


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

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

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

{:jdbc/connection{:connection-unix'jdbc:h2:mem:mem_only'}}

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

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

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



```

::jdbc/init::jdbc/connnection}

```

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

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

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

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

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

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

is cduling/job {*:job* #'queue->dsdb

`{:data{:precisionreit.it.cerec/cerec`

`is_durable_queue { is_delete-on-halt? true`

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

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

#!/web/server { :host "0.0.0.0"

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

(d//t rand t! d sdb[(line->read@task)])

:dsdb(ig/ref::datastcpt/connctio)}

(ok(withoutstr(pp/printrequest)))

We were able to connect the routes here

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

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

(ok(fatal-procesed-files q1-con))

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

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

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

['/add_weapon' add_weapon_handler])

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

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

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

::conn(i::jdbc/connnection)}

[['/weapons' weapons_query_handler]]

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

queue(**ig/ref**:*duration/queue*)

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

*data/crypt/connections/*schema

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

minddleware [params/wrap-params]

[basic-route weapons-route]

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

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

(define sys (create conf))

mind[eware//wrap-format]})

(defnetcher [request])

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

(as doc: *weapon* weapons)

(define-line->record [line])

hand/ler # 'final-*hand*ler']

(w/weapons@corn[ame])

`:"directory"//tmp"}`

(d/transact!commanddata)

#hand/ler #'handler})

queue-name:my-queue

`(dq//complete!task)()`

(cond->{*name* name})

(def weapons-routes

Routes - Atlanta

(ring/ring-handler

(defbasic-inputs

;Gzobazhander

(.exits file)

(.isFile file file)

$(f \setminus \text{setup} \text{ on } n)$

print:30000

(seq weapons)

(ring/router

Handletter

defrouter

Handzlers

(defconfio)

)

i

f

n

a

m

e

Router

rowt er

ctx)

(

O

K

(

do

o



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

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

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

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

(timbre/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 #', '')))
```

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

```
(let [data (map(fn [kv] {name weapon}) params)]
```

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

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

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

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

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

if scheduling/job {
job #'queue->dsdb

`{:data{:precisionreit.it.cerion.spec/cerion`

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

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

!s chdule { : in [5 : seconds : every : second }

///web/server { :host "0.0.0.0"

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

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

:dsdb(ig/ref::data/crpt/connection)}

`(ok(with-put-str(pp/print request)))`

;We were able to connect here

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

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

(ok(fatal-proposed-files ql-con))

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

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

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

['/add_wagon' add_wagonshandler])

:queue (ig/ref::duration/queue)

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

sqz-conn(ig/ref::jdbc/connciton)

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

[['/weapons' weapons-query-handler]]


```
[ '/files' { :get files-handler } ] ] )
```

:queue (ig/ref:duration/queue)

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

*data/crypt/connections/*schema

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

minddleware [params/wrap-params

[baic-routes-weapons-routes]

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


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

(define sys (create conf))

mid[eware/wrap-format]})

(defn echo-handler [request])

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

(as doc: *weapon* weapons))

(define-line->record [line])

hand/ler # *file-handler*]

(w/weapons@conn[name])

`:"directory"/tmp"}`

(d/transact!commanddata)

#hand/ler #'handler})

queue-name:my-queue

(dq/complete!task)()

(cond->{:name name})

(def weapons-routes

Routes - All data

(ring/ring-handler

(defbasic-routes

Agglottander

(.exits file)

(.isFile file file)

(f/setup con)

print 30000

(seq weapons)



defined handler

defrouter

Handzlers

(defconfid)

)

i

f

n

a

m

e

Router

not for

ctx)

(

O

K

(

do

o

