

`go generate` is only useful if you have tools to use it with! Here is an incomplete list of useful tools that generate code.

- `goyacc` – Yacc for Go.
- `stringer` – Implements `fmt.Stringer` interface for enums.
- `gostringer` – Implements `fmt.GoStringer` interface for enums.
- `jsonenums` – Implements `json.Marshaler` and `json.Unmarshaler` interfaces for enums.
- `go-syncmap` - Generates Go code using a package as a generic template for `sync.Map`.
- `go-syncpool` - Generates Go code using a package as a generic template for `sync.Pool`.
- `go-atomicvalue` - Generates Go code using a package as a generic template for `atomic.Value`.
- `go-nulljson` - Generates Go code using a package as a generic template that implements `database/sql.Scanner` and `database/sql/driver.Valuer`.
- `go-enum` - Generates Go code using a package as a generic template which implements interface `fmt.Stringer` | `binary` | `json` | `text` | `sql` | `yaml` for enums.
- `enumerator` - Generates Go code that convert Go enum to/from strings.
- `go-import` — Performs auto import of non go files.
- `gojson` - Generates go struct definitions from example json documents.
- `vfsген` - Generates a `vfsdata.go` file that statically implements the given virtual filesystem.
- `goreuse` - Generates Go code using a package as a generic template by replacing definitions.
- `embedfiles` - Embeds files into Go code.
- `ragel` - State machine compiler
- `peachpy` - x86-64 assembler embedded in Python, generates Go assembly
- `bundle` - Bundle creates a single-source-file version of a source package suitable for inclusion in a particular target package.
- `msgp` - A Go code generator for MessagePack
- `protobuf` - protobuf
- `thriftrw` - thrift
- `gogen-avro` - avro
- `swagger-gen-types` - go types from swagger specifications
- `avo` - generate assembly code with Go
- `Wire` - Compile-time Dependency Injection for Go
- `sumgen` - generate interface method implementations from sum-type declarations
- `interface-extractor` - generates an interface of a desired type, with only methods used within the package.
- `deep-copy` - creates a deep copy method for the given types.
- `libfsm` - fsm toolkit supporting (among others) Go and Go-flavored amd64 assembly for matching regexps
- `re2c` - lexer generator for C, C++ and Go

- re2dfa - Transform regular expressions into finite state machines and output Go source code
- pigeon - a PEG parser generator for Go