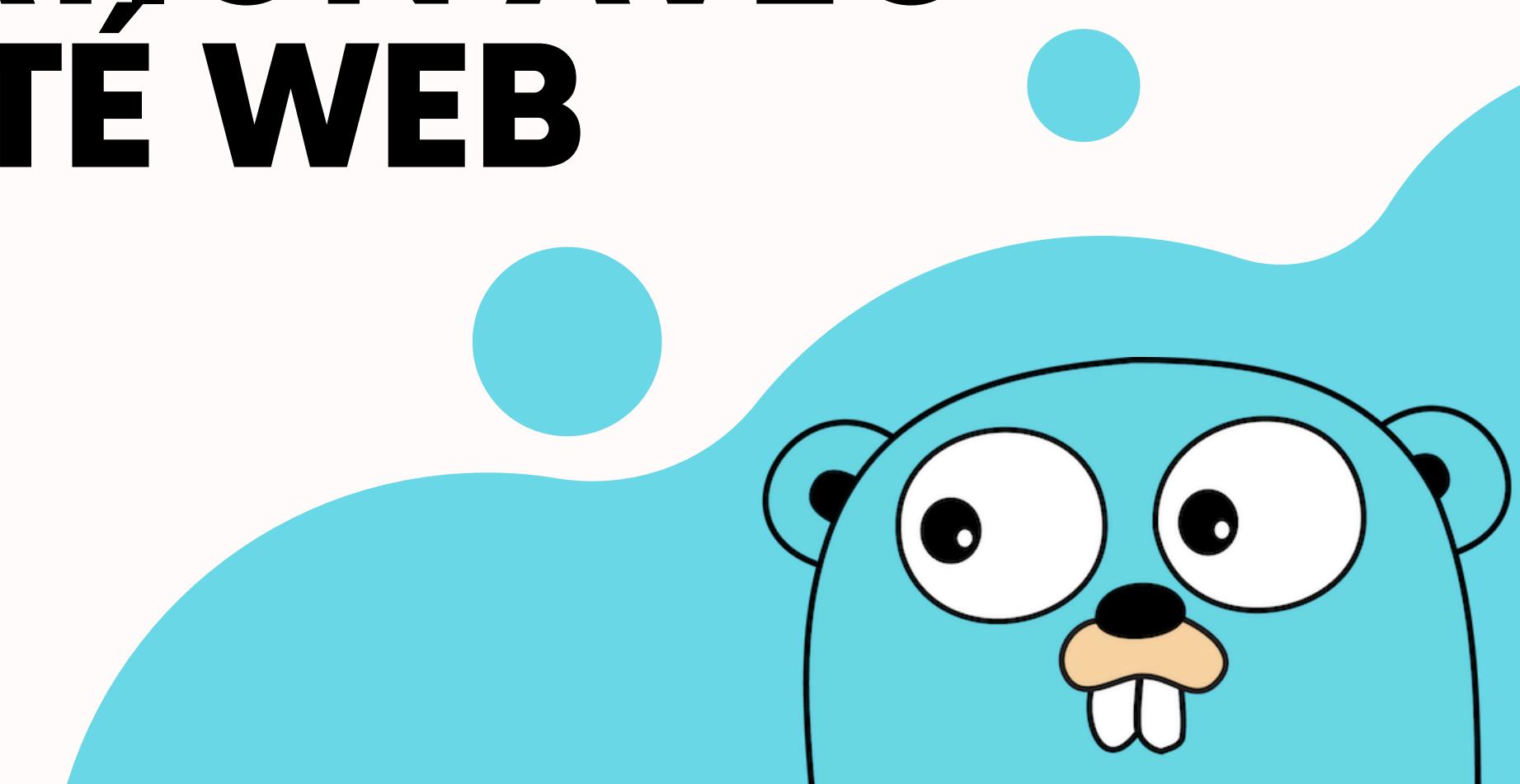
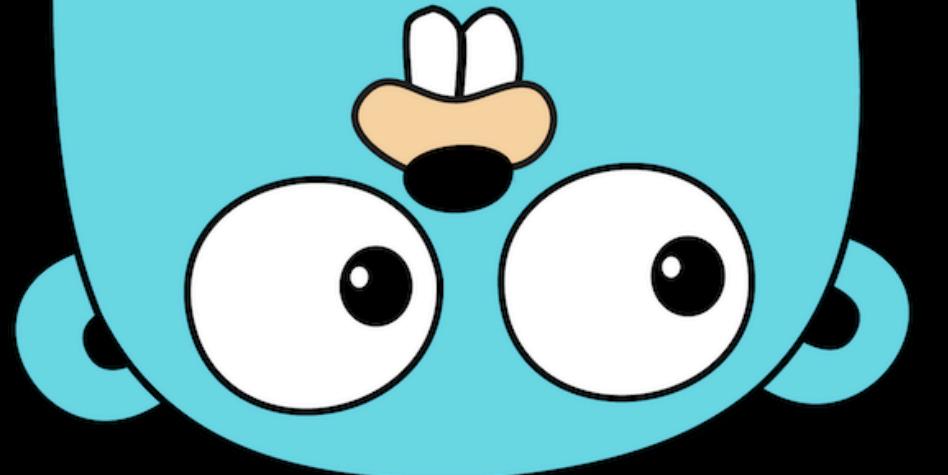


