SCALA USER GROUP @ REWE DIGITAL

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

tim.steffens@rewe-digital.com ~ https://github.com/tmstff ~ https://rewe-digital.com/

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

WHAT TO EXPECT OF THIS TALK #1

- Lots of small frameworks and libraries for building microservices
- Focus: http communication (as server and client)
- Presumably very popular (and part of this talk):
  - play
  - akka-http
  - finatra or finch on top of finagle
  - http4s

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

WHAT TO EXPECT OF THIS TALK #2

- Implementation of sample application „image-cache“ in
  - play
  - akka-http
  - finatra on top of finagle
- Show code
- Share my experiences
- Share your experiences

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

PLEASE ...

- … interrupt me in case of questions
- … or if I'm talking bullshit
- I'd be happy to profit from your knowledge :-)

- Disclaimer
  - new to anything but play :-D
  - Last talk ~10-15 years ago at university - please don't expect perfection ;-)

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

SAMPLE APP „IMAGE-CACHE“

- API
  - GET image?url=http://nonsense.com/someimage.jpg
  - GET image/metadata?url=http://nonsense.com/someimage.jpg
- Image (well, or basically anything else) will be copied to database and retrieved from there once copied
- CouchDB: nice http interface, well suited to demonstrate async streams (AFAIK no relational DB client offers nonblocking I/O)
- CouchDB is started in Docker: `docker run -p 5984:5984 -d couchdb`

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

COUCH DB

- `curl http://127.0.0.1:5984/`
- `curl -X GET http://127.0.0.1:5984/_all_dbs`
- `curl -X PUT http://127.0.0.1:5984/new_db`
- `curl -H "Content-Type: application/json" -X PUT -d '{"url":"http://image.url"}' http://localhost:5984/new_db/xyz`
- `curl -X GET http://127.0.0.1:5984/new_db/xyz`
- `curl -H "Content-Type: text/plain" -X PUT -d 'some text' http://localhost:5984/new_db/xyz/attachment?rev=…`
- `curl -X GET http://127.0.0.1:5984/new_db/xyz/attachment`

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

THOUGHTS ON IMAGE-CACHE

- ID for CouchDB document is the url encoded as Base-64 (not recommended - long IDs do not perform well)
- content type is forwarded to CouchDB
- use async I/O & reactive streams (http://www.reactive-streams.org/) wherever possible
- use HTTP clients & JSON libs provided by framework wherever possible
- make sure DB is created on startup
- make code available on GitHub: https://github.com/tmstff/SUGC-rewe/
- ~~write tests in the way suggested by framework~~ (sorry, time was up …)

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

PLAY FRAMEWORK

- strongly opinionated, makes loads of decisions for you
- not just a lib, requires a sbt plugin to run
- based on Akka, supports nonblocking I/O and Reactive Streams
- provides nice routing-DSL (which needs to be pre-compiled to scala)
- comes with Play Json
- h2 database & browser out of the box
- HTML templating + assets + selenium based testing

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

PLAY FRAMEWORK SETUP & LINKS

- Create new play project
  - `activator new testProj play-scala *`

- Documentation: https://www.playframework.com/documentation/2.5.x/ScalaHome
- Github: https://github.com/playframework/playframework

- * https://www.lightbend.com/community/core-tools/activator-and-sbt

PLAY FRAMEWORK DEMO

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

PLAY FRAMEWORK +/-

- + hot reload
- + documentation & community (> 21k results for „play framework“ on Stackoverflow)
- + complete webapp up & running very quickly - including DB
- + comfortable JSON marshaling
- - no MDC integration out of the box
- - Reactive Streams integration a bit clumsy (for POSTing streams)
- - rather heavy-weight, probably more than you need

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

AKKA HTTP

- library rather than framework
- start & configuration of server completely manual, no „magic“
- continuation of http://spray.io , based on akka
- pure scala routing DSL
- built-in support for spray-json
- simple integration of Actors and Reactive Streams
- (mostly) symmetrical request/response model for client & server

