-

Content Menu

-
-
  -
  - Struct
  - Map

https://pkg.go.dev/github.com/gogf/gf/v2/util/gvalid

```
type Validator
    func New() *Validator
    func (v *Validator) Assoc(assoc interface{}) *Validator
    func (v *Validator) Bail() *Validator
    func (v *Validator) Ci() *Validator
    func (v *Validator) Clone() *Validator
    func (v *Validator) Data(data interface{}) *Validator
    func (v *Validator) I18n(i18nManager *gi18n.Manager) *Validator
    func (v *Validator) Messages(messages interface{}) *Validator
    func (v *Validator) RuleFunc(rule string, f RuleFunc) *Validator
    func (v *Validator) RuleFuncMap(m map[string]RuleFunc) *Validator
    func (v *Validator) Rules(rules interface{}) *Validator
    func (v *Validator) Run(ctx context.Context) Error
```

1. New
2. Assoc
3. Bail
4. Ci
5. Run
6. I18nI18Ni18n
7. Datamapstruct
8. Rules[]stringmap
9. Messagesmap

```
gValidatorgg.Validator()g
```

```
var (
        err error
        ctx = gctx.New()
)
err = g.Validator().
        Rules("min:18").
        Data(16).
        Messages("").
        Run(ctx)
fmt.Println(err.Error())

// Output:
//
```

```
var (
        err  error
        ctx  = gctx.New()
        data = g.Map{
                "password": "123",
        }
)

err = g.Validator().Data("").Assoc(data).
        Rules("required-with:password").
        Messages("").
        Run(ctx)

fmt.Println(err.Error())
```

Struct

```
type User struct {
        Name string `v:"required#"`
        Type int    `v:"required#"`
}
var (
        err  error
        ctx  = gctx.New()
        user = User{}
        data = g.Map{
                "name": "john",
        }
)
if err = gconv.Scan(data, &user); err != nil {
        panic(err)
}
err = g.Validator().Assoc(data).Data(user).Run(ctx)
if err != nil {
        fmt.Println(err.(gvalid.Error).Items())
}

// Output:
// [map[Type:map[required:]]]
```

Map

```
params := map[string]interface{}{
        "passport":  "",
        "password":  "123456",
        "password2": "1234567",
}
rules := map[string]string{
        "passport":  "required|length:6,16",
        "password":  "required|length:6,16|same:password2",
        "password2": "required|length:6,16",
}
messages := map[string]interface{}{
        "passport": "|{min}{max}",
        "password": map[string]string{
                "required": "",
                "same":     "",
        },
}
err := g.Validator().Messages(messages).Rules(rules).Data(params).Run(gctx.
New())
if err != nil {
        g.Dump(err.Maps())
}
```

```
{
    "passport": {
        "length": "616",
        "required": ""
    },
    "password": {
        "same": ""
    }
}
```