Web Development With Clojure

Ray Miller
Cambridge NonDysFunctional Programmers

Prerequisites

- leiningen
  - http://leiningen.org/#install
- git
  - https://git-scm.com/

Overview

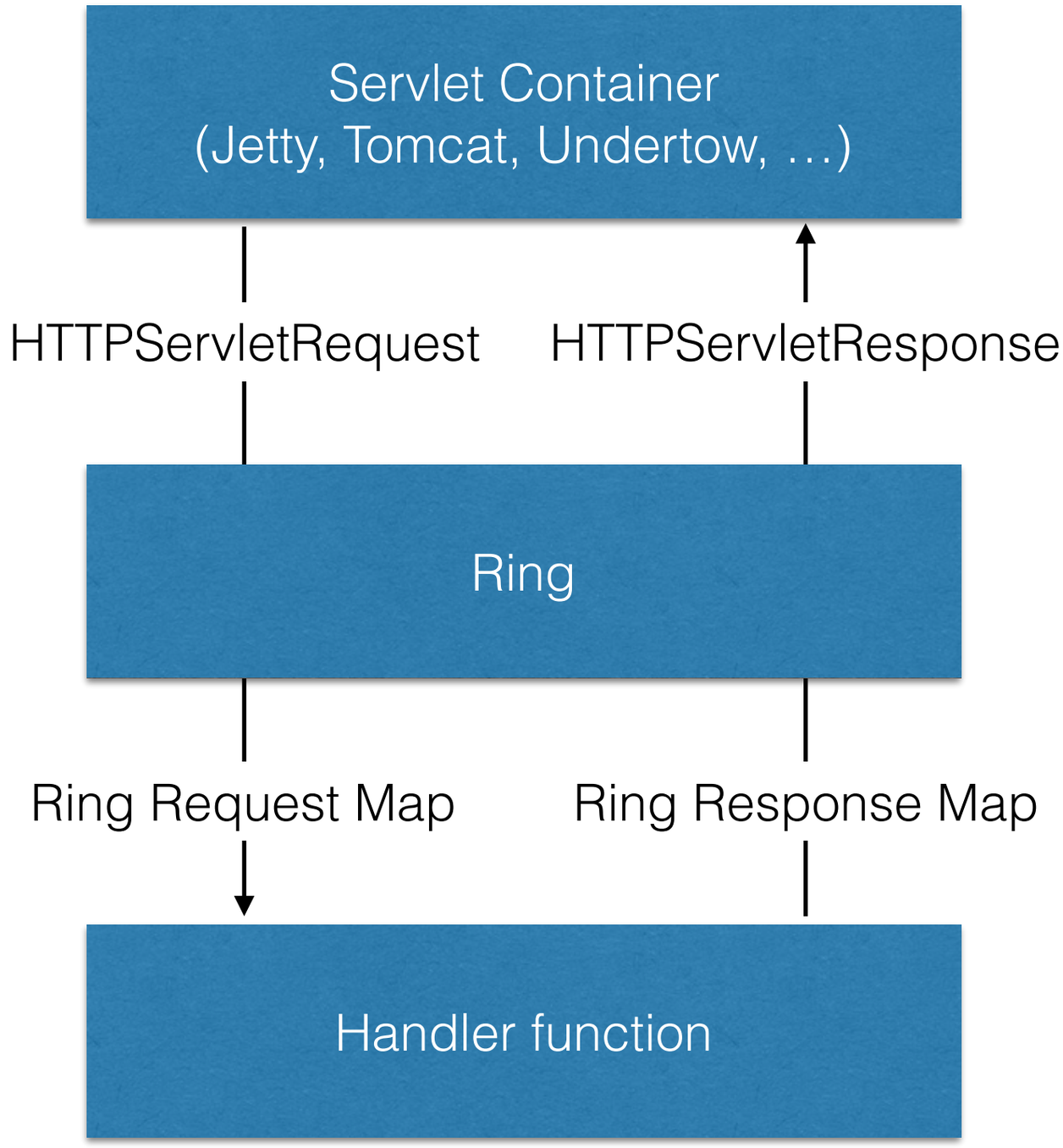
- Your first Ring application
- Routes, Requests, and Responses
- Hiccup
- Middleware
- Authentication
- Databases

The Demo App

- Create ads and upload images
- Randomly select ads
- Output JavaScript to embed your ads in a website
- Track click-throughs

Ring

- Clojure's equivalent of Ruby's Rack, Python's WSGI and Perl's Plack
- Deals with the Java Servlet API
- Provides utilities for parsing requests and generating responses
- Extensible through middleware


Servlet Container
(Jetty, Tomcat, Undertow, …)
HTTPServletRequest
HTTPServletResponse
Ring
Ring Request Map
Ring Response Map
Handler function

