Hands on Ratpack

Danny Hyun
Mindspark

Jeff Beck
SmartThings

Agenda

- Who are we?
- What is Ratpack
- How the Labs are Setup
- Lab 1 - Handlers
- Lab 2 - Handler Refactoring
- Lab 3 - The Context
- Lab 4 - Google Guice
- Lab 5 - Render
- Lab 6 - Security
- Resources to Learn More

Who are we?

What is Ratpack

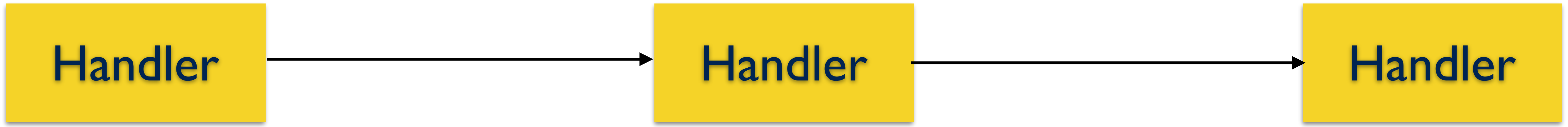
Ratpack is a set of Java libraries that facilitate fast, efficient, evolvable and well tested HTTP applications. It is built on Netty the event-driven networking engine and focused on Reactive principals.

How the Labs are Setup

- Clone the repo bit.ly/ratpackDevoxx
- Each Lab has two directories
  - *lab-01* has failing tests you need to make pass
  - *lab-01-answer* has the example solution
- We will introduce each lab and have some hints posted here
- Lots of details are in comments in the tests themselves
- Each lab has a markdown file with detailed notes

Lab 1 - Handlers

Handlers are the fundamental component of any Ratpack application, all request processing in Ratpack apps is done by composing a chain of handlers.



Lab 1 - Handlers Whats Covered

- Simple routing
- Routing by HTTP method
- Routing by HTTP header
- Grouping handlers with the same prefix
- Routing by regular expression
- Using path tokens
- Static assets

Lab 1 - Handler Sign Posts

Run the tests continuously:

```
./gradlew -t lab-01:test
```

- Handler Manual
- Important Classes
  - ratpack.handling.Chain
  - ratpack.handling.Context
  - ratpack.path.PathBinding

- Interesting Ratpack Specs
  - PathroutingSpec
  - PathAndMethodRoutingSpec
  - TokenPathBinderSpec

```
Line 1
Line 2




Line n
```

Lab 2 - Handler Refactor

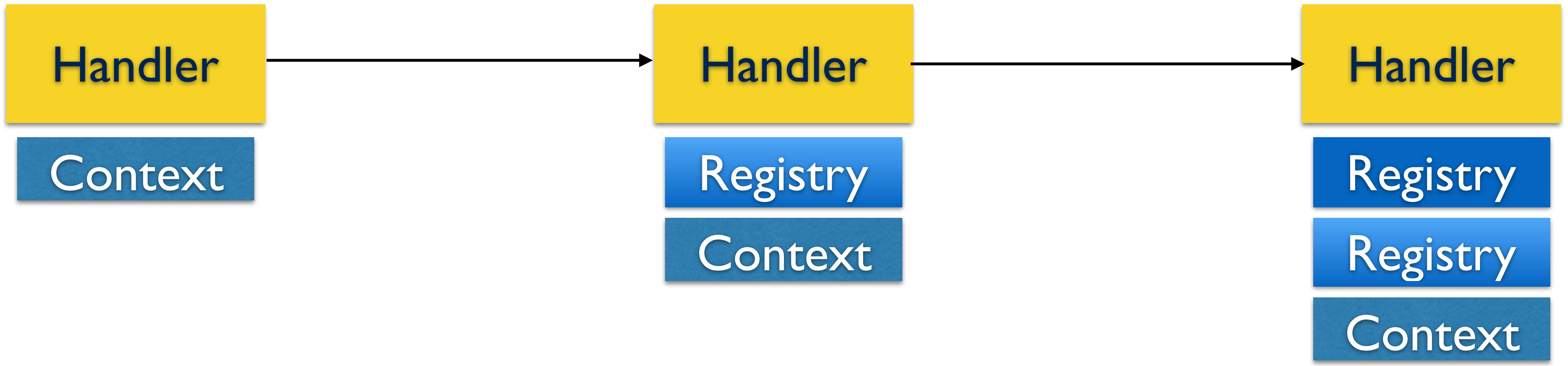
- Improve readability
- Allow handlers to be easily unit tested
- Share common handlers across applications
- Extend the handler DSL with your own shortcut methods

Lab 2 - Sign Posts

- ratpack.handling.Chain#prefix(prefix, action)
- ratpack.handling.Handler

Lab 3 - Context

The **Context** provides access to the HTTP Request and Response. More than that though, the Context is also a **Registry** of objects. It provides access to these contextual objects via type-lookup and allows arbitrary objects to be "pushed" into the context for use by downstream handlers.



