Hands on Ratpack Workshop

Agenda

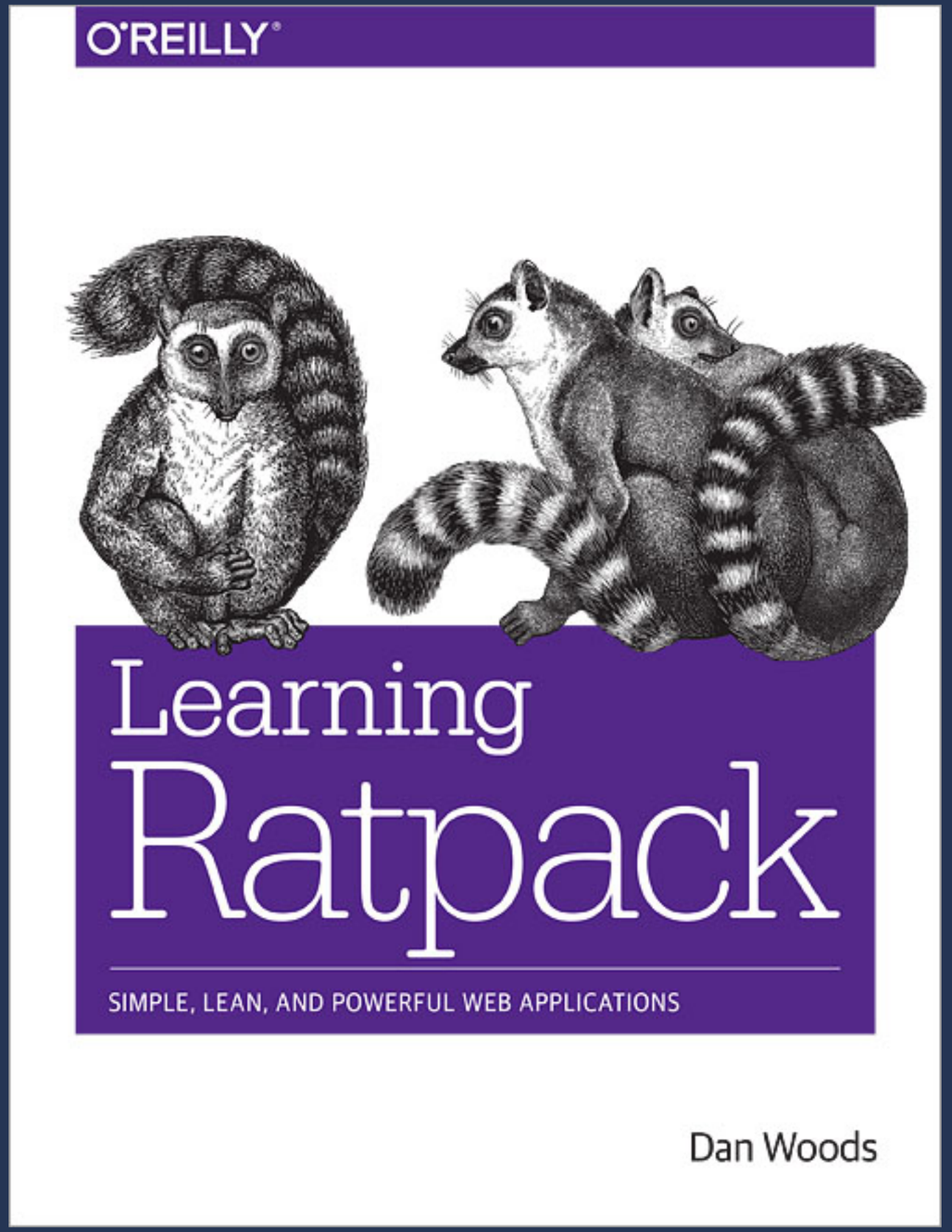
- Who am I?
- What is Ratpack?
- How the Labs are Structured
- Lab 01 - Handlers (75 min)
- Lab 02 - Handler Refactoring (60 min)
- Lab 03 - The Context (45 min)

Agenda

- Lab 04 - Google Guice Pt. 1 (45 min)
- Lab 05 - Templating and Rendering (60 min)
- Lab 06 - Security (60 min)
- Lab 07 - Blocking Database Operations (60 min)
- Resources to Learn More

Who Am I?

- Dan Woods
- GitHub: danveloper, Twitter: danveloper
- Cloud Infrastructure Engineer, Timshel.com
- Core member of the Ratpack team
- Author, Learning Ratpack - O'Reilly 2016



What is Ratpack?

Ratpack is a high-throughput, non-blocking, asynchronous, reactive web framework for the JVM.

Ratpack is designed for *performance* and *agility*.

An unopinionated framework for building web applications. Guides you to best practices and makes sensible things as easy as possible.

Ratpack's functional programming interfaces promote easy-to-test, stateless web applications. A great choice for microservices and production-critical applications.

An easy to use, fluent API for designing applications.

```
public class App {
  public static void main(String[] args) throws Exception {
    RatpackServer.start(spec -> spec
      .serverConfig(c -> c.baseDir(BaseDir.embedded()))
      .handlers(chain -> chain
        .get(ctx -> ctx.render("Hello World!"))
      )
    );
  }
}
```

Integration with Gradle to get up-and-running as easily as possible.

```
buildscript {
  repositories {
    jcenter()
  }
  dependencies {
    classpath 'io.ratpack:ratpack-gradle:1.4.4'
  }
}

apply plugin: 'io.ratpack.ratpack-java'

repositories {
  jcenter()
}
```

A composable set of libraries. Choose only what you need, nothing more. Keep your apps lightweight.

- RxJava
- Hystrix
- Hikari
- Retrofit2
- Dropwizard Metrics
- many, many more integrations...

Performance first: aims to fully utilize the resources given to it, small thread pool, 10s of 1000s req/sec on a single instance

Ratpack is *Java web framework*, written entirely for Java 8. Its internal API design allows DSLs to be built in other languages. Robust integration with Groovy to provide a seamless DSL.

```
import static ratpack.groovy.Groovy.ratpack

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

Developer centric: testability, packaging, development-time reloading

Why is this workshop in Java?

Ratpack's Java API provides a fundamental understanding of how applications are wired together. Groovy and other DSLs are decoration over the Java APIs. Understanding the Java API first will help you in understanding those higher-order DSLs.

What will you get out of this workshop?

- A hands-on understanding of how Ratpack applications are built
- Exposure to fundamental aspects of the framework
- A test-driven approach to building applications
- An understanding of how to learn Ratpack beyond the basics

How the Labs are Structured

- Clone the repo: https://github.com/danveloper/hands-on-ratpack
- Each lab has two directories
  - `lab01` has skeleton structure and failing tests you will work to make pass
  - `lab01-answer` has the example solution

How the Labs are Structured

- You will be introduced to each lab and its concepts at the beginning of each section
- Each lab has a markdown file with detailed notes
- Some comments in the tests themselves to help guide you
- Generous time has been allotted for each lab, so please feel free to ask questions and try things out!