Our first Ring application

```
lein new compojure web-tutorial
cd web-tutorial
lein ring server
```

```
(ns web-tutorial.handler
  (:require [compojure.core :refer :all]
            [compojure.route :as route]
            [ring.middleware.defaults
               :refer [wrap-defaults site-defaults]]))

(defroutes app-routes
  (GET "/" [] "Hello World")
  (route/not-found "Not Found"))

(def app
  (wrap-defaults app-routes site-defaults))
```

The Request Map

```
{:dev {:dependencies [[javax.servlet/servlet-api "2.5"]
                      [ring/ring-mock “0.3.0”]
                      [ring/ring-devel “1.4.0”]]}}
```

```
(ns web-tutorial.handler
  (:require [compojure.core :refer :all]
            [compojure.route :as route]
            [ring.middleware.defaults
                :refer [wrap-defaults site-defaults]]
            [ring.handler.dump :refer [handle-dump]]))

(defroutes app-routes
  (GET "/" [] "Hello World")
  (GET "/dump/*" [] handle-dump)
  (route/not-found "Not Found"))

(def app
  (wrap-defaults app-routes site-defaults))
```

- Restart your Ring server
  - `CTRL-C` *(to stop the running server)*
  - `lein ring server`
- Visit these URLs
  - http://localhost:3000/dump/
  - http://localhost:3000/dump/this/that
  - http://localhost:3000/dump/this/that?q=1&r=2&r=3

The Response Map

```
lein repl
user=> (require '[ring.util.response :as r])
nil
user=> (r/response "Hello World")
{:status 200, :headers {}, :body "Hello World"}
```

```
user=> (-> (r/response "Not found”)
           (r/content-type “text/plain")
           (r/status 404))
{:status 404,
 :headers {"Content-Type" "text/plain"},
 :body "Not found"}
```

Exercise

- Read the API documentation:
  - https://ring-clojure.github.io/ring/ring.util.response.html
- In your REPL, generate a response with a Cache-Control header: `private, max-age=0, no-cache`
- Generate a redirect response to /login

The Ring Handler

A **ring handler** is simply a function that takes a request map and returns a response map

```
(ns web-tutorial.handler
  (:require [compojure.core :refer :all]
            [compojure.route :as route]
            [ring.handler.dump :refer [handle-dump]]
            [ring.util.response
                :refer [response content-type charset]]
            [ring.middleware.defaults
                :refer [wrap-defaults site-defaults]]))

(defn handle-hello
  [request]
  (let [who (get-in request [:params :who])]
    (-> (response (str "Hello " who))
        (content-type "text/plain")
        (charset "utf-8"))))

(defroutes app-routes
  (GET "/" [] "Hello World")
  (GET "/dump/*" [] handle-dump)
  (GET "/hello/:who" [] handle-hello)
  (route/not-found "Not Found"))
```

Hiccup

```
:dependencies [[org.clojure/clojure "1.7.0"]
               [compojure "1.4.0"]
               [ring/ring-defaults "0.1.5"]
               [hiccup "1.0.5"]]
```

```
lein repl
user=> (require '[hiccup.core :refer [html]])
nil
user=> (html [:div [:p "Welcome to hiccup!"]])
"<div><p>Welcome to hiccup!</p></div>"
```

```
user=> (html [:div#main.container [:p "Welcome to hiccup!"]])
"<div class=\"container\" id=\"main\"><p>Welcome to hiccup!</p></div>"
```

```
user=> (html [:p "<oops"])
"<p><oops</p>"
```

```
user=> (require '[hiccup.core :refer [html h]])
nil
user=> (html [:p (h "<oops")])
"<p>&lt;oops</p>"
```

```
user=> (html [:li (map (partial vector :li) ["Cow" "Pig" "Sheep"])])
"<li><li>Cow</li><li>Pig</li><li>Sheep</li></li>"
user=> (html [:li (map (fn [s] [:ul (h s)]) ["Cow" "Pig" "Sheep"])])
"<li><ul>Cow</ul><ul>Pig</ul><ul>Sheep</ul></li>"
```

