Advanced Clojure Microservices
Tobias Bayer
Senior Developer / Software Architect
inovex GmbH

Clojure, Java, Cloud

tobias.bayer@inovex.de

https://github.com/tobiasbayer
(defroutes app
  (GET "/messages/:name" [name] {:body {:message (str "Hello World" " " name)}})
  (route/resources "/")
  (route/not-found "Not Found")
)

(defn -main [& args]
  (run-jetty app {:port 3000 :join? false })))
O RLY?
Project

~ 20 Microservices
High Availability
Polyglot
About 30% of services in Clojure
Running on AWS
Real World Problems

Resource Modeling
Fault Tolerance
Monitoring and Metrics
Cloudability

...
Resource Modeling with Liberator
Resource Modeling with Liberator
# Resource Modeling with Liberator

## Decisions

<table>
<thead>
<tr>
<th>Decision</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>known-method?</td>
<td>Is the request method known?</td>
<td>:known-methods</td>
</tr>
<tr>
<td>malformed?</td>
<td>Is the request malformed?</td>
<td>false</td>
</tr>
<tr>
<td>media-type-available?</td>
<td>Is the requested media-type available? Should set :media-type in representation</td>
<td>:available-media-types</td>
</tr>
<tr>
<td>method-allowed?</td>
<td>Is the request method allowed for this resource?</td>
<td>:allowed-methods</td>
</tr>
<tr>
<td>modified-since?</td>
<td>Was the resource modified since the date given in the headers “If-Modified-Since”?</td>
<td>:last-modified</td>
</tr>
<tr>
<td>moved-permanently?</td>
<td>Was the resource moved permanently?</td>
<td>false</td>
</tr>
<tr>
<td>moved-temporarily?</td>
<td>Was the resource moved temporarily?</td>
<td>false</td>
</tr>
</tbody>
</table>
## Resource Modeling with Liberator

### Actions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:initialize-context</td>
<td>first function to be called. Can be used to initialize the context since 0.14.0</td>
</tr>
<tr>
<td>:post!</td>
<td>called for POST requests</td>
</tr>
<tr>
<td>:put!</td>
<td>called for PUT requests</td>
</tr>
<tr>
<td>:delete!</td>
<td>called for DELETE requests</td>
</tr>
<tr>
<td>:patch!</td>
<td>called for PATCH requests</td>
</tr>
</tbody>
</table>
## Resource Modeling with Liberator

### Handlers

<table>
<thead>
<tr>
<th>Handler</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle-ok</td>
<td>200</td>
</tr>
<tr>
<td>handle-created</td>
<td>201</td>
</tr>
<tr>
<td>handle-options</td>
<td>201</td>
</tr>
<tr>
<td>handle-accepted</td>
<td>202</td>
</tr>
<tr>
<td>handle-no-content</td>
<td>204</td>
</tr>
<tr>
<td>handle-moved-permanently</td>
<td>301</td>
</tr>
<tr>
<td>handle-see-other</td>
<td>303</td>
</tr>
<tr>
<td>handle-not-modified</td>
<td>304</td>
</tr>
<tr>
<td>handle-moved-temporarily</td>
<td>307</td>
</tr>
<tr>
<td>handle-multiple-representations</td>
<td>310</td>
</tr>
<tr>
<td>handle-malformed</td>
<td>400</td>
</tr>
<tr>
<td>handle- unauthorized</td>
<td>401</td>
</tr>
<tr>
<td>handle-forbidden</td>
<td>403</td>
</tr>
<tr>
<td>handle-not-found</td>
<td>404</td>
</tr>
</tbody>
</table>
(defroutes app
  (ANY "/api/users" [] r/users)
  (ANY "/api/users/:userhandle" [userhandle] (r/user userhandle))
  (ANY "/api/authentication" [] r/authentication))

