化

变！—— 认识好玩的人

扎堆也能敲门了，不再错过那个TA

变！—— 参与好玩的事

天下奇葩大聚会，找到组织一起疯

从有线到无线, HTTP 无法满足移动 IM 的场景

Nginx 没得用怎么办?

```
PC     => Nginx / Apache => Jetty / Tomcat
Mobile => ?              => Jetty / Tomcat
```

自有协议，自研服务

- 参考 SIP 协议，http://tools.ietf.org/html/rfc3261
- JDK7 + Netty 4 = LWS 1.0

Nginx Config

```
http {
  index index.html;


  server {
    listen 80 defalut_server;
    server_name _;
    access_log logs/default.access.log main;
    server_name_in_redirect off;


    root /var/www/default/htdocs;
  }
}
```

LWS Proxy Config

https://github.com/typesafehub/config

```
proxy {
  uri = "tls://0.0.0.0:443"
  route {
    pre {
      /reg  = ["tcp://10.0.0.1:5902"]
      /http = ["tcp://10.0.0.1:5903"]
      /rpc  = ["tcp://10.0.0.1:5904"]
    }
  }
  http.white.list = ["/http/[^/]/internal/.*"]
}
```

Config is a DSL

Spray DSL

```
import spray.routing.SimpleRoutingApp

object Main extends App with SimpleRoutingApp {
  implicit val system = ActorSystem("my-system")

  startServer(interface = "localhost", port = 8080) {
    path("hello") {
      get {
        complete {
          <h1>Say hello to spray</h1>
        }
      }
    }
  }
}
```