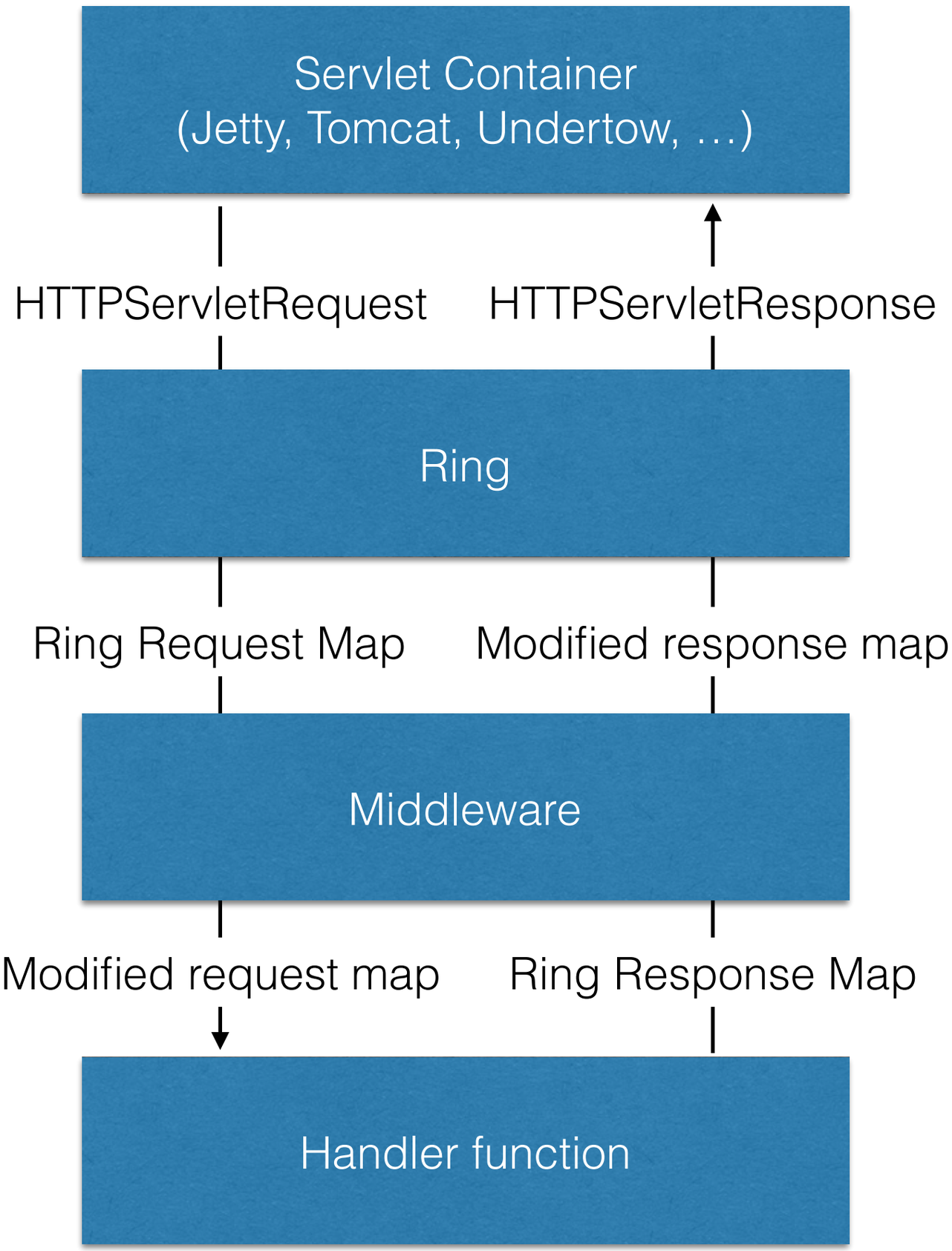
Exercise

- Update your **`handle-hello`** function to return an HTML response

- Hiccup API documentation:

  - http://weavejester.github.io/hiccup/

Middleware


Servlet Container
(Jetty, Tomcat, Undertow, …)
HTTPServletRequest
HTTPServletResponse
Ring
Ring Request Map
Modified response map
Middleware
Modified request map
Ring Response Map
Handler function

Middleware

```
(defn my-middleware [handler]
  (fn [request]
    (handler request)]))
```

```
(defn wrap-params
  [handler & [options]]
  (fn [request]
    (handler (params-request request options))))
```

Ring Defaults

- https://github.com/ring-clojure/ring-defaults/blob/master/src/ring/middleware/defaults.clj

Ad Server

The Demo App

- Create ads and upload images
- Randomly select ads
- Output JavaScript to embed your ads in a website
- Track click-throughs

Reference

- Compojure
    - https://github.com/weavejester/compojure/wiki
    - http://weavejester.github.com/compojure
- Ring
    - http://ring-clojure.github.io/ring/
    - https://github.com/ring-clojure/ring-defaults
- Hiccup
    - https://github.com/weavejester/hiccup/wiki
    - http://weavejester.github.com/hiccup
- Clojure JDBC
    - http://clojure-doc.org/articles/ecosystem/java_jdbc/home.html
    - http://clojure.github.com/java.jdbc/
- Buddy
    - https://funcool.github.io/buddy-auth/latest/