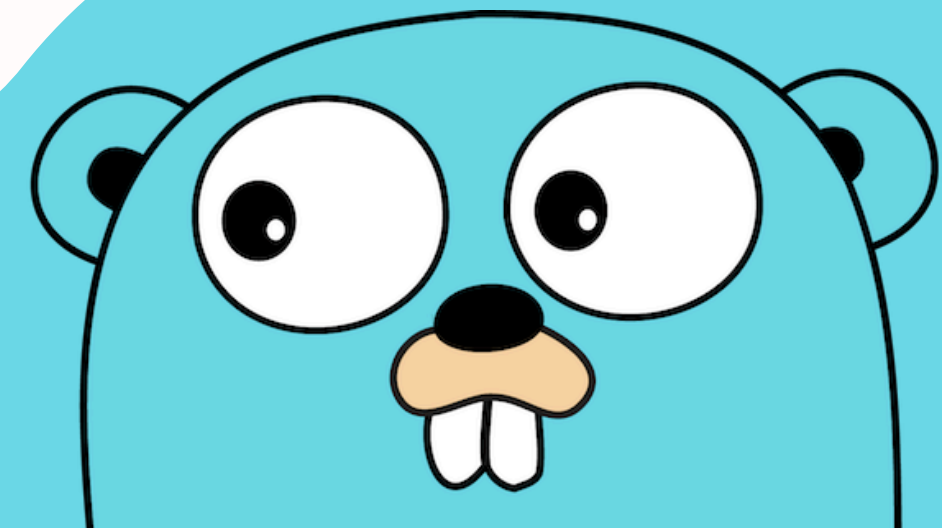
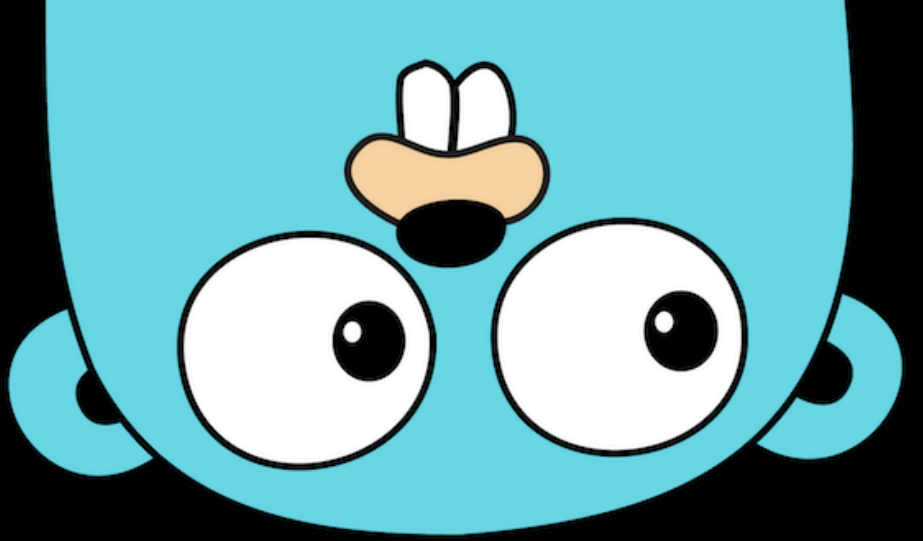


CAPSULE

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

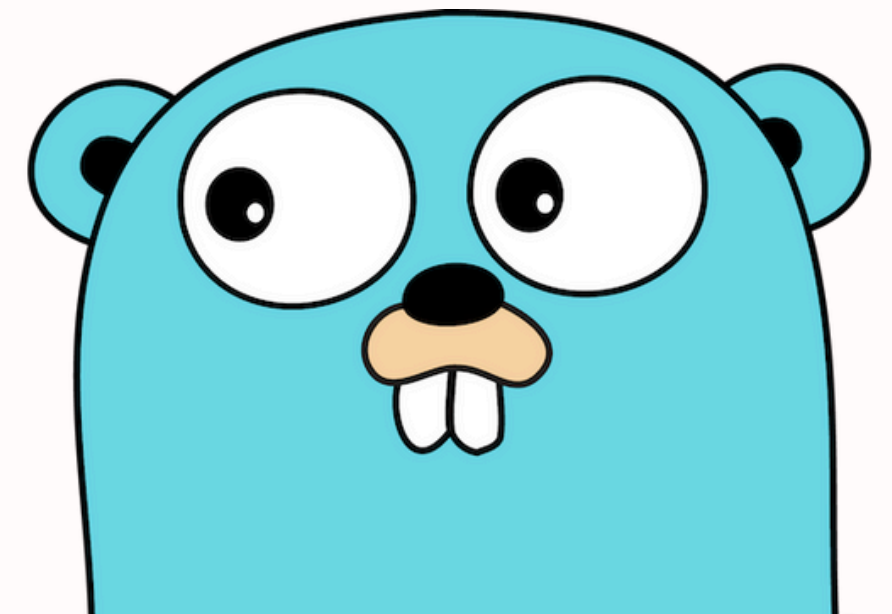


QU'EST CE QU'UN TEMPLATE ?

