# 1、template快速入门

text/template包实现了用于生成文本输出的数据驱动模板。了解基本使用即可。

#### \* Parse函数

```
func TestTamplate1() {
   name := "vision"
   tmp := "Hello, {{.}}}"
   t := template.New("test")参数是模板包
   fmt.Println(t)
   t2, err := t.Parse(tmp)
   if err != nil {
       log.Fatalf("Parse error: %v", err)
   t2.Execute(os.Stdout, name)
func TestTamplate2() {
   // 预定义的结构体
   type Inventory struct {
       Material string
       Count uint
   sweaters := Inventory{"wool", 17}
  tmpl, err := template.New("test").Parse("{{.Count}} items are made of {{.Material}}")
   if err != nil {
       panic(err)
   err = tmpl.Execute(os.Stdout, sweaters)
   if err != nil {
       panic(err)
                                                                       CSDN @Golang-Study
```

#### \* ParseFiles函数

模板引擎在进行替换的时候,是完全按照文本格式进行替换的。除了需要评估和替换的地方,所有的行分隔符,空格等等空白都原样保留。所以,对于要解析的内容,不要随意缩进、随意换行。

```
index.html ×
templates > ♦ index.html > ♦ html > ♦ body
     <!DOCTYPE html>
      <html lang="en">
         <meta charset="UTF-8">
          <meta http-equiv="X-UA-Compatible" content="IE=edge">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <title>Document</title>
      </head>
          {{.}}
 13
          _{{23 -}} + {{- 32}}
          {{23}} / {{32}}
          {{23}} * {{32}}
          {{23}} - {{32}}
      </body>
 19 </html>
 Ocument
← → C ① localhost:8080/hello
Output to Html 23+32 23 / 32 23 * 32 23 - 32
                                                          CSDN @Golang-Study
```

### 2、基本介绍

html/template包实现了数据驱动的模板,用于生成可防止代码注入的安全的HTML内容。它提供了和text/template包相同的接口,Go语言中输出HTML的场景都应使用html/template这个包。

### 3、Go语言的模板引擎

Go语言内置了文本模板引擎text/template和用于HTML文档的html/template。它们的作用机制可以简单归纳如下:

- 1、模板文件通常定义为.tmpl和.tpl为后缀(也可以使用其他的后缀),必须使用UTF8编码。
- 2、模板文件中使用{{和}}包裹和标识需要传入的数据。
- 3、传给模板这样的数据就可以通过点号(.)来访问,如果数据是复杂类型的数据,可以通过{ { .FieldName }}来访问它的字段。

4、除{{和}}包裹的内容外,其他内容均不做修改原样输出。

# 4、模板引擎的使用

Go语言模板引擎的使用可以分为三部分: 定义模板文件、解析模板文件和模板渲染。

定义好了模板文件之后,可以使用下面的常用方法去解析模板文件,得到模板对象:

```
func (t *Template) Parse(src string) (*Template, error)
func ParseFiles(filenames ...string) (*Template, error)
func ParseGlob(pattern string) (*Template, error)
```

也可以使用 func New(name string) \*Template 函数创建一个名为name的模板 , 然后对其调用上面的方法(Parse)去解析模板字符串或模板文件。

#### \*\*\* 模板引擎

渲染模板简单来说就是 使用数据去填充模板 ,实际上可能会复杂很多。

```
func (t *Template) Execute(wr io.Writer, data interface{}) error
func (t *Template) ExecuteTemplate(wr io.Writer, name string, data interface{})
error
```

```
template.go X
emplates > 🚥 template.go > 😭 TestTemplate3
                                                                                                                                    templates > 

hello.tmpl >
                                                                                                                                             <!DOCTYPE html>
<html lang="zh-CN">
                                                                                                                                             <head>
            if err != nil {

fmt.Println("create template failed, err:", err)
                                                                                                                                                  content charset="UTF-8">
cmeta charset="UTF-8">
cmeta name="viewport" content="width-device-width, initial-scale=1.0">
cmeta http-equiv="X-UA-Compatible" content="ie=edge">
ctitle>Hello</title>
            // 利用给定数据渲染模板,并将结果写入w
tmpl.Execute(w, "HELLO")
                                                                                                                                            <body>
Hello {{.}}
</body>
</html>
       func TestTemplate3() {
            http.HandleFunc("/", sayHello)
err := http.ListenAndServe(":9090", nil)
if err != nil {
                                                                                                  Hello
                   err != nil {
fmt.Println("HTTP server failed,err:", err)
                                                                                                 ← → C (i) localhost:9090
                                                                                                 Hello HELLO
                                                                                                                                                                                                                      CSDN @Golang-Study
```

