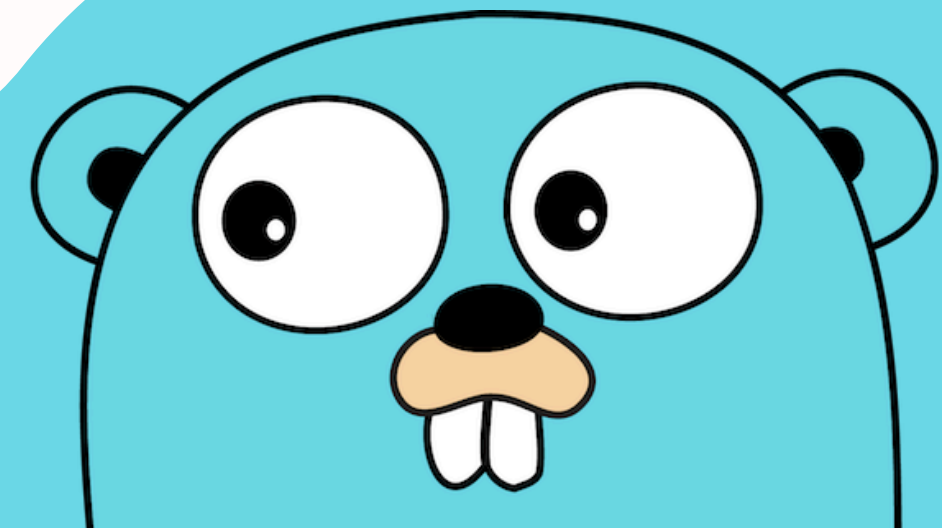
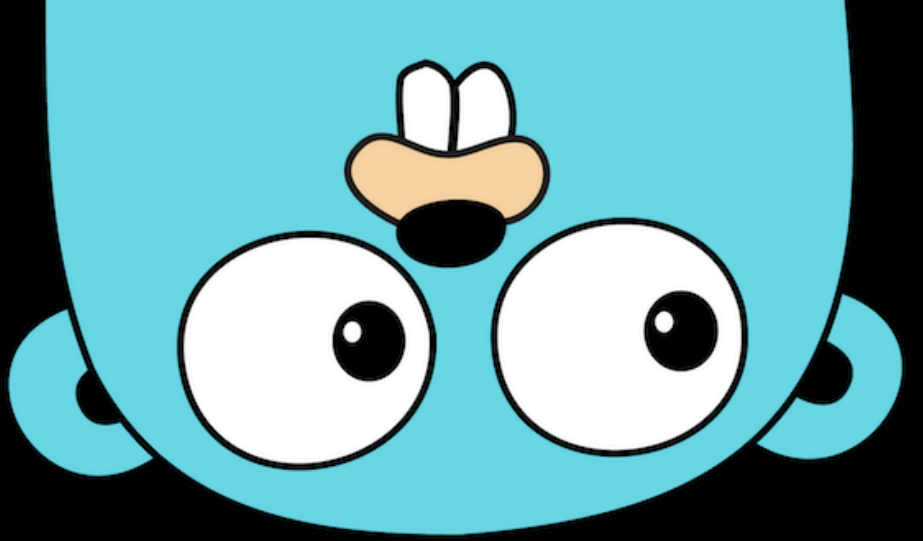