Un **Template**, ou modèle, est un gabarit préétabli constitué d'éléments modifiables qui sert de base pour créer des documents, des pages web, des emails ou d'autres types de contenus. Également appelé modèle de vue, il permet de standardiser la structure et le format d'une page web ou d'un document en utilisant des variables, des conditions ou des boucles qui rendent les pages dynamiques. En Go le package **"html/template"** fournit tous les éléments pour utiliser des Templates orientés web ci-dessous le lien vers la documentation :

<https://pkg.go.dev/html/template>

<https://pkg.go.dev/text/template#hdr-Actions>

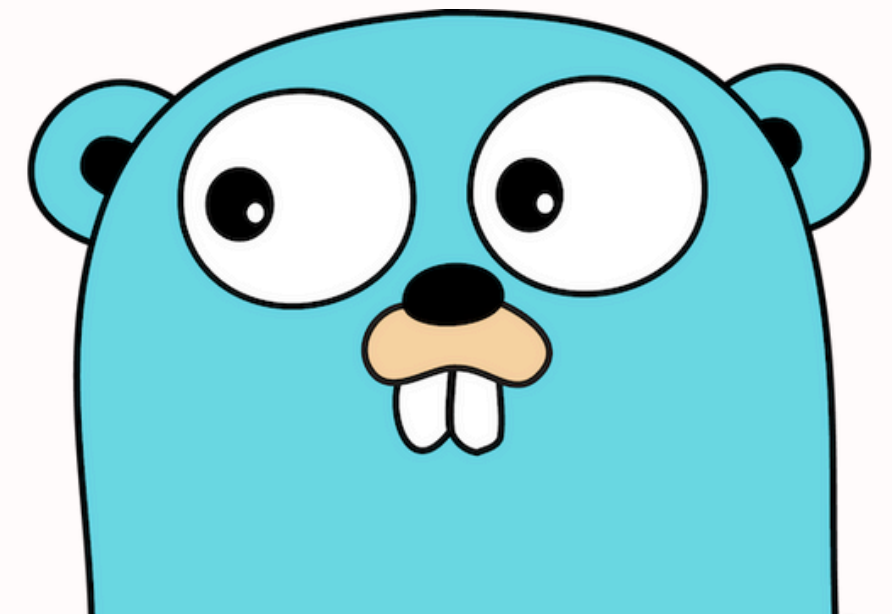


COMMENT UTILISE-T-ON UN TEMPLATE AVEC GO ?

Pour associer un modèle de vue, ou bien encore un Template à une route pour l'envoyer au client comme réponse il faudra agir en plusieurs étapes :

- Récupérer le Template (Récupérer les fichiers HTML)
- Préparer et exécuter le Template (Remplacer les actions, puis envoyer la vue en réponse)

Note – Dans notre cas les templates seront des fichiers HTML



LA MÉTHODE "PARSEGLOB"

La méthode **"ParseGlob"** permet de récupérer plusieurs templates situés dans un dossier en utilisant un pattern. Elle charge tous les templates en une seule fois. Cette méthode prend comme argument le chemin du dossier et renvoie un pointeur vers les Template (*template.Template) ainsi qu'une erreur (error) en cas de problèmes lors de la récupération (structure, syntaxe, etc.)

```
1 // Permet de récupérer l'ensemble des Templates correspondant au paterne indiqué en argument
2 listTemp, errTemp := template.ParseGlob("./templates/*.html")
3 // listeTemp, contient l'ensemble des modèles de vue de type *template.Template
4 // errTemp, contient l'erreur, égale à nil s'il n'y a pas de problème
```

Note – Un pattern est une chaîne qui correspond à des fichiers selon une structure définie, comme *.html pour sélectionner tous les fichiers HTML dans un dossier.

