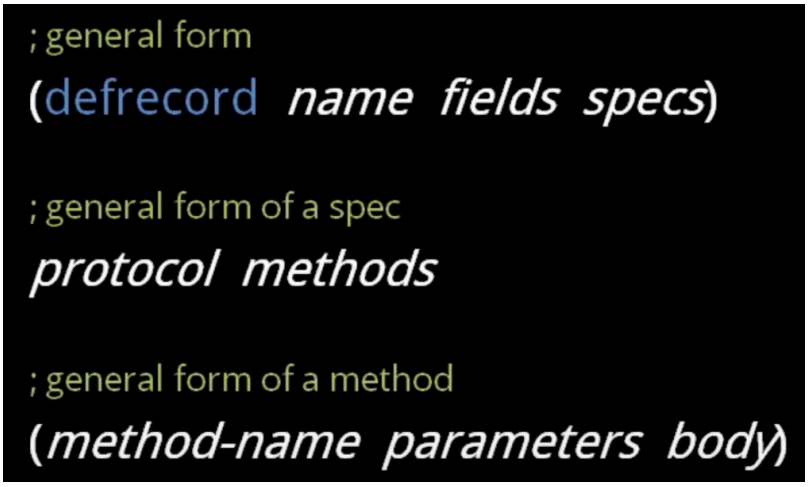
defrecord

```
(defrecord CarModel [make model])

(def fiat-500 (->CarModel "Fiat" "500"))

(def ford-focus (map->CarModel {:make "Ford" :model "Focus"}))

(map->CarModel {}) => {:make nil, :model nil}
(defprotocol Display
  (title [this])
  (desciption [this description]))


(defrecord CarModel [make model]
  Display
  (title [this] (str "This is a " make " " model))
  (desciption [this descr] (str "The " make " " model " is " descr)))

(def fiat-500 (->CarModel "Fiat" "500"))

(title fiat-500) => "This is a Fiat 500"
(def toaster (->Product "Toaster"))

(extend-protocol Display
  Product
  (title [p] (str "This product is a " (:name p)))
  (desciption [p descr] (str "The " (:name p) " is " descr))) => nil

(title toaster)

(desciption toaster " flippin nice") => "The Toaster is flippin nice"
```

```
;; from Stu's examples:

(defrecord Person [fname lname address])
-> user.Person
```

```
(defrecord Address [street city state zip])
-> user.Address

(def stu (Person. "Stu" "Halloway"
           (Address. "200 N Mangum"
                     "Durham"
                     "NC"
                     27701)))
-> #'user/stu

(:lname stu)
-> "Halloway"

(-> stu :address :city)
-> "Durham"

(assoc stu :fname "Stuart")
-> #:user.Person{:fname "Stuart", :lname "Halloway", :address
#:user.Address{:street "200 N Mangum", :city "Durham", :state "NC", :zip
27701}}

(update-in stu [:address :zip] inc)
-> #:user.Person{:fname "Stu", :lname "Halloway", :address
#:user.Address{:street "200 N Mangum", :city "Durham", :state "NC", :zip
27702}}
```



```
(defrecord Nadine [x y z]
    Roger
    (foo [this]
        (do-stuff y))
    (foo [this a b]
        (do-other-stuff b this a)))
    (bar [this]
        (yet-more-stuff x z)))

(def adam (new Nadine 3 “hi” false))
(foo adam 7 2)
(. adam x)          ; returns 3
(get adam :x)       ; returns 3
```

```
(assoc adam :x 5)     ; returns new Nadine record {:x 5 :y “hi” :z false}
(assoc adam :a 5)     ; returns new Nadine record {:x 5 :y “hi” :z false :a 5}
(conj adam [:a 5])    ; returns new Nadine record {:x 5 :y “hi” :z false :a 5}
```

```
(def ted (assoc adam :a 5))  ; returns new Nadine record {:x 5 :y “hi” :z false :a 5}
(dissoc ted :x)              ; returns map {:y “hi” :z false :a 5}
(dissoc ted :a)              ; returns the Nadine record {:x 3 :y “hi” :z false}
```

Links

- https://clojuredocs.org/clojure.core/defrecord
- https://github.com/bbatsov/clojure-style-guide#custom-record-constructors-naming

---

관련 문서

Plugin Backlinks: 아무 것도 없습니다.

From:
https://jace.link/ - **Various Ways**

Permanent link:
https://jace.link/open/defrecord

Last update: **2022/06/19 23:59**