DUCT

Описываем зависимости данными

О СЕБЕ

Меня зовут Миша
Software engineer в Takeoff
Пишу на Clojure в прод с 2018

ЧТО ТАКОЕ DUCT?

- модульный data-driven фреймворк для серверных приложений
- основанный на integrant
- следует методологии Twelve-Factor App
- из коробки умеет в Reloaded workflow
- и тд.

INTEGRANT

README.md

Integrant

build passing

> integrant /ˈɪntɪgr(ə)nt/
>
> (of parts) making up or contributing to a whole; constituent.

Integrant is a Clojure (and ClojureScript) micro-framework for building applications with data-driven architecture. It can be thought of as an alternative to Component or Mount, and was inspired by Arachne and through work on Duct.

А ЕСЛИ ПРОЩЕ?

- Позволяет описывать компоненты системы в edn файле в виде map-ок
- Управляет жизненным циклом компонент (prep-key, init-key, halt-key, resume-key, suspend-key)
- Управление зависимостями (#ig/ref, #ig/refset)

```
{:app/jetty                          {:port    8080
                                      :handler #ig/ref :app.rest/router}
 :app.sql/hikaricp                   {}
 :app.rest/router                    {:routes [["/api"
                                                ["/v1"
                                                 {:swagger {:tags ["V1"]}}
                                                 ["/hello"
                                                  {:post {:summary "Saying hello to users."
                                                          :parameters {:body :app.spec.handler.hello/body}
                                                          :responses  {200 {:body :app.spec.handler.hello/success}
                                                                       400 {:body :app.spec.common/bad-request}}
                                                          :handler    #ig/ref :app.handler/hello}}]]]]}
 :app.handler/hello                  {:users-boundary #ig/ref :boundary/users}
 [:app.sql/query :query/users]       {:sql-file "users"}
 [:app.sql/boundary :boundary/users] {:queries    #ig/ref :query/users
                                      :datasource #ig/ref :app.sql/hikaricp}}
```

НО MOUNT УМЕЕТ ТОЖЕ САМОЕ

- Автоматическая валидация spec (pre-init-spec)
- Вывод определения (derive)
- Автоматический импорт ns

```clojure
(ns app.core
  (:gen-class)
  (:require app.jetty
            [mount.core :as mount]))

(defn main [& args]
  (mount/start))
```

```clojure
(ns app.core
  (:gen-class)
  (:require [clojure.java.io :as io]
            [integrant.core :as ig]))

(defn main [& args]
  (let [config (->> (io/resource "config.edn")
                    (slurp)
                    (ig/read-string))]
    (if (seq args)
      (ig/init config (map keyword args))
      (ig/init config))))
```

ОТДЕЛЕНИЕ ЧИСТОГО КОДА

```clojure
(ns app.utils
  (:require [taoensoo.timbre :as log]
            [app.boundary.users :as users]
            [app.boundary.impl.users :refer [users-boundary]]))

(defn say-hello! [name]
  (try
    (let [greetings (users/say-hey users-boundary name)]
      (log/infof "Sent greetings: %s" greetings)
      greetings
      (catch Exception e
        (log/errorf "Cannot say hello to %s" name)
        (throw e)))))
```