Lab 3 - Context Whats Covered

- Registering objects in the registry
- Dynamically adding contextual objects
- Looking up contextual objects from the registry
- Injecting contextual objects into handlers

Lab 3 - Sign Posts

- Context Manual
- ratpack.handling.Context
- ratpack.registry.Registry

Lab 4 - Google Guice

Ratpack provides integration with Google Guice for dependency injection. Guice is of particular importance as additional Ratpack functionality is packaged up as Guice modules.

The Guice integration provides a Guice backed **Registry**. This means that any objects bound with Guice are available in the **Context**.

Lab 4 - Sign Posts

- ratpack.guice.BindingsSpec
- This time the hints are in the TODO within **Lab04.java**

Lab 5 - Render

Rat pack has the concept of **Renderer** which is responsible for rendering an object to the response. Ratpack provides many renderers out-of-the-box but you can also create your own renderers for custom objects.

Lab 5 - What is Covered

- Rendering String
- Rendering Handlebars Templates
- Rendering a custom object with a different content type depending on what has been requested

Lab 5 - Sign Posts

- ratpack.render.Renderer
- ratpack.jackson.Jackson

Lab 6 - Security

The Pac4j library is a powerful security framework. There is a Ratpack module that adapts the framework for use. Note that the module depends on the Session module.

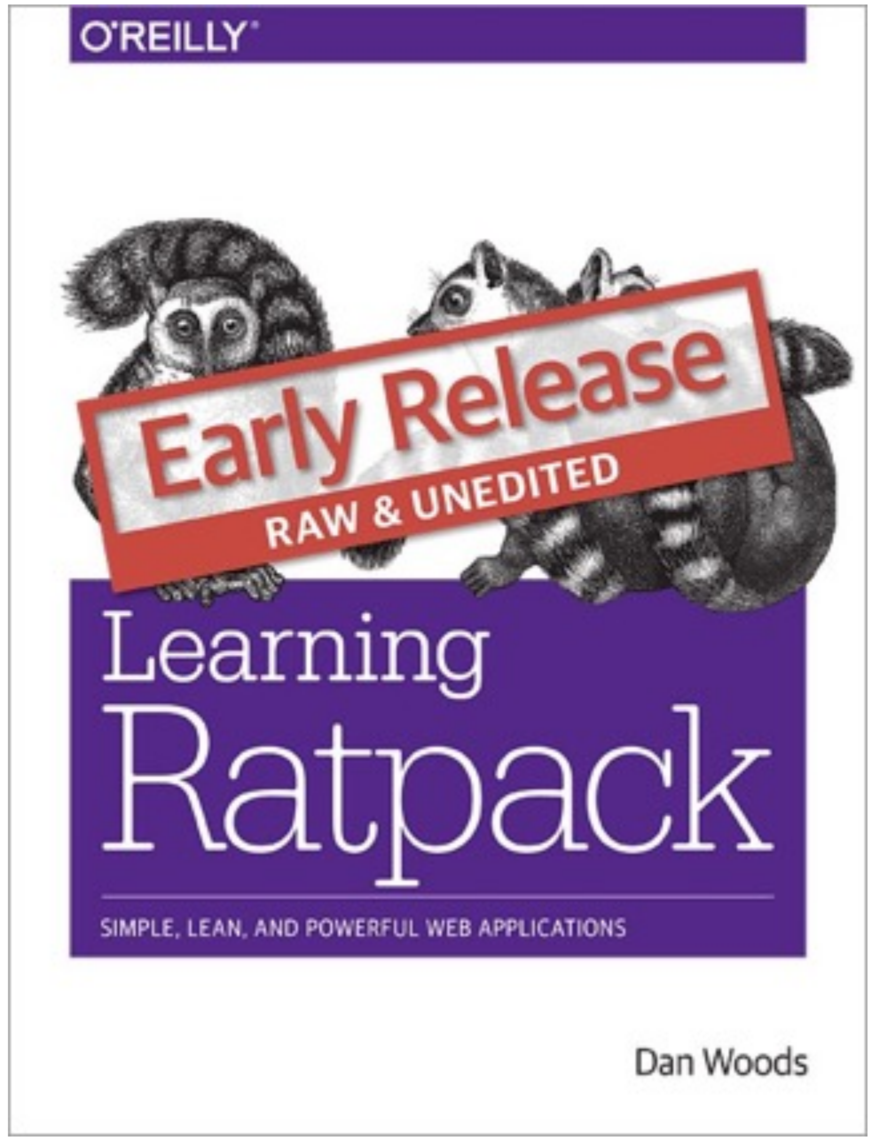
Lab 6 - Sign Posts

- Pac4j
- Pac4j-http
- ratpack.pac4j.RatpackPac4j
- BasicAuthClient

Resources to Learn More

- Website - http://ratpack.io
- Slack - http://slack-signup.ratpack.io/
- The Java Docs - http://ratpack.io/manual/current/api/index.html

Book Coming Soon


O'REILLY
Early Release
RAW & UNEDITED
Learning Ratpack
SIMPLE, LEAN, AND POWERFUL WEB APPLICATIONS
Dan Woods

http://bit.ly/ratpackBook