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

AKKA-HTTP SETUP & LINKS

- Create new akka-http project
  - `activator new image-cache-akka-http akka-http-rest`
- Or simply include a dependency in you build.sbt
  - `"com.typesafe.akka" %% "akka-http-experimental" % "2.4.10"`

- Documentation: http://doc.akka.io/docs/akka/2.4.10/scala/http/
- Github: https://github.com/akka/akka/tree/master/akka-http

AKKA-HTTP DEMO

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

AKKA-HTTP +/-

- + very lightweight
- + nice integration of Actors and Reactive Streams
- + full control, no magic
- ~ the documentation is OK, but still I found it difficult to find the stuff I needed
- ~ the community seems to be not so big(< 3k results for „akka-http“ on Stackoverflow)
- - no MDC integration out of the box
- - huge dependency to akka for all of your code

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

FINAGLE

- RPC-system for the JVM
- more a library than a complete framework
- supports numerous protocols - including http
- claims to be super-fast
- symmetrical request/response model for client & server
- twitter implementation of Future, Promise, Try & Co
- API is rather low-level, there are wrappers like Finatra and Finch for a nicer DSL

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

FINATRA

- routing DSL for Finagle
- Logback MDC integration for contextual logging across futures
- simplified http client API
- Jackson JSON marshalling

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

FINATRA SETUP & LINKS

- Create new akka-http project
  - `activator new image-cache-finatra finatra-http-seed`
- Or simply include a dependency in you build.sbt
  - `"com.twitter" %% "finatra-http" % "2.1.6"`

- Documentation: https://twitter.github.io/finagle/
- Github: https://github.com/twitter/finagle

FINAGLE / FINATRA DEMO

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

FINATRA / FINAGLE +/-

- + very lightweight
- + Logback MDC integration for contextual logging across futures
- + variety of protocols
- - twitter implementation of Future, Promise, Try & Co <> scala pendant
- - documentation is … improvable!
- - community is tiny (~60 results for „Finatra“ on Stackoverflow, ~500 results for „Finagle“ on Stackoverflow)
- - no ansyc streaming
- - httpClient not suitable for accessing arbitrary URLs

PERFORMANCE TEST RESULTS

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

FINCH / FINAGLE

- routing DSL for Finagle
- Unterstützung diverser JSON libs
- tiny community (<10 Results for „finch finagle“ in Stackoverflow)
- no http client - only that of Finagle

- Dependency: `"com.github.finagle" %% "finch-core" % "0.11.0-M3"`
- github: https://github.com/finagle/finch

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

HTTP4S

- more a library than a complete framework
- based on scalaz
- async streams modelled as scalaz-streams
- tiny community (~30 Results for „http4s“ in Stackoverflow)

- Documentation: http://http4s.org/
- Github: https://github.com/http4s/http4s

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

WHAT ELSE?

- Other frameworks proposed by google:
  - http://www.scalatra.org/
  - http://unfiltered.databinder.net/Unfiltered.html
  - http://liftweb.net/
  - https://github.com/jdegoes/blueeyes
  - https://github.com/paypal/squbs
- Maybe part of a future talk :-)

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

CONCLUSIONS - FINAGLE

- Choose FINAGLE (e.g. with FINATRA) if
  - … you want „incredibly fast“
  - … you don’t care for programming comfort
  - … you don’t need a full web framework
  - … you don’t need Reactive Streams
  - … you don’t work with larger data

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

CONCLUSIONS - PLAY

- Choose PLAY if
  - … you want everything to work out of the box
  - … you want programming comfort
  - … your code does not focus on Actors and Reactive Streams
  - … you probably need a full web framework
  - … you probably work with relational databases
  - … you want great documentation and a huge community

WHICH MICROSERVICE FRAMEWORK TO CHOOSE?

CONCLUSIONS - AKKA-HTTP

- Choose AKKA-HTTP if
  - … your code focuses on Actors and Reactive Streams
  - … you want programming comfort
  - … you don’t need a full web framework
  - … you want a minimalistic approach and full control (no magic)
  - … you are willing to spend some time to configure your connections & pooling
  - … you are prepared to „marry“ Akka ;-)

THANK YOU!

PLEASE LEAVE YOUR FEEDBACK :-)