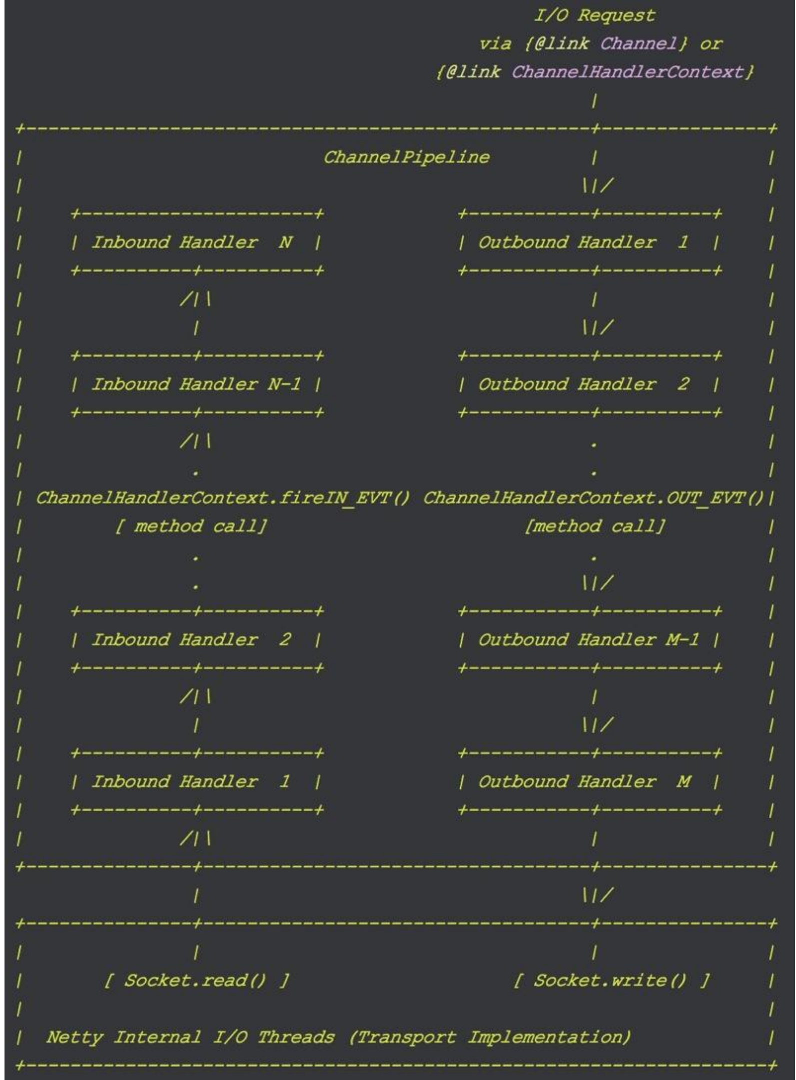
为什么不让配置成为代码的一部分呢？

仅仅是 DSL 还说不上 LEGO

Handler is brick

```java
protected void initChannel(Channel ch)
  throws Exception {

  final ChannelPipeline pl = ch.pipeline();
  pl.addLast(new MessageDecoder());
  pl.addLast(new MessageEncoder());
  pl.addLast(new LogAccessHandler());
  pl.addLast(new ExchangeHandler());
}
```



```
                                             I/O Request
                                        via {@link Channel} or
                                    {@link ChannelHandlerContext}
                                                  |
+-------------------------------------------------+---------------+
|                           ChannelPipeline        |               |
|                                                 \|/              |
|    +---------------------+            +-----------+----------+    |
|    | Inbound Handler  N  |            | Outbound Handler  1  |    |
|    +----------+----------+            +-----------+----------+    |
|              /|\                                  |               |
|               |                                  \|/              |
|    +----------+----------+            +-----------+----------+    |
|    | Inbound Handler N-1 |            | Outbound Handler  2  |    |
|    +----------+----------+            +-----------+----------+    |
|              /|\                                  .               |
|               .                                   .               |
| ChannelHandlerContext.fireIN_EVT() ChannelHandlerContext.OUT_EVT()|
|        [ method call]                       [method call]         |
|               .                                   .               |
|               .                                  \|/              |
|    +----------+----------+            +-----------+----------+    |
|    | Inbound Handler  2  |            | Outbound Handler M-1 |    |
|    +----------+----------+            +-----------+----------+    |
|              /|\                                  |               |
|               |                                  \|/              |
|    +----------+----------+            +-----------+----------+    |
|    | Inbound Handler  1  |            | Outbound Handler  M  |    |
|    +----------+----------+            +-----------+----------+    |
|              /|\                                  |               |
+---------------+-----------------------------------+---------------+
                |                                  \|/
+---------------+-----------------------------------+---------------+
|               |                                   |               |
|       [ Socket.read() ]                    [ Socket.write() ]     |
|                                                                   |
|  Netty Internal I/O Threads (Transport Implementation)            |
+-------------------------------------------------------------------+
```

所以, Lego 是这样的...

一个最简单的 Ping-Pong 服务器

```scala
import lego.dsl._

new Server {
  def name = "ping-pong"


   tcp.bind(port = 12306) {
     Codec() <=> Handle {
       case Request(_, hs, _) => Response(200, hs)
     }
   }
}
```

Scala 在应用中的探索

sbt twitter akka spray

Domain-Specific Language

Eval https://github.com/twitter/util

```scala
import com.xxx.MyConfig

new MyConfig {
  val myValue = 1
  val myTime = 2.seconds
  val myStorage = 14.kilobytes
}
```

```scala
import com.xxx.MyConfig

val config = Eval[MyConfig](new File("config/Development.scala"))
```

Operators

```
tcp.bind(port = 5905) {
  Codec() <=> Handler {
    case Request(_, hs, _) => Response(200, hs)
  }
}
```

```
trait Stage {
  def <=>(right: Stage): Stage = ...
}
```

Operators

```
tcp.bind(port = 5905) {
  Codec() <=> Route {
    case Request(_, h, _) if h ?: ROUTE => direct_to(h :#: ROUTE)
  }
}
```

```
abstrace class Name(prefix: String) {
  def ?:(headers: List[String]) = ...
  def :#:(headers: List[String]) = ...
}


val ROUTE = new Name("route:") {}
```

Question: Why not wrapped class ?

```
tcp.bind(port = 5905) {
  Codec() <=> Route {
    case Request(_, h, _) if h ?: ROUTE => direct_to(h :#: ROUTE)
  }
}
```

```
implicit class Headers(lines: List[String]) {
  def ?(prefix: String) = ...
  def :#(prefix: String) = ...
}

val ROUTE = "route:"
```

Operators

```
"append remote query to received request" in {
  pipeline("10.0.0.1:12345" ~ "10.0.0.2:5902") >>> {
    """
      |LWP /xxx
      |via:tcp://10.0.0.1:12306
      |
      |
    """
  } ==> {
    """
      |LWP /xxx
      |via:tcp://10.0.0.1:12306?r=10.0.0.1:12345
      |
      |
    """
  }
}
```

```
implicit class Pair(a: String) {
  def ~(b: String) = (a, b)
}
```

```
implicit class PipelineD(s: Stage) {
  def >>>(read: Frame) = {...}
  def ==>(expect: Frame) = {...}
}
```

