Read POST Request (application/json)

```
type FormData struct {...}
var fd FormData
// Option 1: With json decoder
err := json.N  ew D e co  der  -
(r e q.B  ody  ).D  e co  de(  -
&fd)
// Option 2: With json unmarshal
body, err := io.Rea  dA l l -
(r  eq.B  ody)
if err == nil {
       err = json.U  nm a r -
sh  al(  body, &fd)
}
```

Loop through an struct var

```
_v := reflect.ValueOf(v)
for _i := 0; _i < _v.Num Fie -
ld(); _i++ {
       if _v.Kind() != reflec -
t.S truct {
                log rus.Er ror -
("Dump expects struct instead of
" + _v.Kin d().St ring())
                return
       }
        fmt.Pr int f("%s =
%s\n", _v.Typ e().Fi eld -
(_i ).Name, _v.Fie ld(_i))
}
```

Futures

```
c := make(chan string, 1)
go func() {c <- proces s()}().
// async
v := <- c // await
```

Scatter / Gather

```
// Scatter
c := make(chan result, 10)
for i := 0; i < cap(c); i++ {
       go func() {
                val, err :=
process()
                c <- result {val,
err}
       }()
}
// Gather
collection := make(s tring,
cap(c))
for i := 0; i < cap(c); i++ {
       res := <- c
       if res.err == nil {
                col lec tion[i]
= res.val
       }
}
```

Consume services with http.Client

```
req, err := http.N ewR equ est -
("PO ST", <ur l>, <bo dy>)
req.He  ad e r.A  dd(  " C on  -
ten  t- t y pe  ", "  ap p l -
ic  ati  on / j so  n")
...
// call req
client := &h  tt p.C l i en  -
t{T  im eout: 15 * time.S  -
econd}
res, err := client.Do  (req)
...
// handle response
data, err := ioutil.Re  ad A l -
l(  res.Body)
json.U nma rsh al( data, &v)
```

By **JA Villalba** (JA Villalba)
cheatography.com/ja-villalba/

Not published yet.
Last updated 21st November, 2022.