LA PORTÉE DES VARIABLES DANS LE LANGAGE DE PROGRAMMATION GO

Présentation

Quand on parle de portée des variables on parle d'**endroits** dans notre code où on peut utiliser telle ou telle variable.

On peut résumer la portée des variables par la question suivante :

"Est-ce que ma variable est accessible dans tel ou tel bloc de mon code ?"

On peut différencier la portée de nos variables par des :

- **Variables locales**
- **Variables globales**
- **Paramètres formels**

Variables locales

Les variables déclarées à l'intérieur d'une fonction ou d'un bloc (un bloc est tout simplement la partie de votre code dans des accolades) sont appelées variables locales. Elles ne peuvent être **accessibles** qu'à l'intérieur de votre fonction ou d'un bloc de code.

Exemple d'une variable accessible que depuis un bloc :

```
package main

import (
    "fmt"
```

```
)

func test() {
    for i := 1; i < 3; i++ {
        a := 20
        a *= i
        fmt.Println(a)
    }
}

func main() {
    test()
}
```

Résultat :

```
20
40
```

Jusqu'ici rien de choquant par contre si on tente de manipuler la variable `a` en dehors de notre bloc ...

```
package main

import (
    "fmt"
)

func test() {
    for i := 1; i < 3; i++ {
        a := 20
        a *= i
        fmt.Println(a)
    }
    fmt.Println(a) // erreur ici
}

func main() {
    test()
}
```

Erreur :

```
undefined: a
```

Si on souhaite exploiter notre variable à la fois dans notre fonction et à la fois dans notre bloc alors il suffit de la déclarer au début de notre fonction, comme ceci

```
package main

import (
    "fmt"
)

func test() {
    a := 20 // déclaration de notre variable locale
    for i := 1; i < 3; i++ {
        a *= i
        fmt.Println("dans ma boucle for :", a)
    }
    fmt.Println("en dehors de ma boucle for :", a)
}

func main() {
    test()
}
```

Résultat :

```
dans ma boucle for : 20
dans ma boucle for : 40
en dehors de ma boucle for : 40
```

On va maintenant tenter d’utiliser une variable en dehors d'une fonction

```
package main

import (
    "fmt"
)

func test() {
    a := 10
    a += 20
    fmt.Println(a)
}
func main() {
    test()
    fmt.Println(a) // l'erreur vient d'ici
}
```

Erreur :

```
undefined: a
```

Hum ça ne fonctionne pas, il serait peut-être temps de voir comment ça se passe au niveau des variables globales.

Variables globales

Les variables globales sont définies en dehors de vos fonctions (généralement au début de votre programme). À l'inverse des variables locales elles **conservent** leur valeur pendant toute la durée de vie du programme et sont accessibles à l'intérieur de n'importe quelles fonctions définies dans votre programme.

Ça tombe bien car ça va résoudre notre problème !

```
package main

import (
    "fmt"
)

var g int // déclaration de notre variable globale

func test() {
    g += 20
    fmt.Println("Pendant ma fonction test() : ", g)
}
func main() {
    fmt.Println("Avant l'utilisation de la fonction test() :", g)
    test()
    fmt.Println("Pendant ma fonction main() : ", g)
    g += 30
    fmt.Println("Modifie moi encore : ", g)
}
```

Résultat :

```
Avant l'utilisation de la fonction test() : 0
```

```
Pendant ma fonction test() :  20
Pendant ma fonction main() :  20
Modifie moi encore :  50
```

> **Avertissement**
>
> Comme vous vous pouvez le constater la modification d'une variable globale est **permanente** quel que soit l'endroit où elle est modifiée.

Paramètres formels

Les paramètres formels sont traités comme des variables locales dans une fonction par contre ils auront toujours une **priorité** sur les variables globales.

```
package main

import (
    "fmt"
)

var g int // déclaration de notre variable formel

func test(g int) { // déclaration de notre paramètre globale
    g += 20 // prend le dessus sur notre variable globale
    fmt.Println("Pendant ma fonction test() : ", g)
}
func main() {
    fmt.Println("Avant l'utilisation de la fonction test() :", g)
    test(20)
    fmt.Println("Pendant ma fonction main() : ", g)
    g += 30
    fmt.Println("Modifie moi encore : ", g)
}
```

Résultat :

```
Avant l'utilisation de la fonction test() : 0
Pendant ma fonction test() :  40
Pendant ma fonction main() :  0
Modifie moi encore :  30
```