Install Go

The official installation instructions for Go are available here.

Go Environment

Go Modules

Go 1.11 introduced Modules. This approach is the default build mode since Go 1.16, therefore the use of `GOPATH` is not recommended.

Modules aim to solve problems related to dependency management, version selection and reproducible builds; they also enable users to run Go code outside of `GOPATH`.

Using Modules is pretty straightforward. Select any directory outside `GOPATH` as the root of your project, and create a new module with the `go mod init` command.

A `go.mod` file will be generated, containing the module path, a Go version, and its dependency requirements, which are the other modules needed for a successful build.

If no `<modulepath>` is specified, `go mod init` will try to guess the module path from the directory structure. It can also be overridden by supplying an argument.

```
mkdir my-project
cd my-project
go mod init <modulepath>
```

A `go.mod` file could look like this:

```
module cmd

go 1.16
```

The built-in documentation provides an overview of all available `go mod` commands.

```
go help mod
go help mod init
```

Go Linting

An improvement over the default linter can be configured using GolangCI-Lint.

This can be installed as follows:

```
brew install golangci-lint
```

Refactoring and your tooling

A big emphasis of this book is the importance of refactoring.

Your tools can help you do bigger refactoring with confidence.

You should be familiar enough with your editor to perform the following with a simple key combination:

- **Extract/Inline variable**. Being able to take magic values and give them a name lets you simplify your code quickly.
- **Extract method/function**. It is vital to be able to take a section of code and extract functions/methods
- **Rename**. You should be able to confidently rename symbols across files.
- **go fmt**. Go has an opinioned formatter called `go fmt`. Your editor should be running this on every file save.
- **Run tests**. You should be able to do any of the above and then quickly re-run your tests to ensure your refactoring hasn't broken anything.

In addition, to help you work with your code you should be able to:

> **View function signature**. You should never be unsure how to call a function in Go. Your IDE should describe a function in terms of its documentation, its parameters and what it returns.

> **View function definition**. If it's still not clear what a function does, you should be able to jump to the source code and try and figure it out yourself.

> **Find usages of a symbol**. Being able to see the context of a function being called can help your decision process when refactoring.

Mastering your tools will help you concentrate on the code and reduce context switching.

Wrapping up

At this point you should have Go installed, an editor available and some basic tooling in place. Go has a very large ecosystem of third party products. We have identified a few useful components here. For a more complete list, see https://awesome-go.com.

Previous
Learn Go with Tests

Next
Hello, World

Last updated 3 months ago