Ratpack

a toolkit for JVM web applications

dan.woods

Who am I?

NETFLIX

Gmail by Google danielpwoods@gmail.com

InfoQ #editor

/danveloper

@danveloper

Overview of Ratpack

- Originally inspired by Ruby Sinatra framework
- Has evolved into much, much more than just a microframework

```
@Grab("io.ratpack:ratpack-groovy:0.9.5")
import static ratpack.groovy.Groovy.*

ratpack {
  handlers {
    get {
      render "Hello world!"
    }
  }
}
```

Overview of Ratpack

- Core APIs are written in Java 7

- First class support for Groovy and Java 8
  - ✓ Makes use of recent groovy features like `@DelegatesTo` and static compilation for performance reasons

Built as a High Speed, Non-Blocking Web Framework

Application Startup Times in Milliseconds

Development-time Reloading w/ SpringLoaded

Ratpack is Your Gateway Drug to Non-blocking

Overview of Ratpack

- Provides app structure over Netty
- Gives familiar, high-level Web/HTTP constructs to developers
- All of the awesomeness of Netty, without the pain

The problem with non-blocking, async programming…

- JVM has no continuations, have to use callbacks
- Callbacks have no *temporal* relationship to the thread of execution
- Introduces non-deterministic executions cycles

Non-blocking, Asynchronous Execution Cycle

```
handler {
  // <1>
  promise { f ->
    // <2>
    Thread.start { f.success(true) }
  } then {
    // <3>
    doSomethingElse()
  }
  // <4>
}
```

Non-Determinism

(1) Definitely happens first

(2) Could happen before/after 3, 4

(3) Could happen before/after 2, 4

(4) Could happen before/after 2, 3

Ratpack: Non-blocking made easy

Executions

- Guarantee: calling execution segments are finished before callback returns are processed
- This provides execution determinism
- HUGELY important for application stability!
- Execution segments can pick up on different threads, but will not block during processing

Ratpack Non-blocking, Synchronous Execution Cycle

```
handler {
  // <1>
  promise { f ->
    // <3>
    Thread.start { f.success(true) }
  } then {
    // <4>
    doSomethingElse()
  }
  // <2>
}
```

Non-Determinism

(1) Definitely happens first

(2) Definitely happens second

(3) Definitely happens third

(4) Definitely happens fourth

Blocking Done Right.

- Any blocking operation can be scheduled to the background in a non-request-handling thread pool
- Scheduler returns a Promise
- Blocking ops can be scheduled with *RxJava*

```
handler {
  blocking { fulfiller ->
    fulfiller.success "done"
  } then { result ->
    context.render result
  }
}
```

Goal: Don't try to be everything!

- Convention over configuration taken too far doesn't allow you to wire things together
- Make wiring things together a painless, one-liner
- Framework components are composable
- Don't get in the way! Instead, scale in complexity.

Documentation as a Programmatic Construct

- Documentation annotations like @Nullable and @NonBlocking let you know very quickly the method's behavior
- Fully tested Javadocs!

Deep Dive Ratpack

The Groovy Script Approach

- Excellent for rapid prototyping
- Great to generate a basic API
- Wonderful for a slim micro-service
- Limited ability to customize and wire

A More Complex Application

- Proper support for HTTP verbs
- Support for prefixed routes
- Extract path tokens
- Serve static content
- Server-side rendering
- Bind & Inject services

Handlers & The Handler Chain

- Handlers are the “edge” of the application
- The chain is comprised of many handlers
- Handlers can exist in many forms
- A handler is a “core” object in Ratpack

The Registry

- Provides runtime component lookup
- Abstraction on the DI framework
- Components can be injected to handlers through varargs, or retrieved directly off of the context
- Ratpack Remote Control manipulates the registry for “hot” testing

Dependency Injection

- By default the Registry backs into a DI framework
- Guice by default, Spring registry available by Dave Syer from Pivotal
- First-class support for Guice modules

Wiring Services

- Directly, through the binding DSL
- Through a Guice Module
- Services can be used by the application, but also by Ratpack (such as Renderers)

Configuration

- Ratpack provides granular tuning of almost every aspect
- Lookup `ratpack.properties` next to `Ratpack.groovy`
- Extra properties available via `LaunchConfig#getOther`

Externalized Configuration

- This story will get much better. Today, you have to roll your own.
- https://github.com/ratpack/ratpack/issues/212
- https://github.com/ratpack/ratpack/issues/295

Testing

- This is a ***BREEZE*** in Ratpack!
- LocalScriptApplicationUnderTest and TestHttpClient provide convenient fixtures for Functional testing
- Application functional testing is a 1st class citizen in Ratpack applications
- Fantastic integration with Spock BDD framework

Testing - EmbeddedApplication

- EmbeddedApplication can be used to test different Guice modules, or subsets of functionality
- Can be used outside of Ratpack!
- Many embedded applications can be run concurrently within tests
- Interactions are the same as regular tests, via testHttpClient

Testing – Remote Control

- Remote control module allows you to influence a running application
- Can be used to bootstrap data in a Functional/Smoke Testing situation
- Great for Ratpack, because Ratpack is EXTREMELY fast!

Packaging & Distribution

- ./gradlew dist
  - Builds the project for distribution, including executable bash script!
- ./gradlew fatjar
  - Uses the Gradle Shadow Plugin to package a runnable JAR file
- These come for free with the Ratpack Gradle plugin!

Beyond the Basics

Many, Many Modules of Support!

- Authentication: **Pac4j**
- Server Side Templating: **Handlebars** & **Thymeleaf**
- Request Caching: **Hystrix**
- Reactive Programming: **RxJava** & **Reactor**
- Monitoring & Metrics: **New Relic** & **Codahale**
- Session Support w/ `ratpack-session`
- many, many more to come!

