LES PACKAGES DANS LE LANGAGE DE PROGRAMMATION GO

Définition

Jusqu'ici nous n'avons écrit notre code que dans un seul fichier, pour l'instant c'était tolérable car la taille de note code était petite, mais plus votre code va grandir plus vous allez vite vous rendre compte qu'il est nécessaire de déplacer des portions de votre code dans d'autres fichiers afin de mieux organiser et maintenir votre programme, et c'est là qu'interviennent les packages !

Un package n'est rien d'autre qu'un répertoire contenant des fichiers de code go, qui peut être exécuté par d'autres fichiers go.

Comme nous en avons discuté au début de nos chapitres, il existe deux types de packages :

- **Package exécutable**
- **Package utilitaire**

Dans un package exécutable on retrouve le fameux **package main** qui permet d'informer votre compilateur Go que le paquet doit être compilé en tant que programme exécutable au lieu d'un package utilitaire.

Un package utilitaire n'est pas auto-exécutable, il améliore plutôt les fonctionnalités d'un package exécutable en lui fournissant des fonctionnalités.

Création de packages

Où créer mon package ?

Pour importer un paquet, il faut utiliser le mot-clé `import` suivi du **nom du paquet**.

Lorsque vous créez un paquet et que vous l'importez sur votre package principal, votre compilateur va d'abord chercher le répertoire du paquet dans le `$GOROOT/src/` et si répertoire n'existe pas il va le cherchera dans le dossier `$GOPATH/src/`.

> **Information**
>
> `$GOROOT` et `$GOPATH` sont deux variables d'environnement créez lors de l'installation de GO.

Il est possible d'afficher toutes vos variables d'environnement Go comme suit :

```
go env
```

Résultat :

```
set GOCACHE=C:\Users\hatim\AppData\Local\go-build
...
set GOPATH=C:\Users\hatim\go
set GOROOT=C:\Go
...
```

Je vais volontairement importer un package qui n'existe pas, pour vous prouver qu'il cherche bel et bien vos packages dans ces deux chemins :

```
package main
import (
    "aucunpackage"
    "fmt"
)
```

```
func main() {
    fmt.Println(aucunpackage.affiche())
}
```

Erreur :

```
cannot find package "aucunpackage" in any of:
C:\Go\src\aucunpackage (from $GOROOT)
C:\Users\hatim\go\src\aucunpackage (from $GOPATH)
```

Le compilateur nous avertis qu'il ne retrouve pas le package dans mon dossier **`$GOROOT`** et **`$GOPATH`**. Il est donc nécessaire de déposer vos packages dans un des deux chemins.

Arborescence

Dans cet nouvel exemple, nous allons créer un simple package qui nous permet d'afficher un nom et un sexe, mais avant de toucher à du code on va d'abord réfléchir à notre arborescence

Déjà premièrement on va déposer nos packages comme go nous le demande à savoir dans le dossier **`$GOPATH/src/`**, qui est l'équivalent sur mon ordinateur au chemin **`C:\Users\hatim\go\src\`** (tapez la commande `go env` pour connaître le chemin de **`$GOPATH/src/`**).

Ensuite pour créer un package il faut créer un dossier avec le nom de notre package suivi d'un fichier (même nom que le package de préférence) ou on déposera le code Go de notre package.

Dans l'exemple qui suit notre package se nommera **affichage** et notre package exécutable se nommera **main.go**. Ce qui nous donnera l'arborescence suivante :

```
$GOPATH/src
|___ affichage
```

```
| |___ affichage.go
main.go
```

Création du package

Voici à quoi va ressembler notre fichier **affichage.go**

```
package affichage

var Nom string = "Hatim"
var sexe string = "masculin"

func AfficheSexe() string {
    return Nom + " est de sexe " + sexe
}
```

> **Attention**
>
> Go exporte une variable dans un autre package **si et seulement si le nom de la variable commence par une majuscule**. Toutes les autres variables ne commençant pas par une lettre majuscule sont privées du paquet.

La ligne de code `package affichage` indique à notre compilateur qu'il s'agit bien d'un package utilitaire nommé "affichage". Ensuite pour le reste du code je déclare deux variables et une fonction qui peuvent être réutilisées depuis le fichier main.go une fois qu'il aura importé le package.

Il ne reste plus maintenant qu'à importer le package affichage dans le fichier main.go et par la suite il sera possible depuis le fichier main.go d'utiliser les fonctions et les variables du package affichage.

Voici à quoi va ressembler main.go :

```
package main

import (
    "affichage"
    "fmt"
)

func main() {
    fmt.Println("Je m'appelle", affichage.Nom)
    fmt.Println(affichage.AfficheSexe())
}
```

Résultat :

```
Je m'appelle Hatim
Hatim est de sexe masculin.
```

Imbrication de paquets

Nous pouvons imbriquer un paquet dans un paquet. Tout ce que nous avons à faire c'est de fournir le chemin relatif du paquet imbriqué.

Nous allons rajouter dans notre exemple précèdent un sous-package nommé **passion** avec une fonction qui retourne un string. Notre nouvelle arborescence ressemblera à ceci :

```
$GOPATH/src
|___ affichage
| |___ affichage.go
| |___ passion
|   |___ passion.go
main.go
```

Voici à quoi va ressembler notre fichier passion.go :

```
package passion

func AffichagePassion() string {
```

```
    return "j'aime programmé en Go !"
}
```

Et voici à quoi va ressembler notre fichier main.go :

```
package main

import (
    "affichage"
    "affichage/passion" // chemin relatif
    "fmt"
)

func main() {
    fmt.Println("Je m'appelle", affichage.Nom)
    fmt.Println(affichage.AfficheSexe())
    fmt.Println(passion.AffichagePassion())
}
```

ici on utilise le chemin relatif pour importer notre package passion imbriqué dans notre package affichage.

Résultat :

```
Je m'appelle Hatim
Hatim est de sexe masculin
j'aime programmé en Go !.
```