


```
(defn weapon-query-handler [{:status [name]} params :keys [conn] :as request}]
```

```
(defn file-handler [{:keys [query]} {:keys [^File kind] :as event}]
```

(f/insert-file on {*name*(getNamel file): *prosed*(Date.)})

{:jdbc/connnection{:connnection-urn'jdbc:h2:mem:mem_only'}}

(tinbre/debug(str 'Detected hang to file: ' (getNamel file)))

```
(defn add-weapon-handler [{:keys [params] :as request}]
```

```
(let [name & weapons] (map cs/trim (cs/spplit line #', ''))])
```


jdgc/init { :cnn (ig/ref :: jdgc/cnn) }

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

(defmethod dig/init-key *:jdbc/init* [_ {*:keys* [conn]}])

`awk/watch { :groups { :paths ['example']`

;(tibre/debug'checking for new items in queue.')

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

is cheduling/job {*job* #'queue->dsdb

(defn queue->dsdb [{:keys [queue-name queue-dsdb]}])

`{:data{:precisionreitit.cerion.spec/cerion`

if durnable/queues { if delete-on-halt? true

is chdule { : in [5 : seconds : every : second }

(tinbre/debug 'Putting data into data script')

:dsdb(ig/ref::data\$ip\$connection)

webserver { host "0.0.0.0"

:con (ig/ref::data input/connetion)

(d//t randt!dsdb[(line->read@task)])

`(ok(without-str(pp/print request)))`

```
(defn filter-handler [{:keys [sql-con]}])
```

We were able to connect the routes here

(ok(w/weapons@acornn(keyparams)))

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

(constantly(not-found'Not found'))

(ok(f/all-proposed-files ql-con))

(timbre/debug 'Adding data queue.')

['/add_webon' add-weapons-handler'])

(when (and (#*modify* *read*) kind)

['/weapons'-query-handler]

sqz-conn(ig/ref:jdb/cnnec tion)

{conn(i:ig/ref::j:dbc/connec:tion)}

```
["/files"{:get_files-handler}] ])
```

:queue (ig/ref::durablere/queue)

(dq/put! queue:my-queue line))

data/cnript/cnnnet_idn w/schema

(w/everybody's-weapons@com))

mind/leware [params/wrap-params]

```
[[['/echo'{:aget_echo_handler}]]
```

[baisic-routes-weapons-routes]

(define sys(create conf))

(defn echo-handler [request])

mind[eware//wrap-format]})

(assoc: *Weapon* weapons))

(doseq[line](line-seq r))]

(define-line->record [line])

hand/zer # *final-e-hand/er* }

`:'directory'"/tmp'}`

(w/weapons@conn[name])

(d/transact!commanddata)

$(dq/d\text{complete!task})'$

queque-nammy-queque

(cond=>{:name name})

(def weapons-routes

Routes - All data

(defbasic-inputs

Agglutinated

(f/setupcon)

(.exits file)

(.isFile file file)

print 30000

(seq weapons)

(ring/router

def handler

defrouter

(defconfid)

Handzlers

)

i

f

n

a

m

e

Router

route

ctx)

(

o

k



do

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


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

(defmethod dig/init-key *jd/bc/init* [_ {*keys* [conn]}])

`awk/watch { :groups { :paths ['example']`

