gingvalid

GoFramegvalid40struct tagGo

gingin go-playground/validator

-
- json tag
-
- re_passwordpasswordemail

```
{"msg":{"email":"email","re_password":"re_passwordPassword"}}
```

if

GoFrame

gvalid

```
import "github.com/gogf/gf/util/gvalid"
```

40

```
// SignUpParam
type SignUpParam struct {
    Username   string `json:"username" form:"username" v:"
username@required|length:6,30#|:min:max"`   //
    Password   string `json:"password" form:"password" v:"
password@required|length:6,16#|:min:max"`       //
    RePassword string `json:"rePassword" form:"rePassword" v:"
rePassword@required|same:password#|"` //
    Nickname   string `json:"nickname" form:"nickname" v:"
nickname@required#"`                                          //
    Email      string `json:"email" form:"email" v:"email@required|email#|"
`                                     //
}
```

username

```
v:"username@required|length:6,30#|:min:max"
```

v

v ginbinding

username@required|length:6,30

- username
- @

Author

Content Menu

- required
- |
- length:6,30

#|:min:max

- #
- required
- |
- :min:max length:6,30:min:maxlength:6,306,30

```
[@][#]
```

-
- structmapstruct
-

```
// BindAndValid
func BindAndValid(c *gin.Context, params interface{}) *gvalid.Error {
    _ = c.ShouldBind(params) // err
    //
    if err := gvalid.CheckStruct(params, nil); err != nil {
        return err
    }
    return nil
}

// SignUp
func SignUp(c *gin.Context) {
    var (
        uParam model.SignUpParam
    )
    if err := BindAndValid(c, &uParam); err != nil {
        c.JSON(http.StatusOK, gin.H{
            "code": http.StatusInternalServerError,
            "data": nil,
            "msg":  err.Error(),
        })
        return
    }
    //  userDto
    c.JSON(http.StatusOK, gin.H{
        "code": http.StatusOK,
        "data": &userDto,
        "msg":  "",
    })

}
```

```
// SignUpParam
// 1.
if err := gvalid.CheckStruct(params, nil); err != nil {
        return err
}

// 2.
c.JSON(http.StatusOK, gin.H{
    "code": http.StatusInternalServerError,
    "data": nil,
    "msg":  err.Error(), //
})
```

json tag

gvalid.CheckStruct()errgvalid.Error

```
// Error is the validation error for validation result.
type Error struct {
    rules     []string          // Rules by sequence, which is used for
keeping error sequence.
    errors    ErrorMap          // Error map.
    firstKey  string            // The first error rule key(nil in
default).
    firstItem map[string]string // The first error rule value(nil in
default).
}
```

GoFramee.FirstItem()e.FirstString()

Postman

username

json

```
{
  "username": "ce",
  "nickname": "1",
  "password": "1234561",
  "rePassword": "1234561",
  "email": "sadasdasdsasd@qq.com"
}
```

```
{
    "code": 500,
    "data": null,
    "msg": "630"
}
```

json

```
{
  "username": "usernameceshi",
  "nickname": "1",
  "password": "1234561",
  "rePassword": "1234562",
  "email": "sadasdasdsasd@qq.com"
}
```

```
{
    "code": 500,
    "data": null,
    "msg": ""
}
```

json

```
{
  "username": "usernameceshi",
  "nickName": "1",
  "password": "1234561",
  "rePassword": "1234561",
  "email": "sadasdasdsasd@q@q.com"
}
```

```
{
    "code": 500,
    "data": null,
    "msg": ""
}
```

GoFrame

```
"msg": err.Error(), //
```

err.Error()err.FirstString()

json

```
{
  "username": "usernameceshi",
  "nickname": "1",
  "password": "1234561",
  "rePassword": "1234562",
  "email": "sadasdasdsasd@q@q.com"
}
```

```
{
    "code": 500,
    "data": null,
    "msg": ""
}
```

json

```
{
  "username": "usernameceshi",
  "nickname": "1",
  "password": "1234561",
  "rePassword": "1234561",
  "email": "sadasdasdsasd@q@q.com"
}
```

```
{
    "code": 500,
    "data": null,
    "msg": ""
}
```

GoFrameGoFrameGo()

GoFrame-