CAPSULE

LA PROGRAMMATION AVEC GOLANG ORIENTÉ WEB

Par Cyril RODRIGUES





01 COMMENT FONCTIONNE UN SITE WEB ?

02 IMPLÉMENTATION D'UN SERVEUR HTTP

03 LES TEMPLATES

04 LES TEMPLATES & LES ACTIONS

05 UTILISATION DE RESSOURCES STATIQUES

06 TRAITEMENT ET UTILISATION DE FORMULAIRES

07 DES SUGESTIONS POUR APPROFONDIR

LES TEMPLATES & LES ACTIONS

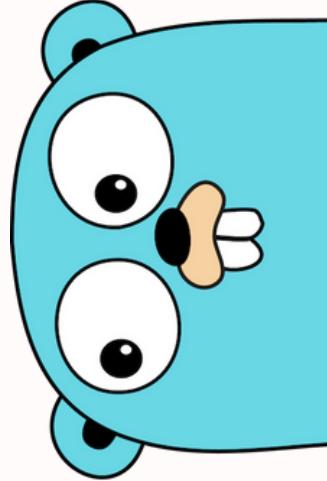


LES ACTIONS UTILISABLES AVEC LES TEMPLATES

Les **Templates** permettent d'insérer des éléments dynamiques, appelés **actions**, qui seront remplacés par des valeurs lors de l'exécution. Les actions sont délimitées par des doubles accolades : {{ action }}. Le moteur prend en charge différentes actions telles que l'utilisation de variables, les conditions, les boucles, la nomination de Templates, ou l'imbrication de Templates.

Note : Les données sont transmises lors de l'exécution du Template sous forme d'une structure de données avec des champs correspondant aux éléments dynamiques utilisés dans le Template.

LA NOMINATION D'UN TEMPLATE



Pour nommer un Template, il faut utiliser la combinaison de deux actions {{define 'NomDuTemplate'}} et {{end}} qui englobe le code source du Template (soit au début et à la fin). Le nom du Template doit être une chaîne de caractères indiqué après le mot-clé define. Voici un exemple :

