



{:jdbc/condition{:condition-unix'jdbc:h2:men:men\_only'}}

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

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

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

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

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

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



*:= 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 *jdbc/init* [{:keys [conn]}])

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

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

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

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



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

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

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

`dsdb(ig/ref::data/cnn/connection)`

webserver { host "0.0.0.0"

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

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

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



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

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

*We were able to connect the route here*

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

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

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

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

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



['/add\_wapon' add-wapons-handle'])

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

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

*sqz-conn(ig/ref:jdb/c/connnection)*

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

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

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

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



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

*data/cnript/cnncnwn/schema*

(w/everybody's weapons @onn))

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

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

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

[baisic-routes-weapons-routes]

(define sys(create conf))



(defn echo-handler [request])

mind[eware/wrap-format]})

(asoc: *weapon* weapons)

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

(define-line->record [line])

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

(w/weapons@conn[name])

`:"directory"//tmp"}`



(d/transact! command)

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

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

queue-name:my-queue

(def weapons-routes

*Routes - All data*

(cond=>{:name name})

**(ring/ring-handler**



(defbasic-inputs

Agglutiner

**(f/setup con)**

**(.exits file)**

**(.isFile file file)**

*print* 30000

(seq weapons)

**(ring/router**



Handletter

defrouter

(defconfid)

*Handzlers*

if name

*Router*

route er

ctx)



(

o

k

(

d

o



