(**defn** weapons-query-handler [{{*:strs* [name]} *:params :keys* [conn] *:as* request}]

(**defn** file—handler [{:keys [queue conn] :as ctx} {:keys [^File file kind] :as event}]

(f/insert-file conn {:name (.getName file) :processed (Date.)})))

{:connection-uri "jdbc:h2:mem:mem_only"} {::jdbc/connection

(timbre/debug (str "Detected change to file: " (.getName file)))

(**defn** add-weapons-handler [{:keys [params conn] :as request}]

(let [[name & weapons] (map cs/trim (cs/split line #","))]

{:conn (ig/ref ::jdbc/connection)} ::jdbc/init

(**let** [data (**map** (**fn** [[k v]] {*:name* k *:weapon* v}) params)]

(defmethod ig/init-key ::jdbc/init [_ {:keys [conn]}]

{:groups [{:paths ["example"] ::hawk/watch

;(timbre/debug "Checking for new items in queue...")

(when-some [task (dq/take! queue queue-name 10 nil)]

{:job ::scheduling/job #'queue->dsdb (defn queue->dsdb [{:keys [queue-name queue dsdb]}]

{:data {:coercion reitit.coercion.spec/coercion

{:delete-on-halt? true ::durable/queues

{:in [5 :seconds] :every :second} :schedule

(timbre/debug "Putting data into datascript")

(ig/ref ::datascript/connection)} :dsdb

{:host "0.0.0.0" ::web/server

(ig/ref ::datascript/connection) :conn

```
(d/transact! dsdb [(line->record @task)])
```

(ok (with-out-str (pp/pprint request))))

(defn files-handler [{:keys [sql-conn]}]

;We were able to compose the routes here

```
(ok (w/weapons @conn (keys params))))))
```

```
(cs/ends-with? (.getName file) ".csv"))
```

(constantly (not-found "Not found"))))

(ok (f/all-processed-files sql-conn)))

(timbre/debug "Adding data to queue.")

["/add_weapon" add-weapons-handler]])

(when (and (#{:modify :create} kind)

[["/weapons" weapons-query-handler]

:sql-conn (ig/ref ::jdbc/connection)

(ig/ref ::jdbc/connection)} :conn

["/files" {:get files-handler}]])

:queue (ig/ref ::durable/queues) (dq/put! queue :my-queue line)))

::datascript/connection w/schema

(w/everybodys-weapons @conn))))

:middleware [params/wrap-params

[["/echo" {:get echo-handler}]

[basic-routes weapons-routes]

(defonce sys (create config))

(defn echo-handler [request]

middleware/wrap-format]}}))

(assoc :weapon weapons))))

(doseq [line (line-seq r)]

(defn line->record [line]

:handler #'file-handler}]

"/tmp"} :directory

(w/weapons @conn [name])

(d/transact! conn data)

(dq/complete! task))))

:queue-name :my-queue

(cond-> {:name name}

(def weapons-routes

;Routes - All data

(def basic-routes

;Global handler

(f/setup conn))

(.exists file)

(.isFile file)

3000 :port

(**seq** weapons)

(ring/router

(def handler

(def router

(def config

;Handlers

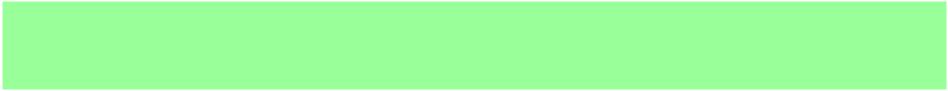
name

;Router

router







{:conn (ig/ref ::jdbc/connection)} ::jdbc/init

(**let** [data (**map** (**fn** [[k v]] {*:name* k *:weapon* v}) params)]

(defmethod ig/init-key ::jdbc/init [_ {:keys [conn]}]

{:groups [{:paths ["example"] ::hawk/watch