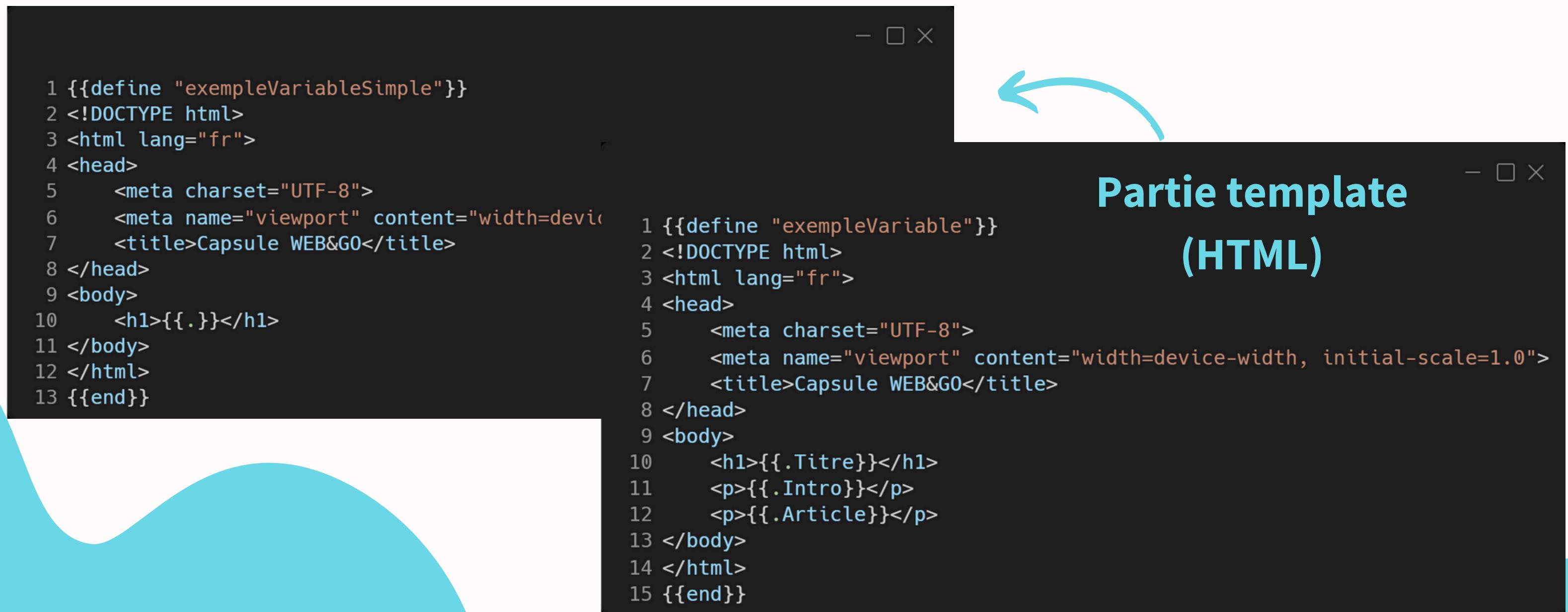
```
1 {{define "index"}}
2 <!DOCTYPE html>
3 <html lang="fr">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Exemple Template</title>
8 </head>
9 <body>
10  <h1>Hello world !</h1>
11 </body>
12 </html>
13 {{end}}
```

Début du template nommé : index

Fin

L'UTILISATION D'UNE VARIABLE

Il est possible d'utiliser des variables pour afficher leur contenu et rendre l'affichage dynamique. Pour utiliser une variable, on emploie l'action suivante : {{.NomChamp}} ou simplement {{.}} lorsqu'une seule variable est transmise au template. Voici des exemples d'utilisation dans les deux cas :



The diagram illustrates the use of variables in a template. It shows two code snippets in separate windows, connected by a blue curved arrow pointing from the first to the second.

Partie template (HTML)

```
1 {{define "exempleVariableSimple"}}
2 <!DOCTYPE html>
3 <html lang="fr">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Capsule WEB&GO</title>
8 </head>
9 <body>
10  <h1>{{.}}</h1>
11 </body>
12 </html>
13 {{end}}
```

```
1 {{define "exempleVariable"}}
2 <!DOCTYPE html>
3 <html lang="fr">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Capsule WEB&GO</title>
8 </head>
9 <body>
10  <h1>{{.Titre}}</h1>
11  <p>{{.Intro}}</p>
12  <p>{{.Article}}</p>
13 </body>
14 </html>
15 {{end}}
```

