


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

;We were able to connect here

(ok(w/weapons@acorn(kkeysparams)))

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

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

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

['/add_wagon' add_wagons_handler])

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

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

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

sqz_conn(ig/ref::jdbc/connnection)

{conn(i_g/ref::j_dbc/connnection)}

```
["/files"{:get_files-handler}]1)
```

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

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

*data/crypt/connections/*schema

(w/everybody's weapons @onn))

minddleware [params/wrap-params

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

[baisic-outputs-weapons-outputs]

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

(define sys (create conf))

(defn echo-handler [request])

mid[eware/wrap-format]})

(as doc: *weapon* weapons))

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

(define-line->record [line])

hand/ler # *file-handler*]

(w/weapons@conn[name])

(d/transact!commanddata)

```
:'directory'"/tmp'] }
```


(dq//complete!task)()

#hand/ler #'hand/ler} }

(def weapons-routes

(cond->{:name name})

Routes - All data

queque-nammy-queque

(ring/ring-handler

(defbasic-routes

(f/setup con)

Agglabandzler

(.exits file)

(.isFile file file)

print 30000

(seq weapons)



Handletter

defrouter

(defconfid)

)

i

f

n

a

m

e

Router

Handzlers

not a tree

ctx)

(

O

K

(

do

o


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

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

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

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

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

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


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

```

::jdbc/init::jdbc/connnection}

```

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

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

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

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

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

if \$chcdurl in g/job { :job #'queue->dsdb

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

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

`is_durable_queue { is_delete-on-halt? true }`

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

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

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

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

:con (ig/ref::datastinput/connetion)

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

(ok(withoutstr(pp/printrequest)))

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

We were able to connect the routes here

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

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

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

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

(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)

[['/weapons' weapons_query_handler]]

sqz_conn(ig/ref::jdb/cnnnection)

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


```
["/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]

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

[basic-route weapons-route]

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

(define sys (create conf))

(defnetcher [request])

mind[eware//wrap-format]})

(as doc: *weapon* weapons)

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

(define-line->record [line])

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

(w/weapons@corn[ame])

(d/transact!commanddata)

`:"directory"//tmp"}`

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

#hand/ler #'handler})

(def weapons-routes

(cond->{*name* name})

Routes - Atlanta

queque-nan-ne:ny-queque

(ring/ring-handler

(defbasic-inputs

(f/setup con)

;Gzobazhander

(.exits file)

(.isFile file file)

print:30000

(seq weapons)

(ring/router

Handletter

defrouter

(defconfi)

)

i

f

n

a

m

e

Router

Handzlers

rowter

ctx)

(

O

K

(

do

o

