rapid web application development with

&

by Andrey Adamovich @codingandrey

About me

Andrey Adamovich

- Bio: Developer, coach, speaker, author
- Company: Aestas/IT (http://aestasit.com)
- E-mail: andrey@aestasit.com
- Linkedin: http://www.linkedin.com/in/andreyadamovich
- Lanyrd: http://lanyrd.com/profile/andrey-adamovich
- GitHub: https://github.com/aadamovich
- SO: http://stackoverflow.com/users/162792/andrey-adamovich
- Twitter: @codingandrey, @aestasit

Quick Start!

0. Prerequisites

- Install Java 8+
- Install Groovy 2.4+
- (Optionally) install Gradle 2+ (or just use Gradle Wrapper)

1. Type in...

```
01. @Grab("org.slf4j:slf4j-simple:1.7.10")
02. @Grab("io.ratpack:ratpack-groovy:0.9.11")
03. import static ratpack.groovy.Groovy.ratpack
```

1. Continue...

```
01. ratpack {
02.   handlers {
03.     get {
04.       response.send new Date().toString()
05.     }
06.   }
07. }
```

2. Save as...

```
01. ratpack.groovy
```

3. Start!

```
01. groovy ratpack.groovy
```

4. Enable some more logging

```
01. JAVA_OPTS=-Dgroovy.grape.report.downloads=true
```

Ratpack facts I

Ratpack is a toolset that combines several Java libraries that allows efficiently developing performant and testable HTTP applications.

Ratpack facts II

- Inspired by Sinatra framework
- Requires Java 8
- Does not require a container
- Does not implement Servlet API
- Goes under Apache 2.0 License

Ratpack facts III

- Core is very minimal and is only based on few abstractions (*Handler* and *Registry*)
- Many additional modules exist and it's easily to develop new ones
- Modules are injected through DI (there is not specialized plugin system)
- Out-of-the-box integration with Guice and Spring

Stack


Ratpack
A toolkit for JVM web applications


<your fav lib>
ratpack-*
ratpack-core
guice
netty

It's alive and very active!

Release history

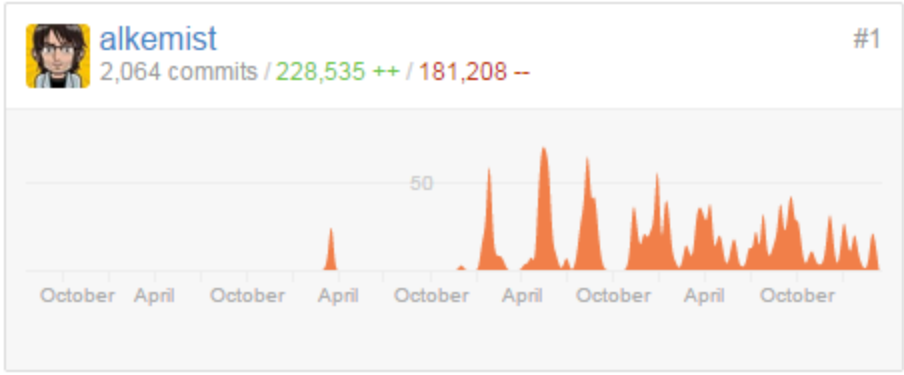
- 0.5.2 - Jul 21, 2012
- 0.6.1 - Nov 29, 2012
- 0.9.0 - Jan 02, 2014
- 0.9.1 - Feb 01, 2014
- 0.9.2 - Mar 01, 2014
- 0.9.3 - Apr 01, 2014
- 0.9.4 - May 01, 2014
- 0.9.5 - Jun 01, 2014
- 0.9.6 - Jul 01, 2014

Release history

- 0.9.7 - Aug 01, 2014
- 0.9.8 - Sep 01, 2014
- 0.9.9 - Oct 01, 2014
- 0.9.10 - Nov 02, 2014
- 0.9.11 - Dec 01, 2014
- 0.9.12 - Jan 01, 2014
- 0.9.13 - Feb 01, 2015
- 0.9.14 - Mar 01, 2015

Commit history

Heroes


alkemist
#1
2,064 commits / 228,535 ++ / 181,208 --
50
October
April
October
April
October
April
October
April
October

Modules I

- Asynch: reactor, rx
- Authentication: pac4j
- Build/Packaging: gradle
- Common: config, session
- Database: h2, hikari

Modules II

- Dependency Injection: guice, spring-boot
- JSON: jackson
- Language: groovy, kotlin
- Reliability: hystrix, codehale-metrics, newrelic
- Templates: handlebars, thymeleaf, groovy
- Testing: test, groovy-test

Java + Groovy = ?