L'UTILISATION D'UNE VARIABLE

```
- □ ×

1 func main() {
2     // Récupération de l'ensemble des templates
3     temp, err := template.ParseGlob("./templates/*.html")
4     if err != nil {
5         fmt.Println(fmt.Sprint("ERREUR => %s", err.Error()))
6         return
7     }
8
9     // Déclaratrion d'une structure correspondant aux champs du template
10    type PageVariable struct {
11        Titre string
12        Intro string
13        Article string
14    }
15
16    http.HandleFunc("/variable/structure", func(w http.ResponseWriter, r *http.Request) {
17        dataPage := PageVariable{"Les bases du WEB",
18            "Le web est devenu un incontournable pour....",
19            "Pour commencer cet article sur les bases du WEB nous allons aborder la culture du WEB"}
20        // Appel du template nommé "exempleVariable" avec les données stockées dans la variable dataPage
21        temp.ExecuteTemplate(w, "exempleVariable", dataPage)
22    })
23
24    http.HandleFunc("/variable/simple", func(w http.ResponseWriter, r *http.Request) {
25        titre := "Hello les développeurs 👋"
26        // Appel du template nommé "exempleVariableSimple" avec les données stockées dans la variable titre
27        temp.ExecuteTemplate(w, "exempleVariableSimple", titre)
28    })
29
30    // Initialisation du serveur HTTP
31    http.ListenAndServe("localhost:8080", nil)
32 }
```



Partie Golang

L'UTILISATION DE CONDITIONS

Il est possible d'utiliser des conditions if/else pour n'exécuter certaines parties du template qu'en fonction de conditions. Rappel une condition est de type boolean ! Voici un exemple d'utilisation :

```
1 {{define "exempleConditionSimple"}}
2 <!DOCTYPE html>
3 <html lang="fr">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Capsule WEB&GO</title>
8 </head>
9 <body>
10  {{if .Check}}
11    <p>La condition du champ 'Check' est vraie</p>
12  {{end}}
13
14  {{if .CheckOwner}}
15    <p>Le propriétaire est vérifié</p>
16  {{else}}
17    <p>Oupsss le propriétaire n'est pas vérifié...</p>
18  {{end}}
19 </body>
20 </html>
21 {{end}}
```



Partie template
(HTML)

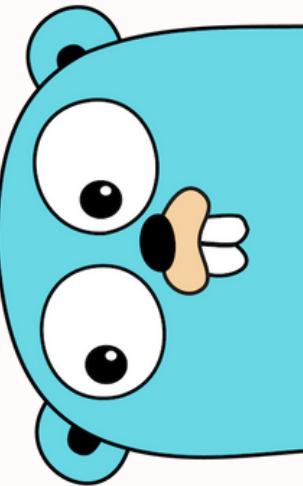
L'UTILISATION DE CONDITIONS



Partie Golang

```
1 func main() {
2     // Récupération de l'ensemble des templates
3     temp, err := template.ParseGlob("./templates/*.html")
4     if err != nil {
5         fmt.Println(fmt.Sprint("ERREUR => %s", err.Error()))
6         return
7     }
8
9     // Déclaratrion d'une structure correspondant aux champs du template
10    type PageConditionSimple struct {
11        Check      bool
12        CheckOwner bool
13    }
14
15    http.HandleFunc("/condition/simple", func(w http.ResponseWriter, r *http.Request) {
16        dataPage := PageConditionSimple{true, false}
17        // Appel du template nommé "exempleConditionSimple" avec les données stockées dans la variable dataPage
18        temp.ExecuteTemplate(w, "exempleConditionSimple", dataPage)
19    })
20
21    // Initialisation du serveur HTTP
22    http.ListenAndServe("localhost:8080", nil)
23 }
```

L'UTILISATION DE CONDITIONS AVEC OPÉRATEUR DE CONDITION

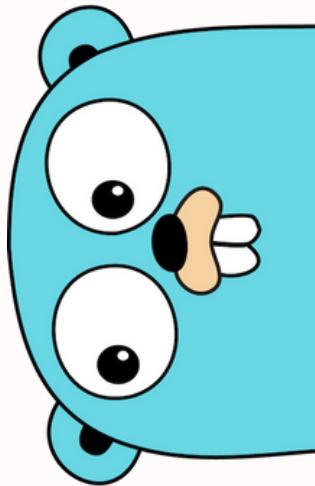


Les templates supportent les opérateurs de comparaison pour effectuer des tests entre deux valeurs. Cependant, il est recommandé de gérer les conditions complexes avec Go et de transmettre le résultat sous forme de variable au template afin d'éviter les bugs.

