Le langage GO

Site du lange GO EN

Le langage GO a pris naissance en 2007 au sein de Google, d'une façon originale. Le géant californien faisait face à des défis de développement de taille, mais rencontrait un problème : aucun des langages de programmation existants ne convenait réellement aux problématiques maison en termes de performances, de productivité et facilité de maintenance.

- Ainsi, C++ produisait du code efficace, mais son usage était complexe et le temps de compilation trop long.
- Le code Java était aisément portable mais lent à la compilation et là encore trop complexe pour un usage courant.
- Python était aisé à apprendre et à utiliser, mais peu rapide à l'exécution.

Les principes du Go

Simplicité

Go est un langage aisé à aborder. Sa syntaxe concise et claire facilite l'écriture et la maintenance du code. Ses concepteurs ont veillé à limiter le nombre de mots-clés et à définir des règles de formatage uniformes. La documentation officielle est de qualité, ce qui facilite la découverte du langage. Les développeurs peuvent ainsi se concentrer sur la résolution de problèmes plutôt que sur les arcanes du langage Go.

Une exécution rapide

La compilation est quasi instantanée et de plus, le compilateur Go produit un exécutable rapide, approprié à des applications de haute performance. Autre atout, la compilation Go génère un seul fichier qui intègre l'ensemble de l'application et peut être exploité tel quel. Les déploiements sont donc simplifiés.

Multitaches

La prise en charge native du multitâche aide à élaborer des programmes impliquant l'exécution de plusieurs tâches en parallèle sur des processeurs multi cœurs.

Gestion automatique de la mémoire

Go gère un « garbage collector » (vidage automatique des éléments qui ne sont plus nécessaires) qui simplifie l'optimisation de la mémoire. Portabilité

Un programme écrit en Go peut être compilé et exécuté sur divers environnements sans modification du code source.

Un écosystème Open Source

Le caractère open source de Go a engendré l'apparition d'une multitude de bibliothèques et de frameworks (composants logiciels réutilisables) tels que Gin, Echo et Revel pour le développement web, et gRPC, une technologie qui permet à différentes applications de communiquer entre elles. En matière de data science, sont apparus des outils comme Gonum pour le calcul numérique ou Gorgonia pour le deep learning.

Quels usages pour le langage Go ?

Développement web backend

De plus en plus d'entreprises, comme Uber et SoundCloud, s'appuient sur Go pour construire leurs systèmes backend. Il se trouve que Go excelle dans la construction de microservices, des architectures de plus en plus populaires pour leur flexibilité et leur évolutivité.

Infrastructure

Go est omniprésent dans le paysage DevOps. Des outils comme Terraform, Prometheus et Docker Swarm, tous écrits en Go, sont devenus des références dans leurs domaines respectifs.

Data science

Bien qu'il soit moins connu que Python dans ce domaine, Go a gagné du terrain grâce à des bibliothèques comme Gonum et Gorgonia.

Conteneurs

Le langage Go sert couramment au développement de conteneurs Docker, une technologie qui permet d'encapsuler une application avec tout ce qu'elle contient (code, bibliothèques, dépendances, etc.) dans un conteneur, notamment avec l'outil Kubernetes conçu pour gérer et automatiser le déploiement de conteneurs.

Installation de Go : Étapes simples et rapides

- 1-Téléchargez l'installateur ou l'archive Rendez-vous sur la page officielle de téléchargement de Go ↗ et choisissez la version correspondant à votre système d'exploitation.
- 2-Installez Go sur votre système
  - Windows : Téléchargez le fichier **.msi**, exécutez-le et suivez les étapes de l'assistant d'installation. L'installateur configure automatiquement le chemin d'accès C:\Go\bin.
  - Linux/macOS : Téléchargez l'archive **.tar.gz**, puis exécutez les commandes suivantes dans le terminal :

ex1.txt

```
sudo tar -C /usr/local -xzf go1.xx.x.linux-amd64.tar.gz
```

(Remplacez **go1.xx.x.linux-amd64.tar.gz** par le nom du fichier téléchargé.)