CAPSULE

LA PROGRAMMATION AVEC GOLANG ORIENTÉ WEB

Par Cyril RODRIGUES

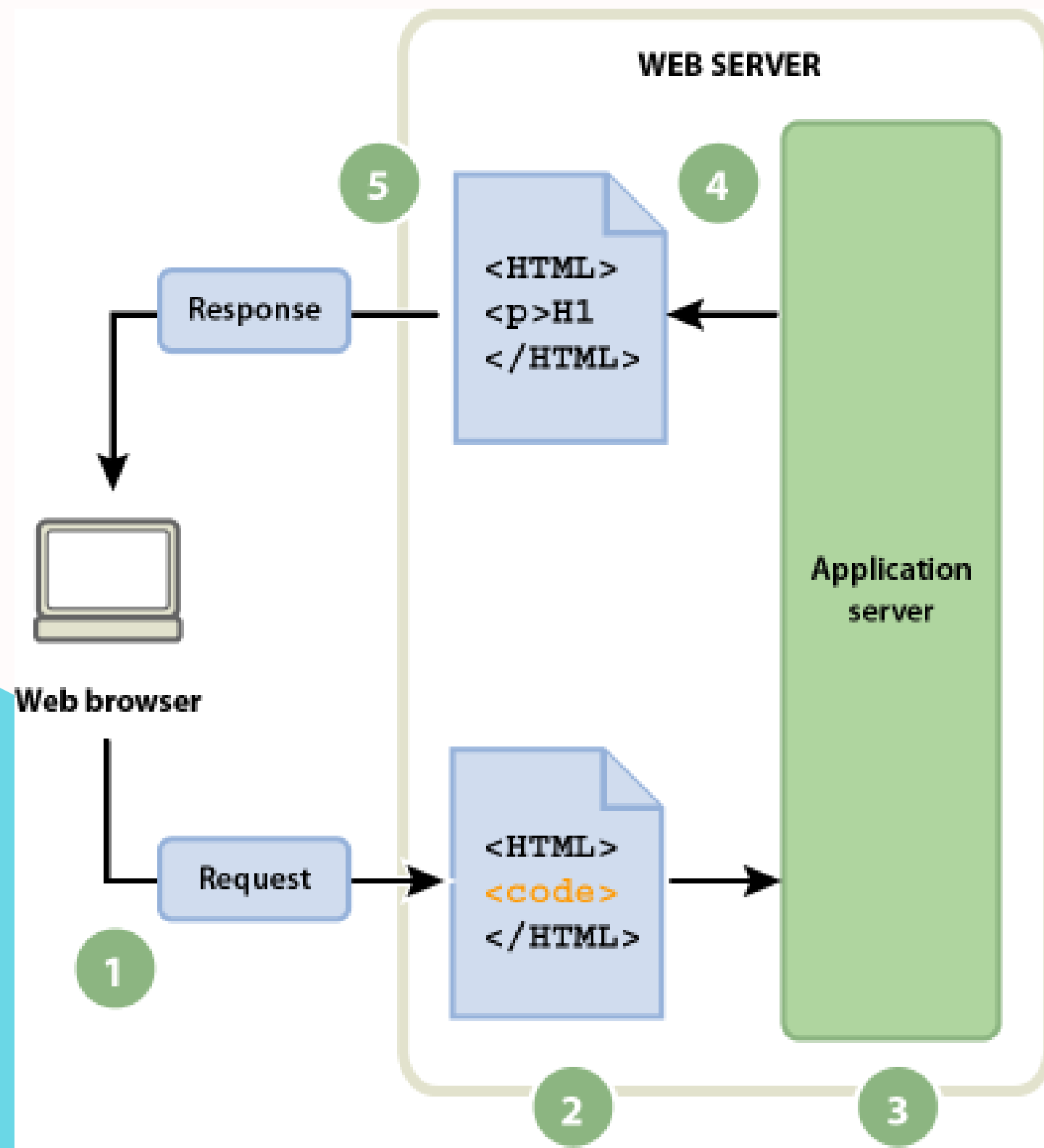


- 
- 01 COMMENT FONCTIONNE UN SITE WEB ?
 - 02 IMPLÉMENTATION D'UN SERVEUR HTTP
 - 03 LES TEMPLATES
 - 04 LES TEMPLATES & LES ACTIONS
 - 05 UTILISATION DE RESSOURCES STATIQUES
 - 06 TRAITEMENT ET UTILISATION DE FORMULAIRES
 - 07 DES SUGESTIONS POUR APPROFONDIR



COMMENT FONCTIONNE UN SITE WEB ?