Resource Modeling with Liberator
Resource Modeling with Liberator

(defresource users
  :available-media-types ["application/user-v1+json"]
  :allowed-methods [:post]
  :malformed? (fn [ctx] (s/validate-user (body ctx)))
  :handle-malformed (fn [ctx] (ring-response {:status 400
                                :body
                                {:message (str (s/validate-user (body ctx))))}))
  :post! (fn [ctx] (if (user-exists? (body ctx))
                       {:conflict true}
                       {:created-entity (create-user (body ctx))}))
  :handle-created (fn [ctx] (if (:conflict ctx)
                               (ring-response {:status 409})
                               (r/user-output (:created-entity ctx))))
Fault Tolerance with Hystrix

Circuit Breaker
Netflix
hystrix-clj
clj-http-hystrix
Fault Tolerance with Hystrix

```clojure
(h/add-hook {:hystrix/timeout-ms (:HYSTRIX_TIMEOUT config/settings)
             :hystrix/threads (:HYSTRIX_THREADS config/settings))

(defn search-customer [param]
  (client/get
   (str api-endpoint "'/customers?search=" param)
   {:content-type :json
    :as :json
    :hystrix/group-key "customers"
    :hystrix/fallback-fn (constantly {:body {:name "Not" :firstname "Found"}}}))
)```
Metrics with Prometheus

Currently Registered Customers: 32164

Registrations within last 24 hours: + 210

Disabled Users: 0

Cache Hits: 3294

Cache Misses: 644

Cache Hit Rate in %: Avg: 83.66% Current: 83.03%
Metrics with Prometheus
Metrics with Prometheus
Metrics with Prometheus

prometheus-clj

(defroutes app
  ;; ...
  (ANY "/internal/metrics" []
    (prometheus/dump-metrics (:registry @metrics/metrics-store)))))

(def handler
  (-> app
    ;; ...
    (prometheus/instrument-handler "" (:registry @metrics/metrics-store))))
Metrics with Prometheus

(defonce metrics-store (atom (prometheus/init-defaults)))

(defn- register-metrics []
  (-> @metrics-store
      (prometheus/register-gauge "users" "users_count" "Number of users in DB" [:"users_count"])
      (prometheus/register-gauge "users" "users_disabled" "Number of disabled users in DB" [:"users_disabled"])
      (prometheus/register-counter "users" "cache_hits"
        "Number of cache hits when requesting a customer" [:"cache_hits"])
      (prometheus/register-counter "users" "cache_misses"
        "Number of cache misses when requesting a customer" [:"cache_misses"]))))

(defn init []
  (reset! metrics-store (register-metrics))
  (HystrixPrometheusMetricsPublisher/register (:registry @metrics-store)))
(defn update-metrics []
  (prometheus/set-gauge @metrics-store "users" "users_count" (p/count-users) ["users_count"])
  (prometheus/set-gauge @metrics-store "users" "users_disabled"
    (p/count-users {:enabled false}) ["users_disabled"]))

(defn inc-cache-hits []
  (prometheus/increase-counter @metrics-store "users" "cache_hits" ["cache_hits"] 1))

(defn inc-cache-misses []
  (prometheus/increase-counter @metrics-store "users" "cache_misses" ["cache_misses"] 1))
(defn process-event [event]
  (when (= (:eventType event) "SubscriptionChanged")
    (log/info "Received event" event)
    (handle-subscription-changed-event event)))

(defn start-worker! [fn-processor]
  (kinesis/worker! :app (:APP_NAME config/settings)
                  :stream (:EVENT_STREAM config/settings)
                  :endpoint (:KINESIS_ENDPOINT config/settings)
                  :region-name (:AWS_REGION config/settings)
                  :processor (fn [records]
                   (doseq [record records]
                    (process-record record fn-processor))))))
USED CLOJURE FOR MICROSERVICES

WORKED LIKE A CHARM
Wir suchen Verstärkung in den Bereichen
• Digital Consulting
• Web Frontend & Backend
• Replatforming
• Apps & Smart Devices
• Big Data
• Data Science
• Search
• Business Intelligence
• DevOps
• Cloud
• Operations

Hamburg, Köln, München, Karlsruhe, Pforzheim

jobs@inovex.de
www.inovex.de/jobs
+49 721 619 021 50