

Read POST Request (application/json)

```
type FormData struct {...}
var fd FormData
// Option 1: With json decoder
err := json.NewDecoder(r.Body).Decode(&fd)
// Option 2: With json unmarshal
body, err := io.ReadAll(r.Body)
if err == nil {
    err = json.Unmarshal(body, &fd)
}
```

Loop through an struct var

```
_v := reflect.ValueOf(v)
for _i := 0; _i < _v.NumField(); _i++ {
    if _v.Kind() != reflect.Struct {
        logrus.Errorf("Dump expects struct instead of %s", _v.Kind().String())
        return
    }
    fmt.Printf("%s = %s\n", _v.Type().Field(_i).Name, _v.Field(_i))
}
```

Futures

```
c := make(chan string, 1)
go func() {c <- process()}().
// 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(string, cap(c))
for i := 0; i < cap(c); i++ {
    res := <- c
    if res.err == nil {
        collection[i] = res.val
    }
}
```

Consume services with http.Client

```
req, err := http.NewRequest("POST", <url>, <body>)
req.Header.Add("Content-type", "application/json")
...
// call req
client := &http.Client{Timeout: 15 * time.Second}
res, err := client.Do(req)
...
// handle response
data, err := ioutil.ReadAll(res.Body)
json.Unmarshal(data, &v)
```