FONCTIONNEMENT D'UN SITE WEB



Le **protocole HTTP** (HyperText Transfer Protocol) permet l'échange de ressources comme des pages HTML, des images, ou des données. Il repose sur **un modèle de communication client-serveur** : le client (généralement un navigateur) envoie une requête, et le serveur renvoie une réponse contenant les ressources demandées. **Ce protocole est la pierre angulaire du web car il permet la navigation entre les pages et la consulter le contenu.**

L'évolution de ce protocole vers HTTPS (HTTP Secure) ajoute une couche de sécurité en chiffrant les échanges via SSL/TLS.



IMPLÉMENTATION D'UN SERVEUR HTTP

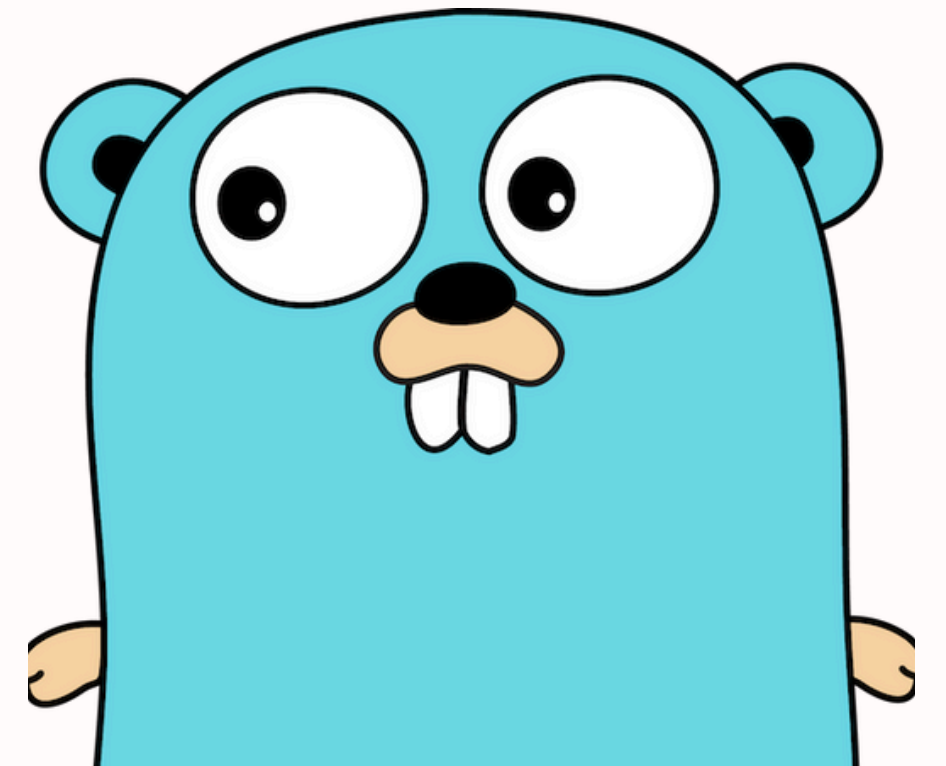


QU'EST-CE QU'UN SERVEUR HTTP ?

Un **serveur HTTP** est un programme qui reçoit et traite des requêtes HTTP provenant de clients, comme les navigateurs web. Il envoie des réponses contenant des ressources demandées (pages HTML, fichiers, etc.). En Go, le package `net/http` fournit tous les outils nécessaires pour créer facilement un serveur HTTP. Ce package permet de définir des routes, de traiter des requêtes et de renvoyer des réponses. Par exemple, un serveur simple en Go peut être mis en place en quelques lignes de code.

