From Monolith to Microservices with lagom

In Scala

An Introduction

Steve Swing

@sswing

Lagom Framework

- The Reactive Microservices Framework
- “The Opinionated Microservices Framework for moving away from the Monolith”
- Lagom — Swedish word for 'just right’
- Microservices Just Right

About What Is Lagom Opinionated?

- Requires Java 8
- Message Driven
- Asynchronous Request/Response, Circuit Breaker
- Streaming
- Distributed Persistence, Event Sourcing/CQRS, Cassandra, Kafka, RDBMS
- Domain-Driven Design
- Immutability commands/messages/entities/collections

Origins

- Open Source -- Licensed under the Apache License, Version 2.0
- https://github.com/lagom/lagom.git
- Earliest public commits March 2016
- Implemented in Scala - Java & Scala API
- Maven support added August 2016
- Current Stable Release version 1.3.1
- Scala API Released with version 1.3.0

Technologies (1.3.1)

- Scala — 2.11.8
- Java — 8
- Play! Framework — 2.5.10
- Akka Streams, Clustering, & Persistence — 2.4.17
- Cassandra — 3.0.9
- Guice — 4.0
- Guava — 19.0
- Jackson — 2.7.8
- Kafka — 0.10.0.1
- Netty — 4.0.40.Final
- ScalaTest — 3.0.1
- sbt — 0.13.13
- Maven — 3.3.9

Features

- Optimized for rapid development
  - Embedded container
  - Hot code reloading
  - Automatic Service Discovery
  - Activator Templates
  - Maven Archetypes
  - sbt & Maven runAll commands
- No application server container means no fighting with classpath and classloaders

“Any sufficiently advanced technology is indistinguishable from magic.”

– Arthur C. Clarke

“All magic comes with a price, dearie!”

— Rumplestiltskin

Lagom Project Structure

```
├── front-end
├── hello-api
├── hello-impl
├── hello-stream-api
├── hello-stream-impl
├── integration-tests
└── project
```

Lagom API: Writing Services

Service Interface

```
trait ServiceCall[Request, Response] {
  def invoke(Request request): Future[Response]
  def invoke()(implicit evidence: Request =:= NotUsed): Future[Response] =
    this.asInstanceOf[ServiceCall[NotUsed, Response]].invoke(NotUsed)
}
```

Lagom API: Writing Services

Service Descriptors

```scala
import akka.{Done, NotUsed}
import com.lightbend.lagom.scaladsl.api.{Service, ServiceCall}
import play.api.libs.json.{Format, Json};

trait HelloService extends Service {
  def hello(id: String): ServiceCall[NotUsed, String]
  def useGreeting(id: String): ServiceCall[GreetingMessage, String]

  override final def descriptor = {
    import Service._
    named(“hello”).withCalls(
      pathCall("/api/hello/:id", hello _)
      pathCall("/api/hello/:id", useGreeting _)).withAutoAcl(true)
  }
}
case class GreetingMessage(message: String)
```

Lagom API: Writing Services

Implementing Services

```scala
package com.example.hello.impl

import com.lightbend.lagom.scaladsl.api.ServiceCall
import com.lightbend.lagom.scaladsl.persistence.PersistentEntityRegistry
import com.example.hello.api.HelloService

class HelloServiceImpl(registry: PersistentEntityRegistry) extends HelloService {
  override def hello(id: String) = ServiceCall { _ =>
    val ref = registry.refFor[HelloEntity](id)
    ref.ask(Hello(id, None))
  }
  override def useGreeting(id: String) = ServiceCall { request =>
    val ref = registry.refFor[HelloEntity](id)
    ref.ask(UseGreetingMessage(request.message))
  }
}
```

Lagom API: Writing Services

Consuming Services: Binding a Service Client

```scala
package com.example.hello.impl

import com.lightbend.lagom.scaladsl.server._
import play.api.libs.ws.ahc.AhcWSComponents
import com.example.hello.api.HelloService

abstract class MyApplication(ctx: LagomApplicationContext)
  extends LagomApplication(ctx)
    with AhcWSComponents {

  override lazy val lagomServer = LagomServer.forServices(
    bindService[HelloService].to(wire[HelloServiceImpl])
  )
}
```

