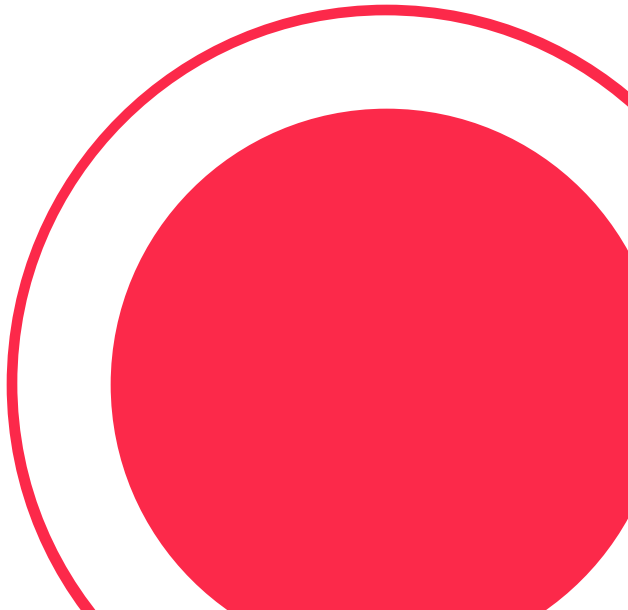
Building blocks for backend @

iHeartRADIO



Taiwan Scala Meetup by Victor Chan

Introduction

Intro to myself

- Siu Leung Chan 陳肇良 (Victor)
- Senior Software Engineer @ iHeartRadio
- Building Scala API Backend for 4 years

1. About our team, the radio music backend
2. Open source projects
3. Make it a platform
4. Q & A

Numbers

Fact Sheet for iHeartRadio

- 100 million registered users
- 90 platforms
- 3 countries
- 2 subscription services
- 2,000 live radio stations
- 1 million artists
- 30 million songs
- thousands of podcasts
- over a billion app downloads/updates
- 85 million social followers across our network
- quarter billion monthly broadcast listeners outside of our digital streams

Our backend team

- Team of 10
- ~ 70 github repository, 50 of the are mainly in Scala, i.e., micro-service, Kafka Consumer Jobs, recurrent Jobs, architectural service, e.g. Orchestration tools, DNS, Giter8 template, Sbt plugins, Load test, Monitoring tools, …..etc
- 8+ data source, e.g. multiple Mongo cluster, Postgre, Couchbase, Cassandra, Elastic, MSSQL, Kafka…etc
- Technology stacks – ELK, Akka, Kubernetes, Prometheus
- FP – cats, shapeless, internal libs

Open-sourced

Play-swagger (https://github.com/iheartradio/play-swagger)

A library that generates swagger specs from route files and case class reflection, no code annotation needed

Fully automated when the gateway service is up

Tremendous success on testing/QA purposes and acting as bridge between our client and product team

Play-swagger (https://github.com/iheartradio/play-swagger)

```
###
# summary: create a card
# responses:
# 200:
# description: success
# schema:
# $ref: '#/definitions/com.iheart.api.Protocol.CardCreated'
###
POST /users/:profileId/contexts/:contextName/cards
controllers.api.Cards.createCard(profileId: Int, contextName: Option[String])
```

Happy-path (https://github.com/iheartradio/happy-path)

FutureEither[T] is a monad transformer to unifies Future[T],
Future[Option[T]], Future[Either[U, T]], Future[Try[T]],
Future[Boolean]

Right biased => Left (error), right (value)

Error would expands to an ADT of Reason, e.g. ValidationReason,
ExceptionReason, RegularReason, ItemNotFound…etc

Happy-path (https://github.com/iheartradio/happy-path)

However, it is *just* deprecated right before this demo

Could be replace by cats.data.EitherT
https://typelevel.org/cats/api/cats/data/EitherT.html

Ficus (https://github.com/iheartradio/ficus)

Ficus is a lightweight companion to Typesafe config that makes it more Scala-friendly.

Ficus adds an as[A] method to a normal Typesafe Config so you can do things like config.as[Option[Int]], config.as[List[String]], or even config.as[MyClass]. It is implemented with type classes so that it is easily extensible and many silly mistakes can be caught by the compiler.

Henkan (https://github.com/kailuowang/henkan)

A small shapless library for converting between case classes.

Henkan.convert

Data migration ->
Case class Book(id: Int, name: String)
Case class BookDAO(id: Int, name: String, author = String, publishDate: DateTime)

Val book = …
Val bookDao = …

Book.to[BookDAO].set(author = “JK Rowling”, publishDate = …

Henkan (https://github.com/kailuowang/henkan)

Henkan.Optional

Use Protocol case class

Primary use case: protobuf with backward compatibility

More projects

https://github.com/iheartradio/kanaloa

https://github.com/iheartradio/asobu

Make a platform

Config tools

Supplementary of Typesafe config

Simplify config overriding and decide a final verson of the config that can be viewed by developer

Have every system takes config from config server as highest priority which persist the data on s3 or other platform

0 Confusion for production config generation

Sbt plugins

Help team member to generalize sbt project (not limited to Scala projects)

Provide certain command line helper, e.g. release docker, data migration…etc

To control versioning dependency

Blessed Plugins, e.g.
Scoverage, MIMA, Wartremover, Scalariform, NativePackger, BuildInfo …etc

And unify scalac Option:
https://tpolecat.github.io/2017/04/25/scalac-flags.html

Monitoring

Required tool for micro-services system

We use Prometheus

Thin wrapper that smooth out all the underlying Java Exception

Tag with different kind of reporting stat

Unit Test

Favor Unit test over Integration test

Help to smooth PR and release process by unit testing every feature!

Heavily use ScalaMock and Mockito for any test require external connection, e.g. db or http call

ScalaTest and ScalaCheck

Easily identify product requirement

Orchestra tools

We migrated form Akka Clustering to Kubenetes, allow us to move the complixity from application level to system level

The abstraction could let developer to more focus on business logic

Trend to merge with the architecture of the other team for Ops and SRE

FP

Not required but fun to work on ☺

Cats, Kittens, shapeless, …etc

Easy to use concept: Kleisli, NonEmptyList, EitherT, StateT, Validated….etc

Jenkins DSL (Bonus)

Jenkins is the central piece of platform automation

Jenkins DSL is the job that listen to a repo to create all the jobs

DSL are written in mostly Groovy and slightly Scala

Manage and document all code release, deployment, config generation, unit testing for any PR merge, load test, smoke test on production, recurrect Job

Q & A

https://github.com/joyfulvillage/TWScalaDemo2017

LinkedIn:
https://www.linkedin.com/in/victor-siu-leung-chan-0379424/

Twitter: @joyfulv