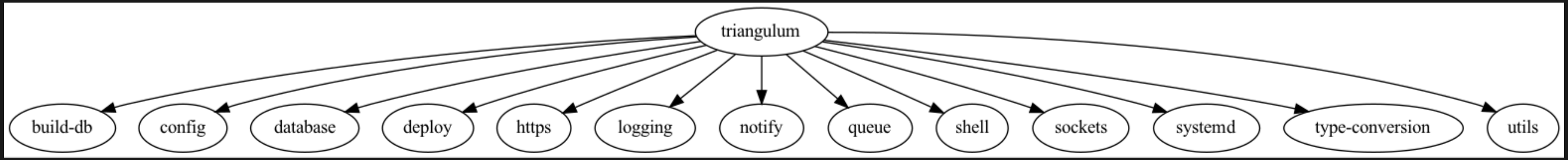
REDUCING ENTROPY WITH THE TRIANGULUM

Created: 2022-10-25 Tue 09:24

OVERVIEW


triangulum
build-db
config
database
deploy
https
logging
notify
queue
shell
sockets
systemd
type-conversion
utils

TRIANGULUM.CONFIG

- Organizing an application's configurations via `config.edn` file in the project's root directory.

CONFIG.EDN

```clojure
;; config.edn
{:database {:host     "localhost"
            :port     5432
            :dbname   "dbname"
            :user     "user"
            :password "super-secret-password"}
 :mode     "dev"
 :mail     {:host "smtp.gmail.com"
            :user "test@example.com"
            :pass "3492734923742"
            :port 587}}
```

USING TRIANGULUM.CONFIG

```
(triangulum.config/get-config :database) ;; -> {:user "triangu
(triangulum.config/get-config :database :user) ;; -> "triangul

(triangulum.config/get-config :server) ;; -> {:http-port 8080
(triangulum.config/get-config :server :http-port) ;; -> 8080
```

TRIANGULUM.LOGGING

```
(require '[trianglum.logging :refer [log log-str]])

(log "Hello world" {:newline? true :pprint? false :force-stdou
(log-str "Hello" "world")
```

SETTING LOG PATH

- By default the above will log to standard out. If you would like to have the system log to YYYY-DD-MM.log, set a log path.

-You can either specify a path relative to the toplevel directory of the main project repository or an absolute path on your filesystem. The logger will keep the 10 most recent logs (where a new log is created every day at midnight). To stop the logging server set path to "".

```
(set-log-path "logs")
(set-log-path "")
```

TRIANGULUM.DATABASE

```
(call-sql "function" {:log? true :use-vec? false} "param1" "pa

(insert-rows! table-name rows-vector fields-map)
(update-rows! table-name rows-vector column-to-update fields-m
```

SQLITE

```
(call-sqlite "select * from table" "path/db-file")
```

SETTING UP DB CONNECTION VIA CONFIG.EDN

```clojure
;; config.edn
{:database {:host     "localhost"
            :port     5432
            :dbname   "pyregence"
            :user     "pyregence"
            :password "pyregence"}}
```

TRIANGULUM.BUILD-DB

```
{:aliases {:build-db {:main-opts ["-m" "triangulum.build-db"]}
```

DATABASE SETUP

```
clojure -M:build-db build-all -d database [-u user] [-p admin
```

DATABASE BACKUPS/RESTORE

```
clojure -M:build-db backup -f somefile.dump
clojure -M:build-db restore -f somefile.dump
```

DATABASE MIGRATIONS

INCOMING

```
clojure -M:build-db migrate
```

TRIANGULUM.SYSTEMD

- Creates a systemd user `.service` file.

```
{:aliases {:systemd {:main-opts ["-m" "triangulum.systemd"]}}}
```

MODIFYING YOUR SERVER.CLJ

Modify your app code to call `(triangulum.notify/ready!)` after all of your application's services are started:

```clojure
(ns <app>.server
  (:require [triangulum.notify :as notify]))
...

(defn app-start []
  (reset! db (jdbc/connect!))
  (reset! queues (q/start!))
  (reset! server (ring/start-server!)
  (when (notify/available?) (notify/ready!))))
```

ENABLING SYSTEMD

And then run:

```
clojure -M:systemd enable -r <REPO> -u <USER> [-p HTTP PORT] [
```

START/STOP/RESTART SYSTEMD

Now your service will be enabled at startup. You can also start, stop, and restart your service with the following commands:

```
clojure -M:systemd start -r <REPO>
clojure -M:systemd stop -r <REPO>
clojure -M:systemd restart -r <REPO>
```