





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

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

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

{:jdbc/connection{:connection-unix'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/conn)}*

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

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

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

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

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

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



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

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

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

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

```

:sc:headul{ :in[5:seconds]:every:second}

```

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

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

*:con (ig/ref::datast/nt/connetion)*



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

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

*We were able to connect the routes here*

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

**(ok(w/weapons@acnn(keysparams)))**

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

**(ok(f/all-procesed-files ql-conn))**

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



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

['/add\_webon' add-weapons-handler])

*:queue (ig/ref::durab le/queue)*

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

*sqz-conn(ig/ref::jdb/c/connec<sup>ti</sup>on)*

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

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

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



```
[["files">{:agetfiles-handler}]])
```

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

*data/crypt/connections/schema*

(w/everybody's weapons @onn))

*mind/leware* [params/wrap-params]

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

[baisic-routes-weapons-routes]

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



(define sys(create conf))

(defn echo-handler [request])

mindd[eware//wrap-format] } } ) )

(also: *weapon* weapons)

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

(define-line->record [line])

*hand/len* # *final-len-handlen* }

(w/weapons@corn[ame])



`:'directory'"/tmp'}`

(d/transact!commanddata)

**(dq/complete!task)()**

*#hand/ler* #'handler'})

*queue-name:my-queue*

(cond  $\rightarrow$  { :name name })

(def weapons-routes

*;Routes--Alldata*



**(ring/ring-handler**

(defbasic-routes

*Agglottation*

(f/setupcon)

**(.exits file)**

**(.isFile file file)**

(seq weapons)

**(ring/router**



*print:30000*

def handler

defrouter

(defconfid

*Handzlers*

)

i

f

n

a

m

e

*Router*

rover



ctx)

(

o

k



do

o





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

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

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



{:jdbc/connnection{:connnection-urn'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 k; :weapon v}) params)]
```

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

(defmethod dig/init-key *jd/bc/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)])

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

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

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

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

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



*: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)))`

*We were able to connect the routes here*

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

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



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

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

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

(timbre/debug 'Adding data queue.')

['/add\_webon' add\_webonshandler])

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

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

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



['/weapons'-query-handler]

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

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

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

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

*data/cnript/cnnect\_idn* w/schema

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

*mind/leware* [params/wrap-params]



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

[baisic-routes-weapons-routes]

```
[[['/echo'][:get_echo_handler]]]
```

(define sys(create conf))

(defn echo-handler [request])

mind[eware//wrap-format]})

(assoc: *Weapon* *Weapon*)

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



(define-line->record [line])

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

(w/weapons@corn[ame])

`:"directory"//tmp"}`

(d/transact!commanddata)

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

*#hand/ler* **#'hand/ler}}**)

queque-nammy-queque



(cond => { :name name })

(def weapons-routes

*;Routes--Alldata*

**(ring/ring-handler**

(defbasic-inputs

*Agglottated*

**(f/setup con)**

**(.exits file)**



**(.isFile file file)**

(seq weapons)

**(ring/router**

*print* 30000

def handler

defined router

(defconfid)

*Handzlers*



)

i

f

n

a

m

e

*Router*

route

ctx)

(

o

k



do