Lagom API: Writing Services

Consuming Services: Using a Service Client

```
class MyServiceImpl(helloService: HelloService)
  (implicit ec: ExecutionContext) extends MyService {

  override def sayHelloLagom = ServiceCall { _ =>
    val result: Future[String] = helloService.sayHello.invoke(“Lagom")
    result.map { response =>
      s"Hello service said: $response"
  }
}
```

Lagom API: Writing Services

Testing Services

```
class HelloServiceSpec extends AsyncWordSpec with Matchers with BeforeAndAfterAll {
  "Hello service" should {
    "say hello" in {
      client.hello("Alice").invoke().map { answer =>
        answer should ===("Hello, Alice!")
      }
    }

    "allow responding with a custom message" in {
      for {
        _ <- client.useGreeting("Bob").invoke(GreetingMessage("Hi"))
        answer <- client.hello("Bob").invoke()
      } yield {
        answer should ===("Hi, Bob!")
      }
    }
  }
}
```

Lagom API: Writing persistent & clustered services

Persistent Entity (Stub)

```
import com.lightbend.lagom.scaladsl.persistence.PersistentEntity

final class Post1 extends PersistentEntity {
  override type Command = BlogCommand
  override type Event = BlogEvent
  override type State = BlogState

  override def initialState: BlogState = BlogState.empty

  override def behavior: Behavior = Actions()
}
```

Lagom API: Writing persistent & clustered services

Persistent Read-Side

```
class BlogServiceImpl(cassandraSession: CassandraSession) extends BlogService {
 override def getPostSummaries() = ServiceCall { request =>
    val response: Source[PostSummary, NotUsed] =
      cassandraSession.select("SELECT id, title FROM blogsummary")
        .map(row => PostSummary(row.getString("id"), row.getString("title")))
    Future.successful(response)
  }
}
```

Getting Started

sbt Giter8 Template

```
$ sbt new -Dsbt.version=0.13.13 lagom/lagom-scala.g8
```

Getting Started

Activator (sbt)

```
$ activator new my-first-system lagom-java

$ activator new twitter-clone lagom-java-chirper
```

Getting Started

Maven

```
$ mvn archetype:generate -DarchetypeGroupId=com.lightbend.lagom \
  -DarchetypeArtifactId=maven-archetype-lagom-java -DarchetypeVersion=1.3.1
```

Example Projects

Resources

- **Martin Fowler**
  - Microservices • GOTO 2014 • Jan 15, 2015
  - https://www.youtube.com/watch?v=wgdBVIX9ifA
  - http://martinfowler.com/bliki/CQRS.html

Resources

- **Greg Young**
  - A Decade of DDD, CQRS, Event Sourcing
  - https://www.youtube.com/watch?v=LDW0QWie21s
  - “CQRS/ES is not a top-level architecture”
  - “Event Sourcing fits really well with the functional programming model. It does not fit well with object-oriented [imperative] model.”

Resources

- **Eric Evans**
  - DDD & Microservices: At Last, Some Boundaries! • GOTO 2015
  - https://www.youtube.com/watch?v=yPvef9R3k-M

Resources

- **Yannick De Turck**
  - Lagom in Practice
  - https://youtu.be/JOGlZzY6ycI
  - https://github.com/yannickdeturck/lagom-shop

Resources

- **Jonas Bonér**
  - Co-founder and CTO of Lightbend, inventor of the Akka project, co-author of the Reactive Manifesto and a Java Champion.
  - Free eBook: “Reactive Microservices Architecture — Design Principles for Distributed Systems.” 2016 O’Reilly Media, Inc.
  - https://www.lightbend.com/blog/reactive-microservices-architecture-free-oreilly-report-by-lightbend-cto-jonas-boner

Resources

- **Markus Eisele**
    - Lightbend Java Developer Advocate, Java Champion, & Former Java EE Spec lead.
    - Free eBook: “Developing Reactive Microservices: Enterprise Implementation in Java”, 2016 O’Reilly Media, Inc.
    - https://www.lightbend.com/blog/developing-reactive-microservices-free-oreilly-mini-book-by-java-champion-markus-eisele
    - https://github.com/lagom/activator-lagom-cargotracker

Resources

- **James Roper**
  - Lightbend Play! Framework Lead Developer
  - https://www.lagomframework.com/
  - http://www.lightbend.com/lagom