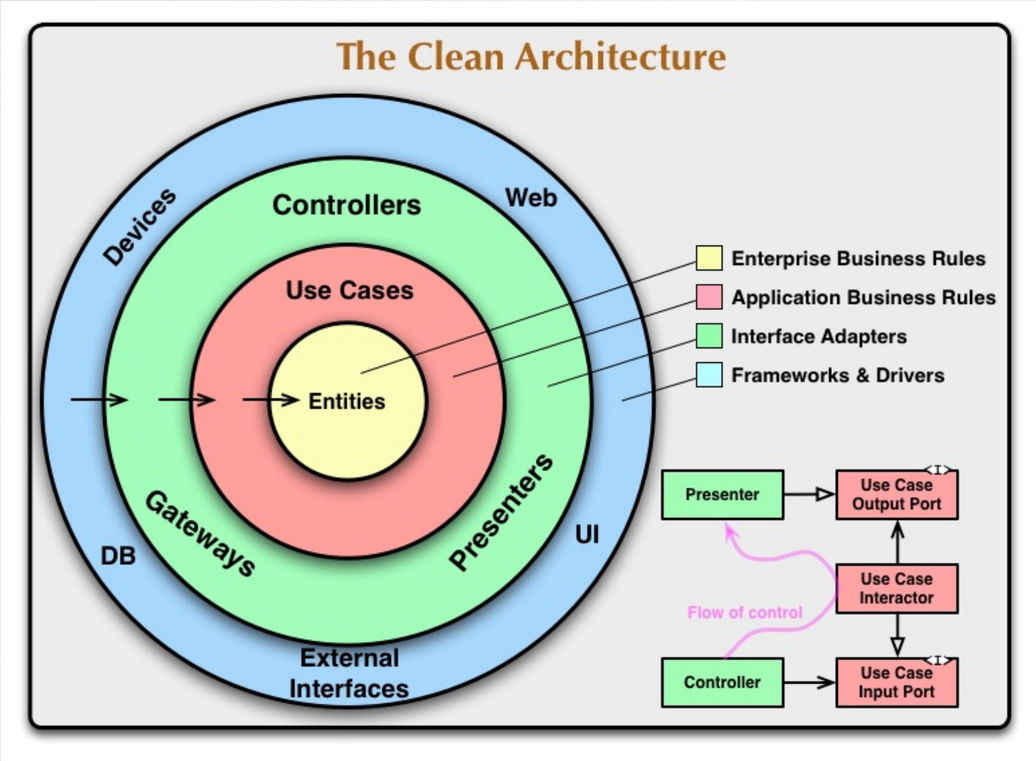
```clojure
(ns app.utils
  (:require [duct.logger :refer [log]]
            [app.boundary.users :as users]))

(defn say-hello!
  [{:keys [users-boundary logger]}
   name]
  (try
    (let [greetings (users/say-hey users-boundary name)]
      (log logger :info (format "Sent greetings: %s" greetings))
      greetings
      (catch Exception e
        (log logger :error (format "Cannot say hello to %s" name))
        (throw e)))))
```

А ЗАЧЕМ DUCT?

- Модули
- Профили
- Наследование
- Reloaded workflow
- Концепт с баундари

НО ДЯДЮ БОБА ВСЕ 100 ЛЕТ НАЗАД ПРОЧИТАЛИ




```
{[:app.boundary/http :boundary/auth]  {:url "https://example.com"}
 [:app.boundary/http :boundary/stock] {:utl "https://other-example.com"}}
```

```
(defmethod ig/init-key :app.boundary/http
  [_ props]
  ... )

(defrecord ExternalService [])
```

```
(defprotocol Auth
  (check-user [this user pass])
  (check-token [this token]))


(extend-protocol Auth
  ExternalService
  (check-user [{:keys [query-fn]} user pass] ... )
  (check-token [{:keys [query-fn]} token] ... ))
```

```
(defprotocol Stock
  (get-stock [this item-id])
  (set-stock [this item-id]))


(extend-protocol Stock
  ExternalService
  (get-stock [{:keys [query-fn]} item-id] ... )
  (get-stock [{:keys [query-fn]} item-id] ... ))
```

ТАМ БЫЛО ЧТО-ТО ПРО НАСЛЕДОВАНИЕ

```
{:duct.logger/timbre [:duct/logger]}
```

```
{:duct.profile/base {:duct.logger/timbre {:set-root-config? true
                                          :level            #duct/env ["LOG_LEVEL" Keyword :or :debug]
                                          :ns-blacklist     []
                                          :ns-whitelist     []
                                          :middleware       []
                                          :timestamp-opts   {:locale   :jvm-default
                                                             :pattern  "yy-MM-dd HH:mm:ss"
                                                             :timezone :utc}}
                     :app.handler/hello  {:logger #ig/ref :duct/logger}}}
```

А ЧТО ЕЩЕ ЕСТЬ?

- Дефолтные определения для констант duct/const
- Решенная проблема чтения переменных из окружения duct/env со встроенным расширяемым коерсером
- Модульные конфиги duct/include
- Чтение ресурсов duct/resource

ПРОФИЛИ

```
{:duct.profile/base {:duct.logger/timbre {:set-root-config? true
                                          :level            #duct/env ["LOG_LEVEL" Keyword]
                                          :ns-blacklist     []
                                          :ns-whitelist     []
                                          :middleware       []
                                          :timestamp-opts   {:locale   :jvm-default
                                                             :pattern  "yy-MM-dd HH:mm:ss"
                                                             :timezone :utc}}
                     :app.handler/hello  {:logger #ig/ref :duct/logger}}
 :duct.profile/dev  {:duct.logger/timbre {:level :debug}}}
```

МОДУЛИ

- sql
- ragtime
- sass
- carmine
- ataraxy
- cljs
- hikarycp
- http-kit

- figwheel
- aleph
- scheduling
- logging
- buddy
- timbre
- jetty

#{tao·enssō/open-source}

СПАСИБО

https://github.com/mikebohdan

https://t.me/mishabohdan