### 5、模板语法

#### 5.1 {{.}}

模板语法都包含在{{和}}中间,其中{{.}}中的点表示当前对象。 当传入一个结构体对象时,可以根据.来访问结构体的对应字段。 当传入的变量是map时,也可以在模板文件中通过.根据key来取值。

```
template.go × ••• main_test.go
                                                                                                                                                                                                                                                                                                   templates > ≡ helio.tmpl > ...

1 <!DOCTYPE html>
2 <html lang="zh-CN">
    plates > \infty template.go > 😚 sayHello
                                                                                                                                                                                                                                                                                                                         <head>

«meta charset="UTF-8">
«meta name-"viewport" content="width-device-width, initial-scale=1.0">
«meta http-equiv="X-UA-Compatible" content="ie=edge">
<iitle>Mello</title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title></title>
                            tmpl, err := template.ParseFiles("F:/tools/golang/goweb/templates/hello.tmpl")
                                                                                                                                                                                                                                                                                                                              <h1>Hello {{.Name}}</h1>
<h1>性别: {{.Gender}}</h1>
<h1>年龄: {{.Age}}</h1></h1>
                          // 利用語定数据 這些模板,并将结果写入W
tmpl.Execute(w, User(
Name: "Goldang",
Gender: "男",
Age: 19,
                                                                                                                                                                                                           6 Hello
                                                                                                                                                                                                          ← → C ① localhost:9090
               func TestTemplate3() {
                                                                                                                                                                                                        Hello Golang
                          http.HandleFunc("/", sayHello)
err := http.ListenAndServe(":9090", nil)
                                                                                                                                                                                                        性别:男
                                        fmt.Println("HTTP server failed,err:", err)
                                                                                                                                                                                                        年龄: 19
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            CSDN @Golang-Study
```

#### 5.2 模板注释

```
{{/* a comment */}}
```

注释,执行时会忽略。可以多行。注释不能嵌套,并且必须紧贴分界符始止。

#### 5.3 pipeline管道

pipeline是指产生数据的操作。比如{{.}}、{{.Name}}等。Go的模板语法中支持使用管道符号|链接多个命令,用法和unix下的管道类似:|前面的命令会将运算结果(或返回值)传递给后一个命令的最后一个位置。

注意:并不是只有使用了|才是pipeline。Go的模板语法中,pipeline的概念是传递数据,只要能产生数据的,都是pipeline。

```
olates > 🥯 template.go > 🗘 sayHello
   func TestHtmlTemplate() {
                                                                                                          <html lang="en":
       // 创建一个服务器对象
server := &http.Server{
                                                                                                             <meta charset="UTF-8">
                                                                                                              <meta http-equiv="X-UA-Compatible" content="IE=edg
<meta name="viewport" content="width=device-width,</pre>
       http.HandleFunc("/hello", func(w http.ResponseWriter, r *http.Request) {
    t, _ := template_ParseFiles("F:/tools/golang/goweb/templates/index.html")
            t.Execute(w, "<b>Output to Html</b>")
                                                                                                         server.ListenAndServe()
 S Document
← → C ① localhost:8080/hello 🔊 🖈 🗊 🔲 😩 🗄
Output to Html
output"
"output"
"output
"output
output"
"output"
                                                                                                                                              CSDN @Golang-Study
```

#### 5.4 变量

还可以在模板中声明变量,用来保存传入模板的数据或其他语句生成的结果。具体语法如下:

```
$obj := {{.}}
```

其中 \$obj是变量的名字,在后续的代码中就可以使用该变量了。

```
nplates > 를 hello.tmpl 2
                                                                                                  <!DOCTYPE html>
<html lang="zh-CN">
    Gender string
                                                                                                       <meta charset="UTF-8">
                                                                                                       <meta name="viewport" content="width=device-width, initial-sc
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Hello</title>
func sayHello(w http.ResponseWriter, r *http.Request) {
    tmpl, err := template.ParseFiles("F:/tools/golang/goweb/templates/")
         fmt.Println("create template failed, err:", err)
                                                                                                  <body>
                                                                                                    {{$obj := .}}
<h2>{{$obj}}</h2>
    tmpl.Execute(w, 999)
                                                                                                  </html>
func TestTemplate3() {
                                                                            Hello
    http.HandleFunc("/", sayHello)
err := http.ListenAndServe(":9090", nil)
                                                                            \leftarrow \rightarrow C ① localhost:9090
                                                                                                                                    🕑 🖈 🗊 🔲 各
                                                                          999
                                                                                                                                                      CSDN @Golang-Study
```

