Don’t repeat yourself

Let the Go compiler do it for you!

Ben Bieker
@Blackgentoo
wwwdata

About me

- Hi, I am Ben
- Working with go for about 1.5 years
- api2go open source library
- mostly using go for developing rest backends
- had no idea about go generate but found the topic very interesting :D
- unfortunately, not a lot of go at current job :(

Go generate introduction

- Introduced in go 1.4
- Special command that can be run before a `go build`
- Scans code for special comments to generate code

Goals

They have an official Proposal document in Google Docs

- Executing all kinds of useful tools before compiling code
- yacc: generating .go files from yacc grammar (.y) files
- protobufs: generating .pb.go files from protocol buffer definition (.proto) files
- HTML: embedding .html files into Go source code
- bindata: translating binary files such as JPEGs into byte arrays in Go source
- Not intended to replace the make utility, because of lack of dependency analysis. But when go generate is used, a lot of make uses can be replaced with it

```go
package main

//go:generate go-bindata -o myfile.go data/
func main() {

}
```

Security Concerns

```
//go:generate rm -rf /
const (
  Whatever sth…
)

func main() {}
```

Stringer example

- Stringer is an interface that can be implemented to represent a custom type as a string
- There also is a tool named Stringer that can help with that.
- Example…

Jsonenums

- https://github.com/campoy/jsonenums
- Validate JSON with custom types and custom MarshalJSON/UnmarshalJSON methods that check for the values.
- Example…

go/parser

- How to write your own tool?
- go/parser can be used to read go source files
- go/ast defines types to represent the syntax tree
- go/token defines constants representing the lexical tokens of the Go programming language and basic operations on tokens (printing, predicates).
- Very minimal example of how to read out constants…

go tool yacc

- Yacc is a tool that generates code to parse a defined grammar.

- For example a calculator that has operations like +, -, *, /

- The code to write by hand would be very tedious.

- There is an example of a calculator app in the go source code but there is no good documentation :(and it’s hard to find some.

Summary / when is code generation useful?

- When you have repetitive code that follows the same schema
- When converting from one format to another
- Keep your code clean, easy to read, maintainable and type safe.