Protocoles Internet
TP 1 : Introduction au Go

Juliusz Chroboczek

2 octobre 2023

Exercice 1.

1. Créez un répertoire « `~/go/src/hello/` ». Dans ce répertoire, tapez « `go mod init hello` ». Quels fichiers ont été créés ? Examinez-en le contenu.
2. Dans *Emacs*, créez un fichier nommé `hello.go`. Vérifiez que le tampon est en mode *Go*. Tapez ce qui suit :

```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Hello, world!")
}
```

3. Reformatez votre code à l'aide de « `M-x gofmt RET` » depuis Emacs puis sauvegardez, ou, si vous préférez travailler depuis la ligne de commande, « `gofmwt -w hello.go` ».
4. Compilez et exécutez à l'aide de la commande « `go run hello.go` » (un fichier binaire temporaire est créé puis supprimé immédiatement).
5. Créez un binaire à l'aide de la commande « `go build` », puis exécutez à l'aide de « `./hello` ».
6. Examinez la documentation de la fonction `fmt.Println` sur `https://pkg.go.dev`. Examinez aussi `fmt.Print` et `fmt.Printf`.

Exercice 2. Écrivez un programme Go qui affiche la liste des nombres premiers inférieurs à 1000 en utilisant l'algorithme du crible d'Eratosthène. Vous pourrez créer un tableau de taille n (un « *slice* ») à l'aide de la syntaxe suivante :

```
a := make([]int, n)
```

Vous n'omettrez pas de formater votre code à l'aide de « `gofmt` » et de le vérifier à l'aide de « `go vet` ».

Exercice 3. Téléchargez le code fourni sur

`https://www.irif.fr/~jch/enseignement/internet/tp1.tar.gz`

Désarchivez-le et exécutez-le.

1. À l'aide d'un navigateur, connectez-vous à `http://localhost:8080/hello.text` et à `http://localhost:8080/hello.html`. Examinez les mêmes URL à l'aide de la commande « `curl -i` ».
2. Modifiez le programme pour que les salutations affichées dans les deux URL de la question précédente le soient dans votre langue maternelle.

Exercice 4.

1. Examinez la page `http://localhost:8080/name-get`. Que fait le bouton ? Pourquoi n'a-t-elle pas d'extension ?
2. Créez un gestionnaire à l'URL `http://localhost:8080/request-name` qui :
   - vérifie que la méthode est soit `HEAD` soit `GET` et retourne une erreur *Method not allowed* si ce n'est pas le cas ;
   - appelle la fonction `r.ParseForm()`
   - retourne une page HTML qui contient le texte « Votre nom est » suivi du nom passé en paramètre de l'URL, que vous pouvez obtenir à l'aide de `r.Form.Get`.

   Testez votre gestionnaire d'abord à l'aide de la page *web* fournie, ensuite à l'aide de `curl -i`.
3. Examinez maintenant la page `http://localhost:8080/name-post.html`, puis créez un gestionnaire à l'URL `http://localhost:8080/request-name-post`
   - vérifie que la méthode est `POST` et retourne une erreur *Method not allowed* si ce n'est pas le cas ;
   - affiche le type du corps de la requête que vous pourrez obtenir à l'aide de `r.Header.Get("Content-Type")` ;
   - affiche le corps de la requête à l'aide de `io.Copy(os.Stdout, r.Body)`.

   Examinez le corps de la requête générée par le formulaire.
4. Modifiez votre gestionnaire pour qu'il affiche une page qui contient le texte « Votre nom est » suivi du nom passé dans le corps du message. Vous pourrez utiliser `r.ParseForm()` et `r.Form.Get` comme ci-dessus.

Exercice 5. Écrivez un serveur web qui affiche un formulaire qui demande à l'utilisateur d'entrer un entier n, puis affiche une page web qui contient la liste des nombres premiers compris entre 2 et n. Vous pourrez vous servir de la fonction `strconv.ParseInt` pour analyser le paramètre passé à votre gestionnaire.

Quelle méthode est plus adaptée : `GET` ou `POST` ?

Exercice 6.

Modifiez votre serveur pour qu'il serve du HTTPS au lieu de HTTP et qu'il utilise le port 8443 au lieu de 8080. Vour pourrez vous servir du paquet `github.com/jech/cert`.