# DEV1 : les modules en Go

[loig.jezequel@univ-nantes.fr](mailto:loig.jezequel@univ-nantes.fr)

## Pour l’instant

* Un fichier par exercice : pas pratique pour un gros projet.
* Toujours un main : pas adapt´e pour une biblioth`eque.

Biblioth`eque

Ensemble de fonctions prˆetes `a ˆetre utilis´ees par des programmes.

## Le cours d’aujourd’hui

* Comment organiser son code pour un gros projet ?
* Comment fabriquer ses propres biblioth`eques ?
* Comment g´erer les d´ependances de son code `a des biblioth`eques externes ?

## Organiser son code

Au sein d’un module on peut organiser son code en plusieurs fichiers et dossiers pour bien s´eparer les diff´erentes fonctionnalit´es.

## Fabriquer ses biblioth`eques

Un module est un moyen de compartimenter son code et de le distribuer aux autres programmeurs.

## G´erer les d´ependances

Un module enregistre les biblioth`eques utilis´ees par le code qu’il contient avec des informations pr´ecises sur la version exacte `a utiliser

Paquet (*package*)

Collection de fichiers sources, tous situ´es dans le mˆeme r´epertoire, et qui sont compil´es ensemble.

* Code partag´e entre les fichiers (une fonction d´efinie dans un fichier est utilisable dans un autre).
* Unit´e d’importation (en Go on importe des paquets).

## Module

Collection de paquets qui sont publi´es, versionn´es et distribu´es ensemble.

* Identifi´e par un chemin (*module path*).
* Fichier go.mod `a la racine.
* Fichier go.sum `a la racine.

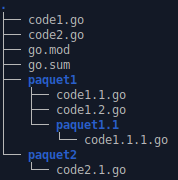


Figure: Organisation classique d’un module

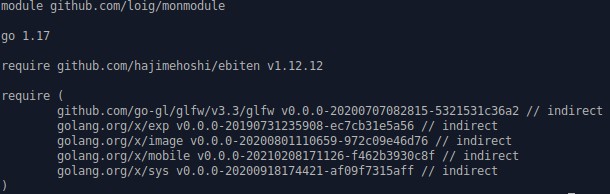


Figure: Contenu d’un fichier go.mod

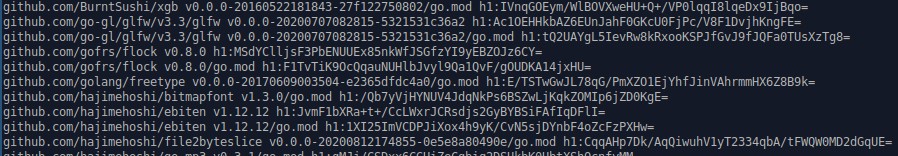


Figure: Contenu (partiel) d’un fichier go.sum

## Commande (dans le dossier ou` sera le code)

go mod init chemin

## Choix du chemin

Le chemin d’un module permettra de l’importer dans d’autres programmes. Il doit respecter certaines r`egles.

Biblioth`eque publique le chemin est l’adresse ou` la trouver. Biblioth`eque priv´ee le chemin est ce qu’on veut.

Programme ex´ecutable le chemin est ce qu’on veut.

Ceci permet d’assurer qu’il n’y a pas deux biblioth`eques (modules) utilisables par tous qui ont le mˆeme chemin.

## Commande (`a la racine du module)

go build

## Main

Si votre module contient de quoi s’ex´ecuter `a la racine (package main + fonction main), un ex´ecutable du nom du module sera produit.

## Commandes alternatives (`a la racine du module)

Installation go install

Les fichiers compil´es sont plac´es dans l’espace de travail Go (variable d’environnement).

Ex´ecution go run

Seulement si le module contient un main, il s’ex´ecute directement.

## Pour quoi faire ?

Rendre le code plus lisible, mieux organis´e, en le s´eparant en plusieurs parties plus ou moins ind´ependantes.

Un exemple

Le paquet *image* de la biblioth`eque standard (cr´eation et manipulation d’images en Go) contient un paquet d´edi´e au traitement des images png et un autre d´edi´e aux images jpeg.

## Comment ?

1. Cr´eer un dossier avec le nom du paquet,
2. faire commencer les fichiers source Go plac´es dans ce paquet par *package xxx* ou` xxx est le nom du paquet.

## Nom long pour l’importation

Pour importer un paquet on utilise son nom long, c’est `a dire le chemin du module suivi du chemin du paquet depuis la racine du module.

## Nom court pour l’utilisation

Pour utiliser un paquet on se sert de son nom court, c’est `a dire le nom du dossier ou` sont les sources (qui est aussi celui qu’on indique au d´ebut des fichiers du paquet).

Visibilit´e, attention

Les fonctions/variables/types d´efinis dans un paquet sont visibles (c’est-`a-dire utilisable `a l’ext´erieur du paquet) si et seulement si leur nom commence par une majuscule.

## Biblioth`eque standard

* Nom court pour importer et utiliser.

## Module disponible publiquement

* Nom long pour importer (et nom court pour utiliser).
* Commande go mod tidy pour r´ecup´erer automatiquement le paquet si n´ecessaire, ainsi que mettre `a jour automatiquement go.mod et go.num.

## Module M non disponible publiquement

* Nom long pour importer (et nom court pour utiliser).
* Ajouter replace M =*>* chemin dans go.mod.
* Commande go mod tidy pour r´ecup´erer le paquet si n´ecessaire, ainsi que mettre `a jour automatiquement go.mod et go.num.

## Structurer son code Go

<https://golang.org/doc/code>

## G´erer les d´ependances de ses programmes

<https://golang.org/doc/modules/managing-dependencies>

## Fonctionnement pr´ecis des modules

<https://golang.org/ref/mod>