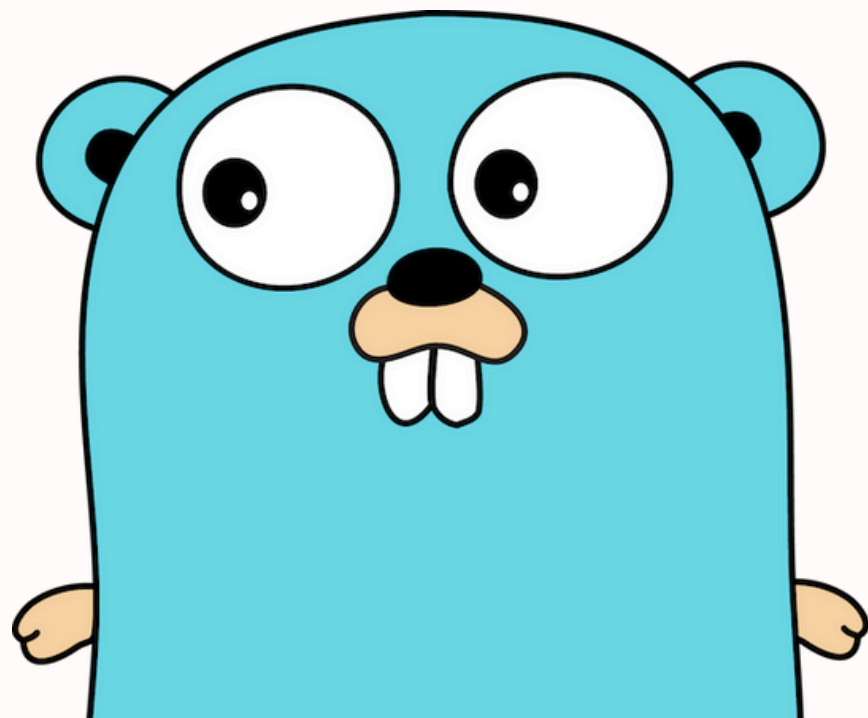
LA MÉTHODE "PARSEFILES"

La méthode **"ParseFiles"** permet de récupérer un ou plusieurs templates en spécifiant leurs chemins sous forme de chaînes de caractères passées en paramètres. Contrairement à ParseGlob, les fichiers sont rechargés à chaque appel, ce qui peut avoir un impact sur les performances dans certains contextes. Elle retourne une instance de type *template.Template qui contient tous les templates récupérés, ainsi qu'une erreur si un problème survient.

```
1 // Récupération d'un template au chemin suivant : ./templates/index.html
2 temp, tempErr := template.ParseFiles("./templates/index.html")
3 // temp contient le template de type *template.Template
4 // tempErr contient l'erreur, égale à nil s'il n'y a pas de problème
5
6 // Récupération de plusieurs templates aux chemins suivants :
7 // ./templates/index.html & ./templates/header.html
8 page, pageErr := template.ParseFiles("./templates/index.html", "./templates/header.html")
9 // page contient les templates de type *template.Template
10 // pageErr contient l'erreur, égale à nil s'il n'y a pas de problème
```

PRÉPARER ET EXÉCUTER UN TEMPLATE

Pour préparer un Template puis l'envoyer en tant que réponse au client, il est possible d'utiliser deux manières différentes qui va dépendre de la manière dont on n'a récupéré le ou bien les Templates. Il faudra placer cette appel de méthode directement dans le gestionnaire associé à la route.



EN CAS DE RÉCUPÉRATION D'UN SEULE TEMPLATE

Il est possible d'utiliser la méthode la méthode « **Exécute** » lié à une variable de type `template.Template` (variable qui contient les Templates), elle va permettre prend 2 arguments un `io.Writer` puis une structure de données correspondant aux variables utilisées dans le Template (qui peut être égale à `nil`).

```
1 http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
2     // Récupération du Template se situant au chemin suivant : ./templates/index.html
3     temp, tempErr := template.ParseFiles("./templates/index.html")
4     if tempErr != nil {
5         fmt.Fprintf(w, "Oups erreur avec le chargement du Template : %s", tempErr.Error())
6         return
7     }
8     // Exécution du Template stocké dans la variable : temp
9     temp.Execute(w, nil)
10 })
```

EN CAS DE RÉCUPÉRATION D'UNE MULTITUDE DE TEMPLATES

Il est possible d'utiliser la méthode « **ExecuteTemplate** » qui permet d'envoyer au client comme réponse une vue (Template) en identifiant le Template à exécuter par son nom. Cette méthode prend trois arguments : un `io.Writer`, le nom du Template, puis une structure de données correspondant aux variables utilisées dans le Template (qui peut être égale à `nil`).

```
1 http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
2     // Récupération du Template se situant aux chemins suivants : ./templates/index.html & ./templates/header.html
3     temp, tempErr := template.ParseFiles("./templates/index.html", "./templates/header.html")
4     if tempErr != nil {
5         fmt.Fprintf(w, "Oops erreur avec le chargement du Template : %s", tempErr.Error())
6         return
7     }
8     // Exécution d'un Template stocké dans la variable temp portant le nom : "index"
9     // Attention ce n'est pas le nom de fichier, mais celui indiqué dans le fichier HTML !
10    temp.ExecuteTemplate(w, "index", nil)
11 })
```

