-Struct

structstruct/json/xml/gconv

gconvStructstruct

```
// Struct maps the params key-value pairs to the corresponding struct
object's attributes.
// The third parameter `mapping` is unnecessary, indicating the mapping
rules between the
// custom key name and the attribute name(case sensitive).
//
// Note:
// 1. The `params` can be any type of map/struct, usually a map.
// 2. The `pointer` should be type of *struct/**struct, which is a pointer
to struct object
//    or struct pointer.
// 3. Only the public attributes of struct object can be mapped.
// 4. If `params` is a map, the key of the map `params` can be lowercase.
//    It will automatically convert the first letter of the key to
uppercase
//    in mapping procedure to do the matching.
//    It ignores the map key, if it does not match.
func Struct(params interface{}, pointer interface{}, mapping ...map[string]
string) (err error)
```

1. paramsstructmap
2. pointerstructstruct
3. mappingmapstrcutparamsmap

structhttps://pkg.go.dev/github.com/gogf/gf/v2/util/gconv

gconvstructstructmapping

1. struct ()
2. params
   - paramsmap struct
   - paramsstruct
   - structslice,map,strcut
3.

(mapstruct)

1. mapping mapping key strcut

2.tagtag params key

taggconv gconv, param, c, p, json tag

3.

4.gconvparams key

:

key:

Content Menu

mapstruct

```
map    struct
name       Name           match
Email      Email          match
nickname   NickName       match
NICKNAME   NickName       match
Nick-Name  NickName       match
nick_name  NickName       match
nick name  NickName       match
NickName   Nick_Name      match
Nick-name  Nick_Name      match
nick_name  Nick_Name      match
nick name  Nick_Name      match
```

pointer**structStructstruct

```
package main

import (
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/util/gconv"
)

func main() {
        type User struct {
                Uid  int
                Name string
        }
        params := g.Map{
                "uid":  1,
                "name": "john",
        }
        var user *User
        if err := gconv.Struct(params, &user); err != nil {
                panic(err)
        }
        g.Dump(user)
}
```

```
{
    Uid:  1,
    Name: "john",
}
```

Struct

structembeddedparamsstruct

```
package main

import (
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/util/gconv"
)

func main() {
        type Ids struct {
                Id         int    `json:"id"`
                Uid        int    `json:"uid"`
        }
        type Base struct {
                Ids
                CreateTime string `json:"create_time"`
        }
        type User struct {
                Base
                Passport   string `json:"passport"`
                Password   string `json:"password"`
                Nickname   string `json:"nickname"`
        }
        data := g.Map{
                "id"          : 1,
                "uid"         : 100,
                "passport"    : "john",
                "password"    : "123456",
                "nickname"    : "John",
                "create_time" : "2019",
        }
        user := new(User)
        gconv.Struct(data, user)
        g.Dump(user)
}
```

```
{
    Id:         1,
    Uid:        100,
    CreateTime: "2019",
    Passport:   "john",
    Password:   "123456",
    Nickname:   "John",
}
```

1

```
package main

import (
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/util/gconv"
)

type User struct {
        Uid      int
        Name     string
        SiteUrl  string
        NickName string
        Pass1    string `c:"password1"`
        Pass2    string `c:"password2"`
}

func main() {
        var user *User

        //
        user = new(User)
        params1 := g.Map{
                "uid":       1,
                "Name":      "john",
                "site_url":  "https://goframe.org",
                "nick_name": "johng",
                "PASS1":     "123",
                "PASS2":     "456",
        }
        if err := gconv.Struct(params1, user); err == nil {
                g.Dump(user)
        }

        // struct tag
        user = new(User)
        params2 := g.Map{
                "uid":       2,
                "name":      "smith",
                "site-url":  "https://goframe.org",
                "nick name": "johng",
                "password1": "111",
                "password2": "222",
        }
        if err := gconv.Struct(params2, user); err == nil {
                g.Dump(user)
        }
}
```

Structmapstructstruct tagStructmap

```
{
    Uid:      1,
    Name:     "john",
    SiteUrl:  "https://goframe.org",
    NickName: "johng",
    Pass1:    "123",
    Pass2:    "456",
}
{
    Uid:      2,
    Name:     "smith",
    SiteUrl:  "https://goframe.org",
    NickName: "johng",
    Pass1:    "111",
    Pass2:    "222",
}
```

2

structstructnil

```
package main

import (
    "github.com/gogf/gf/v2/util/gconv"
    "github.com/gogf/gf/v2/frame/g"
    "fmt"
)

func main() {
    type Score struct {
        Name   string
        Result int
    }
    type User1 struct {
        Scores Score
    }
    type User2 struct {
        Scores *Score
    }

    user1  := new(User1)
    user2  := new(User2)
        scores := g.Map{
                "Scores": g.Map{
                        "Name":   "john",
                        "Result": 100,
                },
        }

    if err := gconv.Struct(scores, user1); err != nil {
        fmt.Println(err)
    } else {
        g.Dump(user1)
    }
    if err := gconv.Struct(scores, user2); err != nil {
        fmt.Println(err)
    } else {
        g.Dump(user2)
    }
}
```

```
{
    Scores: {
        Name:   "john",
        Result: 100,
    },
}
{
    Scores: {
        Name:   "john",
        Result: 100,
    },
}
```