### 5.5 移除空格

在使用模板语法的时候会不可避免的引入一下空格或者换行符,这样模板最终渲染出来的内容可能就和想的不一样,这个时候可以使用{{-语法去除模板内容左侧的所有空白符号,使用-}}去除模板内容右侧的所有空白符号。

```
{{- .Name -}}
```

注意: - 要紧挨{{和}},同时与模板值之间需要使用空格分隔。

#### 5.6 条件判断

Go模板语法中的条件判断有以下几种:

```
{{if pipeline}} T1 {{end}}

{{if pipeline}} T1 {{else}} T0 {{end}}

{{if pipeline}} T1 {{else if pipeline}} T0 {{end}}
```

pipeline为false的情况是各种数据对象为0值:数值0,指针或者接口是nil,数组、切片、map或者string则是len为0.

#### 5.7 比较运算符

布尔函数会将任何类型的零值视为假, 其余视为真。

```
      eq
      如果arg1 == arg2则返回真

      ne
      如果arg1 != arg2则返回真

      lt
      如果arg1 < arg2则返回真</td>

      le
      如果arg1 <= arg2则返回真</td>

      gt
      如果arg1 > arg2则返回真

      ge
      如果arg1 >= arg2则返回真
```

#### 5.8 条件判断和比较综合使用

```
templates > ≡ hello.tmpl

1 <!DOCTYPE html>
                                                                                                                 <html lang="zh-CN">
           Name string
Gender string
                                                                                                                 <head>
                                                                                                                      <meta name="viewport" content="width=device-width, initial-scale=1
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Hello</title>
       func sayHello(w http.ResponseWriter, r *http.Request) {
            tmpl, err := template.ParseFiles("F:/tools/golang/goweb/templates/")
            if err != nil {
   fmt.Println("create template failed, err:", err)
                                                                                                                    {{$Age := .Age}}
{{if (ge $Age 18)}}
<h3>成年了!!!!!!</h3>
{{else}}
72
73
74
75
76
77
78
79
                                                                                                                     <h3>还未成年!!!!!</h3>
            tmpl.Execute(w, User{
                                                                                                                     {{end}}
                 Name: "Go",
Gender: "Male",
                                                                                                                </body>
                                                                                     Hello
       func TestTemplate3() {
                                                                                    \leftarrow \rightarrow C ① localhost:9090
                                                                                                                                                 Ŀ ☆ 🖈 ₹ 🗖 🚨 🗄
            http.HandleFunc("/", sayHello)
err := http.ListenAndServe(":9090", nil)
                                                                                   成年了!!!!!!
                 fmt.Println("HTTP server failed,err:", err)
                                                                                                                                                                               CSDN @Golang-Study
```

#### 5.9 range循环迭代

Go的模板语法中 使用range关键字进行遍历,有以下两种写法,其中pipeline的值必须是 数组、切片、字典或者通道。

```
{{range pipeline}} T1 {{end}}
如果pipeline的值其长度为0,不会有任何输出
{{range pipeline}} T1 {{else}} T0 {{end}}
如果pipeline的值其长度为0,则会执行T0。
```

```
o template.go X ••• main test.go
      type User struct {
                                                                                                                   <html lang="zh-CN">
          Gender string
                                                                                                                        <meta charset="UTF-8">
                                                                                                                        <meta name="viewport" content="width=device-width,
<meta http-equiv="X-UA-Compatible" content="ie=edge</pre>
      func sayHello(w http.ResponseWriter, r *http.Request) {
                                                                                                                   </head>
                                                                                                                   <body>
           tmpl, err := template.ParseFiles("F:/tools/golang/goweb/templates/hello.tmpl")
                                                                                                                     {{$obj := .}}
{{range $x := $obj}}
<h3>{{println $x}}</h3>
{{- end}}
           // 利用给定数据渲染模板,并将结果写入w
tmpl.Execute(w, []string{"Java", "Golang", "Python", "C++"})
                                                                                                                   </html>
      func TestTemplate3() {
                                                                                    Hello
           http.HandleFunc("/", sayHello)
err := http.ListenAndServe(":9090", nil)
                                                                                   ← → C © localhost:9090
                                                                                                                                          환 ☆ ★ 킥 □ ♣ :
           if err != nil {
   fmt.Println("HTTP server failed,err:", err)
                                                                                  Java
                                                                                  Golang
                                                                                  Python
                                                                                  C++
    输出 调试控制台 <mark>终端</mark> COMMENTS
                                                                                                                                                          CSDN @Golang-Study
       F:/tools/golang1.19/src/testing/testing.go:1493 +0x35f
```