EN CAS DE RÉCUPÉRATION D'UNE MULTITUDE DE TEMPLATES

```
1 func main() {
2     // Récupération de l'ensemble des Templates correspondant au paterne
3     // paterne : ./templates/*.html
4     temp, tempErr := template.ParseGlob("./templates/*.html")
5     if tempErr != nil {
6         fmt.Printf("Oops erreur avec le chargement du Template : %s", tempErr.Error())
7         os.Exit(02)
8     }
9
10    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
11        // Instruction à exécuter lors de l'appel de la route "/"
12        // Exécution du Template chargé au lancement stocké dans la variable temp nommé "accueil"
13        temp.ExecuteTemplate(w, "accueil", nil)
14    })
15
16    http.ListenAndServe("localhost:8989", nil)
17 }
```

A meme featuring Woody and Buzz Lightyear from the movie Toy Story. Woody is on the left, looking concerned. Buzz is on the right, holding a green blaster and looking surprised. The background is a dimly lit room.

NO PRESSURE

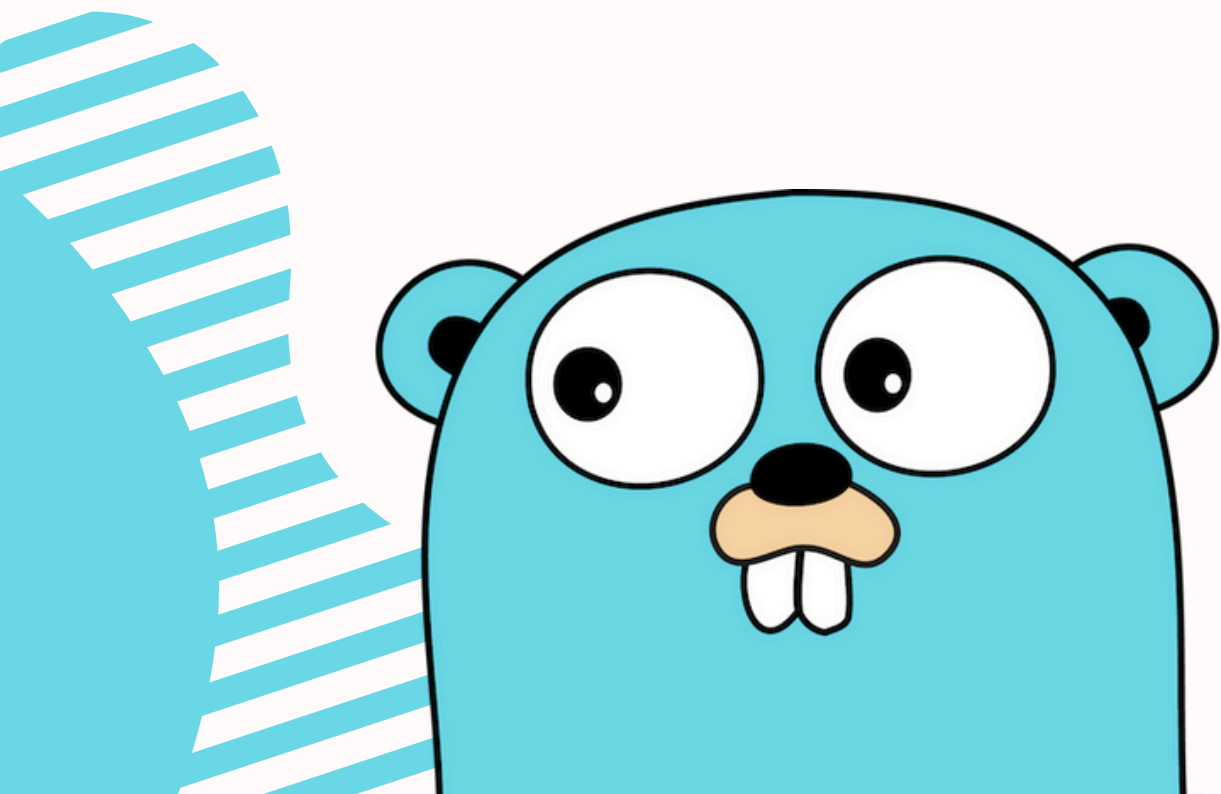
PRATIQUE !

BUT IT'S YOUR TURN

PARTIE II

Créez une route /template qui affiche un template contenant un titre (h1) et un paragraphe. Vous devez créer un fichier template pour cela, et utiliser la méthode ParseFile avec Execute afin de récupérer et afficher ce template.

Vous avez 5 minutes pour accomplir cette tâche.



PS – Nous verrons comment nommer un template dans la partie suivante