- 3-Les variables d'environnement importantes

Pour que Go fonctionne correctement, il est essentiel de configurer certaines variables d'environnement. Voici les plus importantes :

- GOPATH : Indique le répertoire où sont stockés vos projets et dépendances. Par défaut, il est défini dans votre répertoire utilisateur ($HOME/go sur Linux/macOS ou %USERPROFILE%\go sur Windows). Si vous souhaitez personnaliser ce chemin, ajoutez dans votre fichier de configuration shell (~/.bashrc, ~/.zshrc, ou autre)

ex2.txt

```
export GOPATH=$HOME/my-go-projects
```

- GOROOT : Indique où Go est installé. En général, vous n'avez pas besoin de modifier cette variable, car elle est automatiquement définie après l'installation. Par exemple, sur Linux/macOS, elle est définie à /usr/local/go.

- PATH : Ajoutez le chemin du dossier contenant les binaires de Go pour pouvoir utiliser les commandes comme go build ou go run. Par exemple :

ex3.txt

```
    export PATH=$PATH:/usr/local/go/bin:$GOPATH/bin
```

Votre première application Go

Maintenant que vous avez installé Go, vous pouvez créer votre première application Go.

Pour créer un projet Go, vous devez créer un répertoire dans lequel vous stockerez vos fichiers de code. Par exemple, si vous souhaitez créer un projet nommé my-first-go-project, vous pouvez créer un répertoire nommé my-first-go-project dans votre répertoire utilisateur. Créer un fichier Go

Vous pouvez créer un fichier Go en utilisant n'importe quel éditeur de texte. Par exemple, si vous utilisez Visual Studio Code, vous pouvez créer un fichier nommé main.go dans le répertoire my-first-go-project.

Dans votre fichier **main.go**, ajoutez le code suivant :

ex4.go

```
package main

import "fmt"

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

Tout commence par une déclaration de package. En Go, chaque fichier appartient à un package, une sorte de regroupement logique du code. Si votre programme doit s'exécuter (et non juste fournir une bibliothèque), il doit obligatoirement appartenir au package main. C'est ici que tout démarre.

Ajouter des fonctionnalités avec les imports

Vous avez le squelette (main), mais il faut lui donner du muscle. C'est là qu'interviennent les packages. Go possède une bibliothèque standard robuste qui vous évite de réinventer la roue. Par exemple, si vous voulez afficher un message dans le terminal, vous importez le package fmt :

ex5.go

```
import "fmt"
```

Imaginez un instant que vous deviez écrire vous-même un module d'affichage. En Go, pas besoin : tout est prêt à l'emploi.

Et si votre programme a besoin de plusieurs fonctionnalités ? Pas de souci, regroupez vos imports dans un bloc :

ex6.go

```
import (
    "fmt"
    "os"
)
```

Cela rend le tout plus propre et Go adore la propreté. Après tout, l'ordre des choses, c'est la base, non ?

La fonction main est le point d'entrée de tout programme Go. Imaginez-la comme la porte d'entrée d'une maison : sans elle, impossible de visiter l'intérieur. Voici un exemple simple :

ex7.go

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

Ce bout de code fait une seule chose : afficher "Hello, Go!" dans le terminal. C'est minimaliste, mais incroyablement puissant. Pourquoi ? Parce que tout, absolument tout, part de cette fonction.

Et si on complexifie un peu ? Supposons que vous voulez demander le nom de l'utilisateur et afficher un message personnalisé. Avec Go, c'est tout aussi simple :

ex8.go

```
func main() {
    var name string
    fmt.Print("Entrez votre nom : ")
    fmt.Scanln(&name)
    fmt.Printf("Bonjour, %s ! Bienvenue dans Go.\n", name)
}
```

Là, on commence à voir la magie de Go : un code clair, lisible, mais qui fait exactement ce qu'on attend de lui. Et tout ça dans le package main.

Base du langage GO FR La suite ...

Liens Web

001 Apprendre le lange GO FR

002 Apprendre le lange GO FR

Le Livre chez DUNOD : le langage GO FR

Tutos

go-par-l-exemple.pdf