#### 5.10 with ... end

```
{{with pipeline}} T1 {{end}}
如果pipeline为empty不产生输出,否则将dot设为pipeline的值并执行T1。不修改外面的dot。
{{with pipeline}} T1 {{else}} T0 {{end}}
如果pipeline为empty,不改变dot并执行T0,否则dot设为pipeline的值并执行T1。
```

```
templates > ≡ hello.tmpl

1 <!DOCTYPE html>
type User struct {
   Name string
   Gender string
                                                                                                   <html lang="zh-CN">
                                                                                                   <head>
                                                                                                      Age
    tmpl, err := template.ParseFiles("F:/tools/golang/goweb/templates/hello.tmpl")
                                                                                                     {{$obj := .}}
{{range $x := $obj}}
<h3>
                                                                                                          {{with $x}} {{println .}} {{else}} {{"空值"}} {{end}}
    ,// 利用给定数据渲染模板,并将结果写入W
tmpl.Execute(w, []string("Java", "Golang", "Python", "C++", "", ""})
                                                                                                  {{- end}}
</body>
</html>
func TestTemplate3() {
   http.HandleFunc("/", sayHello)
err := http.ListenAndServe(":9090", nil)
                                                                      Hello
   if err != nil {
    fmt.Println("HTTP server failed,err:", err)
                                                                                                                     🕑 🌣 🗯 🗆 🚨 :
                                                                      ← → C ① localhost:9090
                                                                     Java
                                                                     Golang
                                                                     Python
                                                                                                                                                             ≥ go + ∨ □ ii ^
                                                                     C++
                                                                     空值
                                                                     空值
```

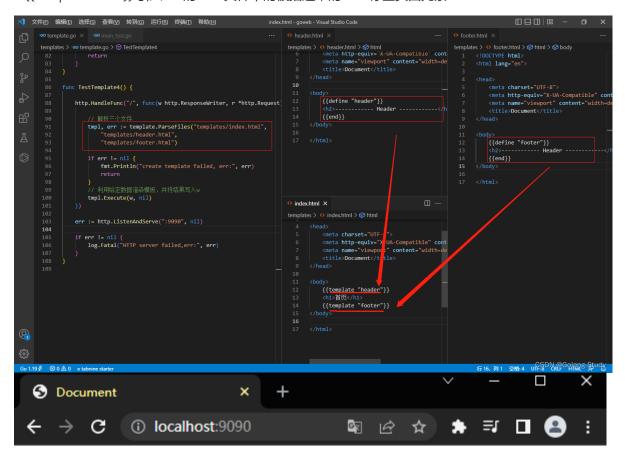
#### 5.11 内置函数

```
and
   函数返回它的第一个empty参数或者最后一个参数;
   就是说"and x y"等价于"if x then y else x"; 所有参数都会执行;
or
   返回第一个非empty参数或者最后一个参数;
   亦即"or x y"等价于"if x then x else y"; 所有参数都会执行;
not
   返回它的单个参数的布尔值的否定
len
   返回它的参数的整数类型长度
index
   执行结果为第一个参数以剩下的参数为索引/键指向的值;
   如"index x 1 2 3"返回x[1][2][3]的值;每个被索引的主体必须是数组、切片或者字典。
print
   即fmt.Sprint
printf
   即fmt.Sprintf
println
   即fmt.Sprintln
htm1
   返回与其参数的文本表示形式等效的转义HTML。
   这个函数在html/template中不可用。
urlquery
   以适合嵌入到网址查询中的形式返回其参数的文本表示的转义值。
   这个函数在html/template中不可用。
```

```
js
返回与其参数的文本表示形式等效的转义JavaScript。
call
执行结果是调用第一个参数的返回值,该参数必须是函数类型,其余参数作为调用该函数的参数;
如"call .X.Y 1 2"等价于go语言里的dot.X.Y(1, 2);
其中Y是函数类型的字段或者字典的值,或者其他类似情况;
call的第一个参数的执行结果必须是函数类型的值(和预定义函数如print明显不同);
该函数类型值必须有1到2个返回值,如果有2个则后一个必须是error接口类型;
如果有2个返回值的方法返回的error非nil,模板执行会中断并返回给调用模板执行者该错误;
```

#### 5.12 嵌套template

- \* {{define "xxx"}} html 标签页面元素 {{end}}
- \* {{template "xxx"}} 引入xxx的html文件中的被括起来的html标签页面元素



----- Header -----

# 首页

------ Header ------

# 5.13 text/template与html/tempalte的区别

html/template针对的是需要返回HTML内容的场景,在模板渲染过程中会对一些有风险的内容进行转义,以此来防范跨站脚本攻击。