- □ ×

```
1 // =====
2 // eq : permet de tester l'égalité entre deux valeurs, équivalant au « == »
3 // ne : permet de tester que les deux valeurs sont différentes, équivalant à « != »
4 // lt : permet de tester que la première valeur est inférieure à la deuxième, équivalant à « < »
5 // le : permet de tester que la première valeur est inférieure ou bien égale à la deuxième, équivalant à « <= »
6 // gt : permet de tester que la première valeur est supérieure à la deuxième, équivalant à « > »
7 // ge : permet de tester que la première valeur est inférieure ou bien égale à la deuxième, équivalant à « >= »
8 // =====
```

L'UTILISATION DE CONDITIONS AVEC OPÉRATEUR DE CONDITION

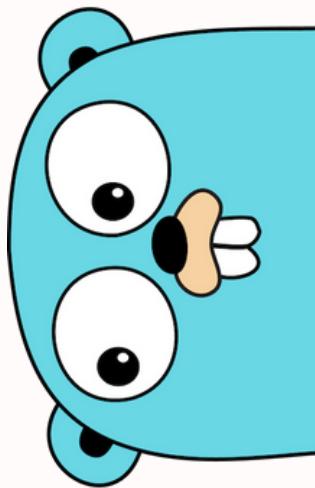


```
1 {{define "exempleConditionComplexe"}}
2 <!DOCTYPE html>
3 <html lang="fr">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Capsule Go&Web</title>
8 </head>
9 <body>
10  {{if le .NbrGuest 10}}
11    <p>L'évent est une petite soirée</p>
12  {{else if le .NbrGuest 20}}
13    <p>L'évent est une soirée</p>
14  {{else}}
15    <p>Aucune information sur le format de l'évent</p>
16  {{end}}
17
18  {{if .CheckList}}
19    <p>La liste est vérifié</p>
20  {{end}}
21 </body>
22 </html>
23 {{end}}
```



Partie template
(HTML)

L'UTILISATION DE CONDITIONS AVEC OPÉRATEUR DE CONDITION



```
1 func main() {
2     // Récupération de l'ensemble des templates
3     temp, err := template.ParseGlob("./templates/*.html")
4     if err != nil {
5         fmt.Println(fmt.Sprint("ERREUR => %s", err.Error()))
6         return
7     }
8
9     // Déclaratrion d'une structure correspondant aux champs du template
10    type PageConditionComplexe struct {
11        NbrGuest int
12        CheckList bool
13    }
14
15    http.HandleFunc("/condition/complexe", func(w http.ResponseWriter, r *http.Request) {
16        dataPage := PageConditionComplexe{12, false}
17        // Utilisation d'une condition complexe
18        dataPage.CheckList = 10 < dataPage.NbrGuest && dataPage.NbrGuest < 15
19        // Appel du template nommé "exempleConditionComplexe" avec les données stockées dans la variable dataPage
20        temp.ExecuteTemplate(w, "exempleConditionComplexe", dataPage)
21    })
22
23    // Initialisation du serveur HTTP
24    http.ListenAndServe("localhost:8080", nil)
25 }
```

Partie Golang

L'UTILISATION DE BOUCLES

Les templates permettent aussi d'itérer sur des collections comme des slices, maps ou arrays grâce à la boucle range. Les boucles sont principalement utilisées sur des collections de types primitifs ou locaux. Voici des exemples :

Partie template
(HTML)

```
1 {{define "exempleBoucleLocal"}}
2 <!DOCTYPE html>
3 <html lang="fr">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Capsule WEB&GO</title>
8 </head>
9 <body>
10  <h1>{{.Title}}</h1>
11  {{range .Users}}
12    <p>{{.FirstName}} {{.LastName}}</p>
13  {{end}}
14
15 </body>
16 </html>
17 {{end}}
```

