

`{:jdbc/cnnec tion{:cnnec tion-un i'jdbc:h2:mem:mem_only'}}`

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

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

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

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

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

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


::jdbc/init::jdbc/condition}

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

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

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

(defnet hodb :*ig* / *init* - key :*jdbc* / *init* [_ { :*keys* [conn] }])

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

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

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

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

if durable/queue { if delete-on-halt? true

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

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

isdule { : in [5 : seconds : every : second }

webserver { host "0.0.0.0"

:con (ig/ref::datainputconnection)

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

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

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

;We were able to connect here

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

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

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

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

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

['/add_weapon' add_weapon-handle'])

:queue (ig/ref::durableref/queue)

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

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

[[['/weapons'weaponry-handler]]

$\{conn(i_g/ref:j_dbc/connec tion)\}$

:queue (ig/ref:duration/queue)

```
["/files":get_files_handler])
```


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

data/crypt/conn *idn* **wschema**

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

minddleware [params/wrap-params]

(with-open[r(io//reader file)])

```
[[['/echo'},{:get_echo_handler}]]
```

[basic-routes-weapons-routes]

(define sys(create conf))

(defn echo-handler [request])

middeeware/wrap-formats})

(as doc: *weapon* weapons)

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

(define-line->record [line])

*:hand/zer#'*final-e-handler}]

(w/weapons@conn[name])

`:"directory"/tmp"}`

(d/transact!command)

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

#hand/ler #'handler'})

queque-nammy-queque

(cond=>{:name name})

Routes - All data

(def weapons-routes

(ring/ring-handler

(defbasic-inputs

;Gzobazhander

(f/setup con)

(.exits file)

(.isFile file file)

print 30000

(seq weapons)

(ring/router

Handletter

defrouter

(defconfi

Handzlers

)

i

f

n

a

m

e

Router

ctx)

route er

(

o

k









