



GolangGoFrame

|

```
{{.value | Func1 | Func2}}
```

```
{{printf "nums is %s %d" (printf "%d %d" 1 2) 3}}
```

## and

```
{{and .X .Y .Z}}
```

and

## call

```
{{call .Field.Func .Arg1 .Arg2}}
```

call

1 2 error nil

## index

indexmap, slice, array, string

```
{{index .Maps "name"}}
```

## len

```
{{printf "The content length is %d" (.Content|len)}}
```

map, slice, array, string, chan

## not

not

```
{{if not .Var}}  
// (.Var, : nil, 0, "", 0slice/map)  
{{else}}  
// (.Var)  
{{end}}
```

### Content Menu

- [and](#)
- [call](#)
- [index](#)
- [len](#)
- [not](#)
- [or](#)
- [print](#)
- [printf](#)
- [println](#)
- [urlquery](#)
- [eq / ne / lt / le / gt / ge](#)
  - 
  -

or

```
{{or .X .Y .Z}}
```

or

print

fmt.Sprint

printf

fmt.Sprintf

println

fmt.Sprintln

urlquery

```
{{urlquery "http://johng.cn"}}
```

```
http%3A%2F%2Fjohng.cn
```

eq / ne / lt / le / gt / ge

if

```
`eq`: arg1 == arg2
`ne`: arg1 != arg2
`lt`: arg1 < arg2
`le`: arg1 <= arg2
`gt`: arg1 > arg2
`ge`: arg1 >= arg2
```

eq

```
{{eq arg1 arg2 arg3 arg4}}
```

:

```
arg1==arg2 || arg1==arg3 || arg1==arg4 ...
```

if

```
{{if eq true .Var1 .Var2 .Var3}}...{{end}}
```

```
{{if lt 100 200}}...{{end}}
```

```
{{if .Var}}  
// (.Var)  
{{else}}  
// (.Var, : nil, 0, "", 0slice/map)  
{{end}}
```

GoFrameeq/ne/lt/le/gt/ge

```
{{eq 1 "1"}}
```

```
panic: template: at <eq 1 "1">: error calling eq: incompatible types for  
comparison
```

panic

GoFrame

GoFrame

```

package main

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

func main() {
    tplContent := `
eq:
eq "a" "a": {{eq "a" "a"}}
eq "1" "1": {{eq "1" "1"}}
eq 1 "1": {{eq 1 "1"}}

ne:
ne 1 "1": {{ne 1 "1"}}
ne "a" "a": {{ne "a" "a"}}
ne "a" "b": {{ne "a" "b"}}

lt:
lt 1 "2": {{lt 1 "2"}}
lt 2 2 : {{lt 2 2 }}
lt "a" "b": {{lt "a" "b"}}

le:
le 1 "2": {{le 1 "2"}}
le 2 1 : {{le 2 1 }}
le "a" "a": {{le "a" "a"}}

gt:
gt 1 "2": {{gt 1 "2"}}
gt 2 1 : {{gt 2 1 }}
gt "a" "a": {{gt "a" "a"}}

ge:
ge 1 "2": {{ge 1 "2"}}
ge 2 1 : {{ge 2 1 }}
ge "a" "a": {{ge "a" "a"}}
`

    content, err := g.View().ParseContent(context.TODO(), tplContent,
nil)
    if err != nil {
        panic(err)
    }
    fmt.Println(content)
}

```

```
eq:
eq "a" "a": true
eq "1" "1": true
eq 1 "1": true
```

```
ne:
ne 1 "1": false
ne "a" "a": false
ne "a" "b": true
```

```
lt:
lt 1 "2": true
lt 2 2 : false
lt "a" "b": true
```

```
le:
le 1 "2": true
le 2 1 : false
le "a" "a": true
```

```
gt:
gt 1 "2": false
gt 2 1 : true
gt "a" "a": false
```

```
ge:
ge 1 "2": false
ge 2 1 : true
ge "a" "a": true
```