- Has similar performance to Java when using `invokeDynamic`
- Supports static compilation and compile-time type checking
- Useful for defining rich DSLs with type checking via `Closure` parameters and `@DelegatesTo` annotations

IDE support

- IntelliJ IDEA recommended
- Eclipse has poor support for Groovy and @DelegatesTo
- NetBeans - haven't even tried

Diving deeper

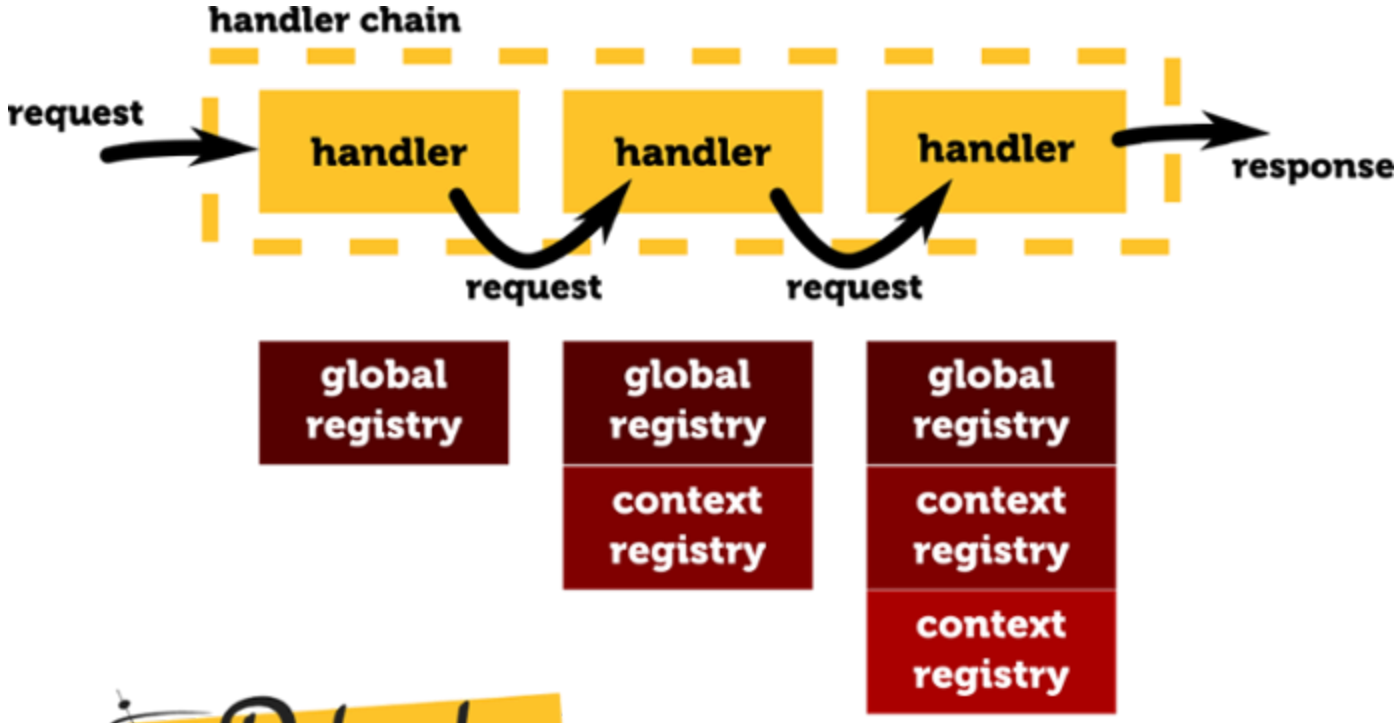
Handlers

- All request processing is done via composition of `Handler`s.
- Each `Handler` in the `Chain` is asked to respond to a `Request` until one actually does.

A handler can

- Send a `Response` to the `Request`.
- Delegate to the next `Handler` in the `Chain`.
- Insert `Handler`s into the `Chain` and immediately delegate to them.
- Change `Context`, which represents the current state of `Request` processing.

Flow



Ratpack
A toolkit for JVM web applications

Paths

```
01. prefix('api') {
02.   get('user/:id') {
03.     render getUser(pathTokens.id)
04.   }
05.   get('friends') {
06.     render getFriendList()
07.   }
08. }
```

Verbs

```
01. handler('user') {
02.   byMethod {
03.     get { ... }
04.     post { ... }
05.     put { ... }
06.     delete { ... }
07.   }
08. }
```

Content types

```
handler('user') {
  byContent {
    json { ... }
    xml { ... }
    type("application/vnd.app.org+json;v=1") {
      ...
    }
  }
}
```

Static content

```
01. assets "public"
02. assets "index.html"
```

Templates

Enough!

Demo 1

Demo 2

Summary

Take-aways

- Ratpack can be used to quickly prototype web APIs and applications.
- Learning curve is really small, you can start in seconds.
- It can be used to create high performance web applications due to non-blocking architecture.
- Ratpack does not lock you in the way you implement data access, session handling, logging, etc.
- Ratpack has vibrant community and actively evolving code base.

Reading material

- http://ratpack.io
- http://www.slideshare.net/search/slideshow?q=ratpack
- https://github.com/ratpack

Questions?

Thank you!

Happy coding!