Mergo

A helper to merge structs and maps in Golang. Useful for configuration default values, avoiding messy if-statements.

Also a lovely comune (municipality) in the Province of Ancona in the Italian region of Marche.

Status

It is ready for production use. It is used in several projects by Docker, Google, The Linux Foundation, VMWare, Shopify, etc.

Latest release

Release v0.3.4.

Important note

Please keep in mind that in 0.3.2 Mergo changed `Merge()` and `Map()` signatures to support transformers. An optional/variadic argument has been added, so it won't break existing code.

If you were using Mergo **before** April 6th 2015, please check your project works as intended after updating your local copy with `go get -u github.com/imdario/mergo`. I apologize for any issue caused by its previous behavior and any future bug that Mergo could cause (I hope it won't!) in existing projects after the change (release 0.2.0).

Donations

If Mergo is useful to you, consider buying me a coffee, a beer or making a monthly donation so I can keep building great free software. :heart_eyes:

Beerpay Beerpay

Mergo in the wild

- moby/moby
- kubernetes/kubernetes
- vmware/dispatch
- Shopify/themekit
- imdario/zas
- matcornic/hermes
- OpenBazaar/openbazaar-go
- kataras/iris
- michaelsauter/crane
- go-task/task
- sensu/uchiwa
- ory/hydra
- sisatech/vcli

- dairycart/dairycart
- projectcalico/felix
- resin-os/balena
- go-kivik/kivik
- Telefonica/govice
- supergiant/supergiant
- SergeyTsalkov/brooce
- soniah/dnsmadeeasy
- ohsu-comp-bio/funnel
- EagerIO/Stout
- lynndylanhurley/defsynth-api
- russross/canvasassignments
- rdegges/cryptly-api
- casualjim/exeggutor
- divshot/gitling
- RWJMurphy/gorl
- andrerocker/deploy42
- elwinar/rambler
- tmaiaroto/gopartman
- jfbus/impressionist
- Jmeyering/zealot
- godep-migrator/rigger-host
- Dronevery/MultiwaySwitch-Go
- thoas/picfit
- mantasmatelis/whooplist-server
- jnuthong/item_search
- bukalapak/snowboard

Installation

```
go get github.com/imdario/mergo

// use in your .go code
import (
    "github.com/imdario/mergo"
)
```

Usage

You can only merge same-type structs with exported fields initialized as zero value of their type and same-types maps. Mergo won't merge unexported (private) fields but will do recursively any exported one. It won't merge empty structs value as [they are not considered zero values](#) either. Also maps will be merged recursively except for structs inside maps (because they are not addressable using Go reflection).

```
if err := mergo.Merge(&dst, src); err != nil {
    // ...
}
```

Also, you can merge overwriting values using the transformer `WithOverride`.

```
if err := mergo.Merge(&dst, src, mergo.WithOverride); err != nil {
    // ...
}
```

Additionally, you can map a `map[string]interface{}` to a struct (and otherwise, from struct to map), following the same restrictions as in `Merge()`. Keys are capitalized to find each corresponding exported field.

```
if err := mergo.Map(&dst, srcMap); err != nil {
    // ...
}
```

Warning: if you map a struct to map, it won't do it recursively. Don't expect Mergo to map struct members of your struct as `map[string]interface{}`. They will be just assigned as values.

More information and examples in [godoc documentation](#).

Nice example

```
package main

import (
    "fmt"
    "github.com/imdario/mergo"
)

type Foo struct {
    A string
    B int64
}

func main() {
    src := Foo{
        A: "one",
        B: 2,
    }
    dest := Foo{
        A: "two",
    }
    mergo.Merge(&dest, src)
    fmt.Println(dest)
    // Will print
    // {two 2}
}
```

Note: if test are failing due missing package, please execute:

```
go get gopkg.in/yaml.v2
```

Transformers

Transformers allow to merge specific types differently than in the default behavior. In other words, now you can customize how some types are merged. For example, `time.Time` is a struct; it doesn't have zero value but IsZero can return true because it has fields with zero value. How can we merge a non-zero `time.Time`?

```
package main

import (
    "fmt"
    "github.com/imdario/mergo"
        "reflect"
        "time"
)

type timeTransfomer struct {
}

func (t timeTransfomer) Transformer(typ reflect.Type) func(dst, src reflect.Value)
error {
    if typ == reflect.TypeOf(time.Time{}) {
        return func(dst, src reflect.Value) error {
            if dst.CanSet() {
                isZero := dst.MethodByName("IsZero")
                result := isZero.Call([]reflect.Value{})
                if result[0].Bool() {
                    dst.Set(src)
                }
            }
            return nil
        }
    }
    return nil
}

type Snapshot struct {
    Time time.Time
    // ...
}

func main() {
    src := Snapshot{time.Now()}
    dest := Snapshot{}
    mergo.Merge(&dest, src, mergo.WithTransformers(timeTransfomer{}))
    fmt.Println(dest)
    // Will print
    // { 2018-01-12 01:15:00 +0000 UTC m=+0.000000001 }
}
```

Contact me

If I can help you, you have an idea or you are using Mergo in your projects, don't hesitate to drop me a line (or a pull request): [@im_dario](#)

About

Written by Dario Castañé.

License