String Interpolator

```
tcp.bind(port = 5905) {
  codec <=> Route {
    case Request(r"/http/.+", _, _) => bbl_lwp_http
  }
}

def insert_from: List[String] => List[String] =
  headers => headers :?: TOKEN map {
    case v @ r"""[^_]+_(.+)$uid""" => FROM -> s"$uid $v" :: headers
  } getOrElse headers
```

```
implicit class RegexContext(val sc: StringContext) extends AnyVal {
  def r = new Regex(sc.parts.mkString, sc.parts.tail.map(_ => "x"): _*)
}
```

WARNING !!!

```
scala> "123.cm" matches ".+\\.cm"
res1: Boolean = true

scala> "123.cm" matches ".+\.cm"
<console>:1: error: invalid escape character
       "123.cm" matches ".+\.cm"
```

```
scala> "123.cm" match { case s @ r".+\\.cm" => s; case s => "Ooops" }
res2: String = Ooops

scala> "123.cm" match { case s @ r".+\.cm" => s; case s => "Ooops" }
res3: String = 123.cm
```

Companion object

```
def filter_header = FilterHeader {
  case <<<(Request(u, hs, _))                => (insert_host(u) andThen insert_from)(hs)
  case >>>(Response(_, hs, _)) if hs ?: UID => hs :-: UID
}
```

```
class FilterHeader(g: PartialFunction[Direction, List[String]]) extends Stage {...}

object FilterHeader {
  sealed trait Direction
  case class >>>(frame: Frame) extends Direction
  case class <<<(frame: Frame) extends Direction

  def apply(g: PartialFunction[Direction, List[String]]) = new FilterHeader(g)
}
```

Funcational Style

Either & For-Comprehension

```scala
def uid(hs: List[String]) = {
  (hs :?: UID) match {
    case Some(uid) => uid
    case None      =>
      (hs :?: TO) match {
        case Some(o) => o.split(' ')(0)
        case None      =>
          (hs :?: TOKEN) match {
            case Some(t) => t.split('_')(1)
            case None => "-"
          }
      }
  }
}
```

```scala
implicit
def e[T]: Option[T] => Either[Unit, T] =
  _.toRight(())

def awk(c: Char)(i: Int) =
  (_: String).split(c)(i)

def uid(hs: List[String]) = (for {
  _ <- (hs :?: UID).left
  _ <- (hs :?: TO map awk(' ')(0)).left
  _ <- (hs :?: TOKEN map awk('_')(1)).left
  _ <- Right("-").left
} yield {}).right.get
```

Code Reuse

Trait or Object ?

```scala
trait T {
  def put(a: Any) {
    println(a)
  }
}

class A extends T {
  def hi() {
    put("hi")
  }
}
```

```scala
object O {
  def put(a: Any) {
    println(a)
  }
}

class B {
  import O._

  def hi() {
    put("hi")
  }
}
```

Trait or Object ?

```scala
trait A {
  case class B(i: Int)
}

class C extends A {
  def !(a:Any) = a match {
    case B(0) => println("B(0)")
    case b: B => println("B")
    case x    => println(s"Oops, $x")
  }
}

class D extends A {
  new C ! B(0)
}

new D // Oops, B(0)
```

```scala
object A {
  case class B(i: Int)
}

class C  {
  import A._
  def !(a:Any) = a match {
    case B(0) => println("B(0)")
    case b: B => println("B")
    case x    => println(s"Oops, $x")
  }
}

import A._
new C ! B(0) // B(0)
```

Trait, no Object !

```
trait T {
  def size: Int
  def isEmpty = size == 0
}

class A extends T {
  def size = 5
}

new A().isEmpty // false
```

Trait & Object !

```
package scala.collection.convert

trait WrapAsScala {
  import Wrappers._
  implicit def asScalaIterator[A](it: ju.Iterator[A]): Iterator[A] = ...
  implicit def enumerationAsScalaIterator[A](i: ju.Enumeration[A]): Iterator[A] = ...
}

object JavaConversions extends WrapAsScala with ...
```

Alias

Type

```scala
type Frame     = (StartLine, Headers, Option[Payload])


type StartLine = String
type Headers   = List[String]
type Payload   = (Array[Byte], Zip)
type Zip       = Boolean
```

```scala
case class Frame(startLine: String, headers: List[String], content: Option[Payload])

case class Payload(data: Array[Byte], zip: Boolean)
```