Roadmap

- Continue to release on the 1st of every month
- Plans to move to support minimum Java 8
  - Can't move here until Heroku supports it!
- No 1.0 Release until the APIs are fully locked down
  - This will be post-Java8
- Need to grow the User Manual and outside-of-code documentation

Roadmap

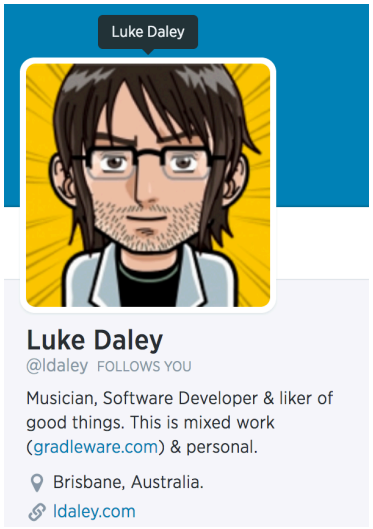
- Meanwhile… Ratpack is no longer “incubating”
- Very much an established framework – 40 worldwide contributors with over 2,000 commits!
- API can be feature-locked at a monthly-release version (0.9.5 right now)
- Growing community & excellent engagement from the core team!

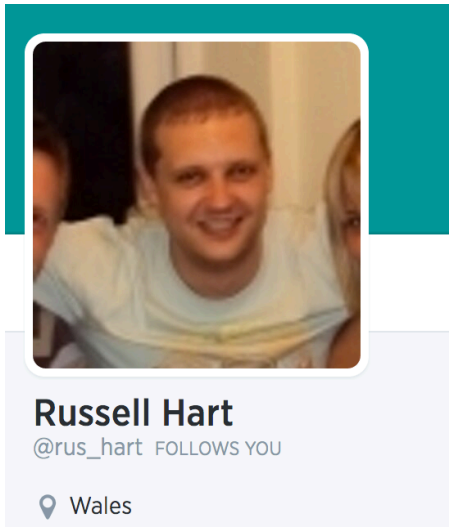
Contributing Back to Ratpack

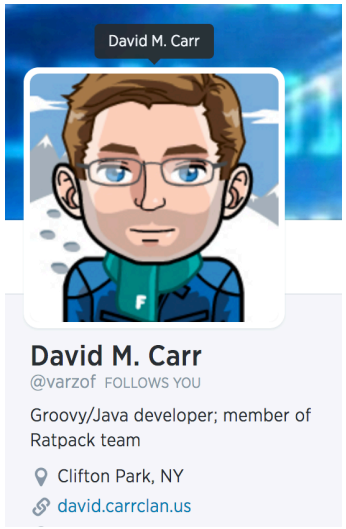
- ***USE IT!*** Let us know what we're missing, file issues.
- Help improve the user manual, provide example applications
- Always need really smart people to help take us to the next level
- ***REACH OUT TO US!***

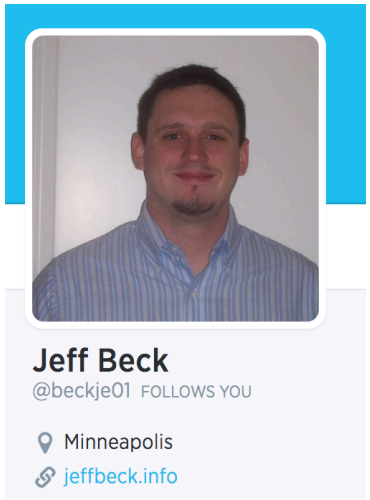
Contact us!

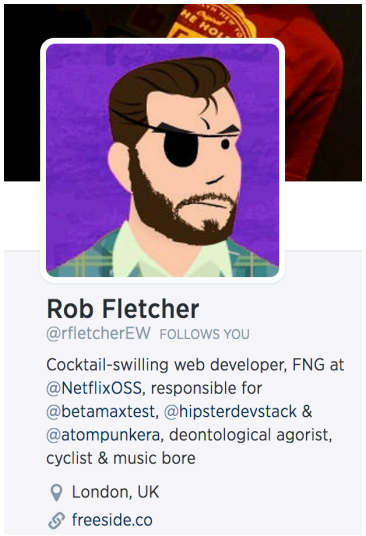
Dan Woods
Dan Woods
@danveloper
Cloud Developer at Netflix (@netflix); Ex-OPI'er (@objectpartners); InfoQ Java Editor (@infoq); Into all-things #groovylang, #grailsfw, #spring, & #java.
Silicon Valley, CA
danveloper.com


Luke Daley
Luke Daley
@ldaley FOLLOWS YOU
Musician, Software Developer & liker of good things. This is mixed work (gradleware.com) & personal.
Brisbane, Australia.
ldaley.com


Russell Hart
@rus_hart FOLLOWS YOU
Wales


David M. Carr
David M. Carr
@varzof FOLLOWS YOU
Groovy/Java developer; member of Ratpack team
Clifton Park, NY
david.carrclan.us


Jeff Beck
@beckje01 FOLLOWS YOU
Minneapolis
jeffbeck.info


Rob Fletcher
@rfletcherEW FOLLOWS YOU
Cocktail-swilling web developer, FNG at @NetflixOSS, responsible for @betamaxtest, @hipsterdevstack & @atompunkera, deontological agorist, cyclist & music bore
London, UK
freeside.co

Marcin Erdmann
@marcinerdmann FOLLOWS YOU
Groovy enthusiast
London
blog.proxerd.pl

A word from our sponsors…

NETFLIX

Come work with me on awesome DevOps stuff.

http://jobs.netflix.com