Liens vers les documentations

- <https://pkg.go.dev/net/http> (documentation officielle)
- <https://gowebeexamples.com/http-server/>

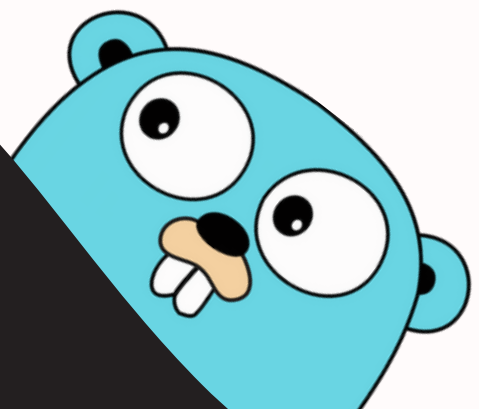


IMPLÉMENTATION D'UN SERVEUR HTTP

```
1 // Initialisation d'un serveur sur le port : 8080  
2 http.ListenAndServe("localhost:8080", nil)
```

La méthode **"ListenAndServe"** permet d'initialiser un serveur HTTP qui écoute sur un port spécifique et gère les requêtes entrantes. Elle prend deux arguments : l'adresse du serveur (souvent une combinaison d'IP et de port) et un gestionnaire généralement "nil".

Dans cet exemple, le serveur est accessible via l'adresse `http://localhost:8080` dans le navigateur.



IMPLÉMENTATION D'UNE ROUTE AU SERVEUR



```
1 http.HandleFunc("/route", func(w http.ResponseWriter, r *http.Request) {  
2 // Ici les instructions exécutées lors de l'appel de cette route  
3 // C'est la partie logique associé à la route  
4 w.Write([]byte("Réponse..."))  
5 })
```

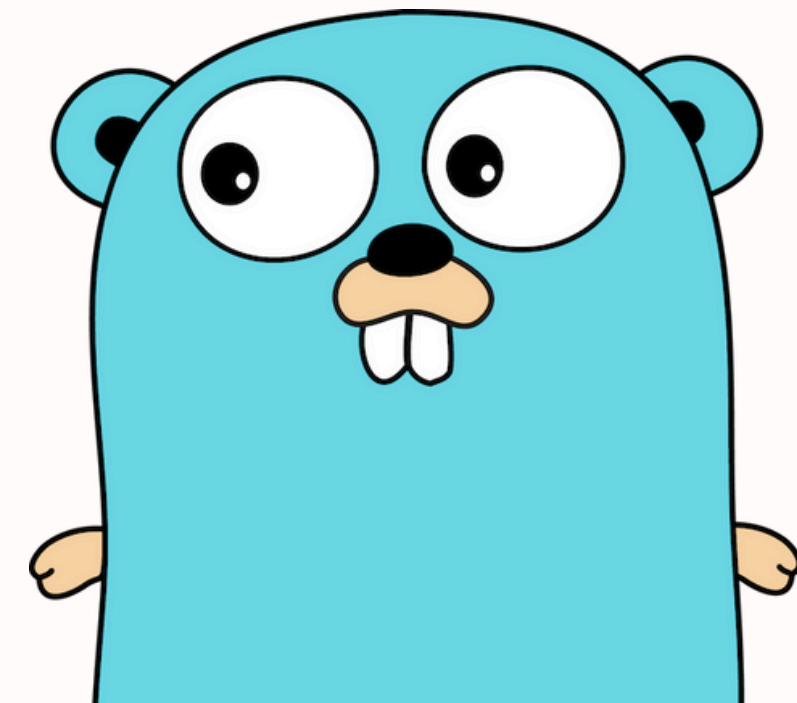
La méthode **"HandleFunc"** permet de créer une route avec son gestionnaire associé. Elle prend deux arguments : une chaîne de caractères représentant la route et une fonction de type Handler. Cette fonction gestionnaire possède deux paramètres : r, qui représente la requête HTTP envoyée par le client, et w, qui est utilisé pour envoyer une réponse.

Note : Les routes doivent être uniques pour éviter les conflits, et elles sont sensibles à la casse (case-sensitive), ce qui signifie que **"/Accueil"** et **"/accueil"** seront considérées comme deux routes différentes.

QUEL EST LE RÔLE D'UN GESTIONNAIRE (HANDLER) ?

