The Go Tool

Ian Harris

December 14, 2018

1 The Go Tool

So, we're going to talk about the Go Tool a little bit, just overview of it really. It has a lot of features, and we'll get to those in different courses actually during the specialization. We'll talk about a little bit of it now, but start off with import.

1.1 `import` Keyword

So, just to restate what `import` does. It's a keyword, and it's used to access other packages. Now, for the most part, the packages that we're going to be importing will be the built-in packages, the ones that come with the Go language, to implement different functions that we're going to use in the course. So, for instance, right now, right off the start, we are going to use this format package, `fmt`, and it has a `printf` statement built into it, and we use it for printing things.

1.2 GOROOT and GOPATH

Now, what happens is when you do an import, the Go Tool when it does a build, it has to find the imported packages. So, it searches through the directory specified by the GOROOT and the GOPATH environment variables. So, if you keep everything inside the GOPATH and the GOROOT, so inside your work space, it'll find them. If you decide you want to import some package from some other place and maybe it's installed in a different directory, something like that, then you're going to have to change your GOPATH and GOROOT paths. You're going to have to increase them, change the path, change the environment variables, so that it can find them. But, that won't be a problem for based of the majority of this course, we're not doing that. But I'm saying in the future, when you're working with really big code, you might need to alter these environment variables in order to be able to find the packages that you're looking for.

1.3 Go Tool Commands

So, the Go Tool. When you download Go, you get this Go Tool, and it's a general tool used to manage Go source code. There are many commands, a bunch of different commands that you can use the Go Tool to do.

1.3.1 `go build` Command for Compilation

The first one is going to be `go build`. So that, it's just compiling the program. The arguments to `go build`, you can have no arguments, in which case it just compiles a `.go` file in the local directory. But you can give it a bunch of packages, a bunch of package names, or a bunch of `.go` files that you want to build. You can give that as the arguments to this `go build` command. It'll go build whatever you tell it to build, or you could just say `go build`. Actually, that's what I did in the demo. I just said `go build`, and I was already in the directory where I had my `main` package, and so it just compiled that. So, it creates an executable for the `main` package, and the executable has the same name as the first `.go` file. So, if you're just using one `.go` file, you're just going to get that as the name. The `.exe` suffix is what you're going to see for executables in Windows in general. So, you'll expect to see a `.exe`, and it's executable, and that should be in the directory where you did the build. If you give it no other arguments, it'll just place it in the same directory.

1.3.2 Go Tool Arguments

Now, there are tons of arguments to these commands, and I'm not really going to go through, but you can have arguments where you can tell it to build and put the executable in a different directory and so on. I'm not going to do that right now. We'll finesse that stuff later.

1.3.3 `go doc` Command

So, some of the other Go Tool commands, just go through these a little bit. `go doc`. `go doc` prints documentation for a package. Now, we'll go over this later, but as a programmer, you have to put the documentation in your package, and `go doc` will just pull it out of all your packages, and print it.

1.3.4 `go format` Command

`go format`, that format source code files. So, we're not going to get heavily into this, but if you program at all, you must have heard arguments about, "Oh, you need this type of indentation and stuff like this." So, this `go format` will just indent it the way it should be done. You just give it the source code file, and it'll indent it right to get past all those arguments. There's a standard indentation. You don't have to use it. Remember, the indentation isn't forced on you. This isn't Python or something like that. You don't have to, but `go format` will do it for you, so why not.

1.3.5 `go get` Command

Go get downloads packages and installs them. So, if you want to get new packages that do interesting things that aren't standard default packages, you can say `go get` and give the name of the package, it'll go online, find the package, download it.

1.3.6 `go list` • `go run` • `go test` Commands

`go list`, list install all the install packages. `go run` compiles go file, and it runs the executable. So if you just say `go build`, that compiles it, and does not execute it. But `go run` actually compiles it and then executes the executable in the end. Or, if it's already compiled, it will just run the executable. Now, you don't need `go run`, like in my demo, I think I did a `go build` to get the executable. I think it was called `hello.exe`. Then, I just typed `hello.exe` at the command line, and it executed it. So I didn't have to use `go run` in order to run the executable but you can. `go test`, actually the last course, the fourth in this specialization is actually about testing, and we'll get to that then. `go test`, it runs tests. It looks stuff, basically, you have a bunch of test files that end with this underscore test `.go`, and you can run these tests using the go test command. But we'll cover that later.