WARNING !!!

```
scala> type Headers = List[String]
defined type alias Headers

scala> :pas
List(1, 2, 3) match {
  case _: Headers => println("wrong!")
  case _          => println("right!")
}

<console>:10: warning: fruitless type test: a value of type List[Int] cannot also
be a List[String] (the underlying of Headers) (but still might match its erasure)
                case _: Headers => println("wrong!")
                        ^
wrong!
```

Val

```
scala> val Headers = List
Headers: scala.collection.immutable.List.type = scala.collection.immutable.
List$@7be117eb

scala> val headers = Headers("mid:1")
headers: List[String] = List(mid:1)

scala> val Headers(a) = List("mid:1")
a: String = mid:1
```

Val & Type

```
package object transport  {
  type Event = pipeline.Event
  type Command = pipeline.Command

  type Write = pipeline.Write
  val Write = pipeline.Write

  type Read = pipeline.Read
  val Read = pipeline.Read

  type Stage = pipeline.Stage[Context]
}
```

Actor Traps

Default Supervisor Strategy

```scala
import akka.actor._

class Handler(var i: Int) extends Actor {
  def receive = {
    case "incr" => i += 1
    case "show" => println(i)
    case "boom" => throw new Exception
  }
}

object Handler {
  def props(i: Int) = {
    Props(classOf[Handler], i)
  }
}
```

```scala
val system = ActorSystem("fun")
val h = system.actorOf(Handler.props(1))

h ! "show"  // 1
h ! "incr"
h ! "show"  // 2
h ! "boom"  // Exception

h ! "show"  // 1
```

WARNING !!!

```
class Connector(remote: Address) extends Actor {
  import context.system

  IO(Tcp) ! Tcp.Connect(remote) // be careful


  def receive = {
    case Tcp.Connected(_, locale) =>
        // handling
    case Tcp.CommandFailed(_: Tcp.Connect) =>
        context stop self
  }
}
```

WARNING !!!

```
class Connector(remote: Address) extends Actor {
  import context.system

  IO(Tcp) ! Tcp.Connect(remote)

  def receive = {...}

  // not work
  override def supervisorStrategy = SupervisorStrategy.stoppingStrategy
}
```

Solution

```
class ConnectingSupervisor extends Actor { // solution 1
  def receive = {
    case remote: Address =>
      val c = context.actorOf(Connector.props(remote))
      ...
  }
  override def supervisorStrategy = SupervisorStrategy.stoppingStrategy
}
```

```
class Connector(remote: Address) extends Actor { // solution 2
  override def preRestart(reason: Throwable, message: Option[Any]) = {
    context stop self
  }
}
```

No Sender

```
class Pong extends Actor {
  def receive = {
    case "ping" => sender() ! "pong"
  }
}
```

```
class Ping(pong: ActorRef) extends Actor {
  pong ! "ping"
  def receive = ...
}

class Ping(pong: ActorRef) {
  pong ! "ping"
}
```

```
def !(message: Any)(implicit sender: ActorRef = Actor.noSender): Unit
```

TCP DEBUG

```
// application.conf
io {
  loggers = ["akka.event.slf4j.Slf4jLogger"]
  loglevel = "DEBUG"

  tcp {
    trace-logging = on
  }
}
```

SBT

github.com/sbt/sbt-native-packager

```
src/universal
├── bin
│   ├── start
│   └── stop
└── conf
    ├── logback.xml
    └── proxy.scala
```

```
target/universal
└── stage
    ├── bin
    ├── conf
    ├── lib
    └── logs
```

```
> universal:package
packageBin           packageOsxDmg        packageXzTarball     packageZipTarball
```

Mirror Respository - Artifactory

```
[repositories]
  local
  sbt: http://mirror:8081/artifactory/sbt/,[organization]/[module]/
  (scala_[scalaVersion]/)(sbt_[sbtVersion]/)[revision]/[type]s/
  [artifact](-[classifier]).[ext]
  sbt-plugins: http://mirror:8081/artifactory/sbt-plugins/,[organization]/
  [module]/(scala_[scalaVersion]/)(sbt_[sbtVersion]/)[revision]/[type]s/
  [artifact](-[classifier]).[ext]
  scala: http://mirror:8081/artifactory/repo/
```

```
> sbt -Dsbt.repository.config=.repos clean test
```

```
curl get.jenv.io | bash
```

@linux_china

布道中的反思

幸运的是

- 一个有力的支持者
- 一块荒芜的新领域

艰难险阻

- 组织架构
- 团队基因