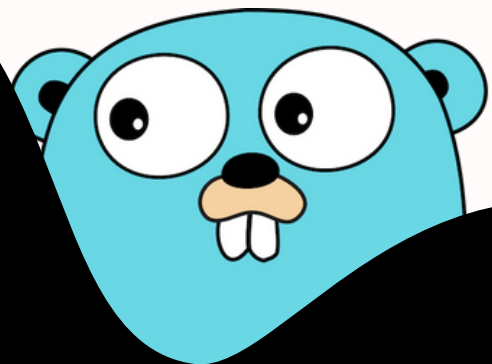
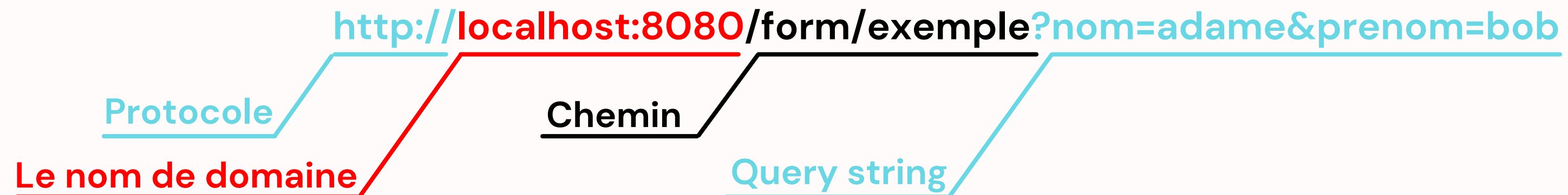
Un gestionnaire (Handler) est essentiel dans un serveur HTTP en Go car **il définit comment traiter une requête pour une route spécifique**. C'est dans cette fonction que qu'il faut définir la logique nécessaire pour répondre à la requête : renvoyer une page HTML, traiter des données soumises par un formulaire, ou bien rediriger l'utilisateur. Le gestionnaire agit comme le cœur du traitement des requêtes et joue un rôle central dans le fonctionnement du serveur.

Note – un gestionnaire doit obligatoirement envoyer une réponse à la requête du client



QU'EST-CE QU'UNE ROUTE ?

Une route est une URL utilisée pour accéder à une ressource spécifique sur un serveur web. Un serveur peut posséder de nombreuses routes, chacune permettant de servir des ressources différentes. Une URL se compose de plusieurs parties. Voici un exemple d'une URL et ses éléments :



EXEMPLE

```
1 func main() {  
2     // Implémentation d'une route : /cours  
3     http.HandleFunc("/cours", func(w http.ResponseWriter, r *http.Request) {  
4         // Réponse envoyé au client : Bonjour bienvenue sur la route /cours  
5         w.Write([]byte("Bonjour bienvenue sur la route /cours"))  
6     })  
7  
8     // Initialisation du serveur HTTP  
9     http.ListenAndServe("localhost:8080", nil)  
10 }
```

Dans cet exemple, nous avons implémenté un serveur HTTP qui écoute sur le port 8080 de l'adresse localhost. Nous avons ensuite mis en place une route /cours qui envoie la réponse "Bonjour..." au client. Pour tester, lancez le programme puis ouvrez un navigateur et accédez à l'adresse suivante : <http://localhost:8080/cours>.

A meme featuring Woody and Buzz Lightyear from the movie Toy Story. Woody is on the left, looking concerned with a worried expression. Buzz is on the right, wearing his green and white space suit, with his arms outstretched in a dramatic gesture. The background is a dimly lit room with a window showing a night sky with stars.

NO PRESSURE

PRATIQUE !

BUT IT'S YOUR TURN

PARTIE I

Vous devez implémenter un serveur HTTP qui écoute sur l'adresse locale (localhost ou bien 127.0.0.1) de votre machine sur le port 8000.

Puis pour terminer vous devez implémenter une route `"/index"` qui va afficher le texte de votre choix.

Vous disposez de 5 min pour effectuer cette tâche.