L'UTILISATION DE BOUCLES

Partie Golang



```
1 func main() {
2     // Récupération de l'ensemble des templates
3     temp, err := template.ParseGlob("./templates/*.html")
4     if err != nil {
5         fmt.Println(fmt.Sprint("ERREUR => %s", err.Error()))
6         return
7     }
8
9     type User struct {
10         FirstName string
11         LastName  string
12     }
13 // Déclaratrion d'une structure correspondant aux champs du template
14 type PageLocal struct {
15     Title string
16     Users []User
17 }
18
19 http.HandleFunc("/boucle/local", func(w http.ResponseWriter, r *http.Request) {
20     data := PageLocal{"Liste des mentors",
21         []User{{"Cyril", "RODRIGUES"},
22             {"Kheir-eddine", "MEDERREG"},
23             {"Alan", "PHILIPPIERT"}}}
24     // Appel du template nommé "exempleBoucleLocal" avec les données stockées dans la variable data
25     temp.ExecuteTemplate(w, "exempleBoucleLocal", data)
26 })
27
28 // Initialisation du serveur HTTP
29 http.ListenAndServe("localhost:8080", nil)
30 }
```

L'UTILISATION DE BOUCLES

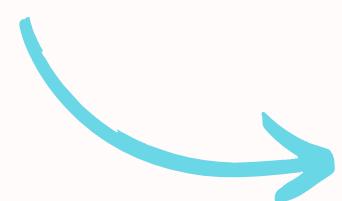
Partie template
(HTML)

```
1 {{define "exempleBouclePrimitif"}}
2 <!DOCTYPE html>
3 <html lang="fr">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Capsule WEB&GO</title>
8 </head>
9 <body>
10  <h1>{{.Title}}</h1>
11  <ul>
12    {{range .ListId}}
13      <p>{{.}}</p>
14    {{end}}
15  </ul>
16 </body>
17 </html>
18 {{end}}
```

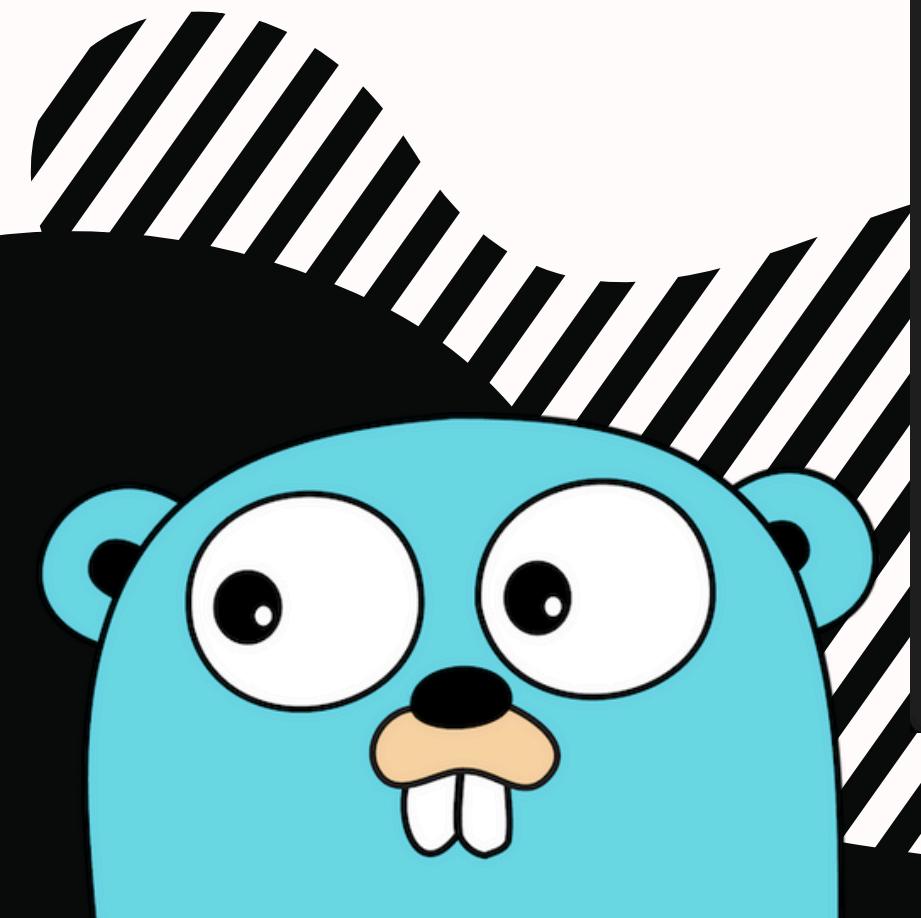
L'UTILISATION DE BOUCLES



Partie Golang



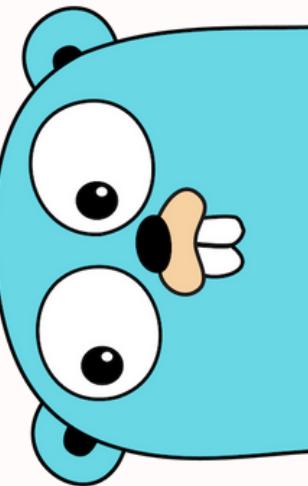
```
1 func main() {
2     // Récupération de l'ensemble des templates
3     temp, err := template.ParseGlob("./templates/*.html")
4     if err != nil {
5         fmt.Println(fmt.Sprint("ERREUR => %s", err.Error()))
6         return
7     }
8
9     // Déclaratrion d'une structure correspondant aux champs du template
10    type PagePrim struct {
11        Title string
12        ListId []int
13    }
14
15    http.HandleFunc("/boucle/primitif", func(w http.ResponseWriter, r *http.Request) {
16        data := PagePrim{"Liste des références des commandes", []int{10, 87, 65, 12, 47, 89, 5874, 247}}
17        // Appel du template nommé "exempleBouclePrimitif" avec les données stockées dans la variable data
18        temp.ExecuteTemplate(w, "exempleBouclePrimitif", data)
19    })
20
21    // Initialisation du serveur HTTP
22    http.ListenAndServe("localhost:8080", nil)
23 }
```



LES AUTRES ACTIONS

Il y a encore d'autres actions qui peuvent être très utiles voici une liste non exhaustive d'actions :

- L'action {{block}}
- L'action {{template}}
- L'action {{with}}



NO PRESSURE

PRATIQUE !

BUT IT'S YOUR TURN

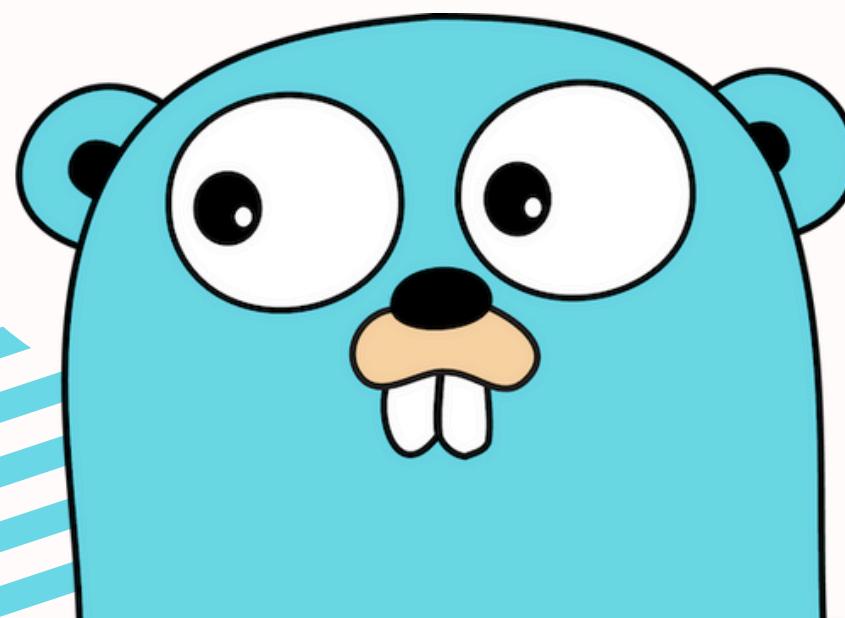
PARTIE III

Créez une route /condition qui affiche un template avec un titre (h1), un paragraphe, et une condition qui contrôle l'affichage d'un autre paragraphe en fonction d'une action.

Implémentez également une route /boucle qui affiche un template contenant un titre passé en variable, et une boucle pour afficher une liste de prénoms.

Enfin, créez une route /template/test qui affiche un template utilisant trois actions différentes parmi celles déjà citées (conditions, boucles, etc.)

Vous avez 15 minutes pour accomplir cette tâche.



Ps – Utilisez ParseGlob et ExecuteTemplate, en nommant correctement vos templates.