VIM-GO

Mittwoch, 4. Oktober 2017 14:39

[TUTORIAL](https://github.com/fatih/vim-go-tutorial)

INSTALLATION

Fetch and install vim-plug along with vim-go:

```
$ curl -fLo ~/.vim/autoload/plug.vim --create-dirs
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
$ git clone https://github.com/fatih/vim-go.git ~/.vim/plugged/vim-go
```

Add following content to ~/.vimrc:

```
set autowrite
let g:go_test_timeout = '10s'
let g:go_fmt_command = "goimports"
let g:go_metalinter_autosave = 1

set runtimepath+=~/.vim/plugged/ultisnips

call plug#begin()
Plug 'fatih/vim-go', { 'do': ':GoInstallBinaries' }
Plug 'AndrewRadev/splitjoin.vim'
Plug 'SirVer/ultisnips'
Plug 'honza/vim-snippets'
call plug#end()
```

From <https://github.com/fatih/vim-go-tutorial>

! For UltiSnips you may have to create

UltiSnips also needs that Vim sources files from the

ftdetect/ directory.

Unfortunately, Vim only allows this directory in the .vim directory. You

therefore have to symlink/copy the files: >

```
mkdir -p ~/.vim/ftdetect/
ln -s ~/.vim/ultisnips_rep/ftdetect/* ~/.vim/ftdetect/
```

From <https://github.com/SirVer/ultisnips/blob/master/doc/UltiSnips.txt>

Call

! **:PlugInstall** after addering new Plugins!!

:GoInstallBinaries - Install all required programs to the GOPATH (gogetdoc, guru, golint, fillsruckt, godef, motion, …)

For UPDATES call:

:GoUpdateBinaries

BASICS

:GoRun

:GoBuild
:GoInstall
❗ **:set autowrite**

TEST
:GoTest
:GoTestFunc (you have to go inside the TestBar function and call :GoTestFunc)
:GoTestCompile (only compiles the test file, without running it.)

:GoCoverage - Check the tests coverage with a syntax highlighting.
:GoCoverageClear - Clear highlightning
:GoCoverageToggle - Toggle highlightning.

FORMATTING
:GoFmt

IMPORTS
:GoImport strings - Imports the strings package automatically if needed
:GoImportAs str strings - Imports the strings package as str.
:GoDrop strings - removes the strings packages from the imports declarations
let g:go_fmt_command = "goimports" - add to have the imports added automatically when the file is saved
You can:
❗ **:set autowrite -** save the file automatically when you call :GoBuild

⭐ **SELECT FUNCTIONS**
:vaf - selects (visualizes) a whole function
:vif - visualizes text inside a function

SNIPPETS
See a list of all snippets at:
https://github.com/fatih/vim-go/blob/master/gosnippets/UltiSnips/go.snippets

Usage:
Type **errp** <tab> to add new "error panic" snippet
Type **json** after a variable in a struct to insert the json formated variable name for json.Marshal

Print variables:
fn -> fmt.Println()
ff -> fmt.Printf()
ln -> log.Println()
lf -> log.Printf()

From <https://github.com/fatih/vim-go-tutorial>

CODE CHECK
:GoMetaLinter - Runs go vet, golint, and errcheck automatically
:GoLInt
:GoVet

⭐ **NAVIGATION**
:GoAlternate - Change between go-File and go-Test-File
:GoDef - **gd** -Jump to the definition of the declaration.
:GoDefPop -ctrl t - Jumps back to the Go declaration
Ctrl-o - Jumps back to the last vim jump position.

If Errors in QUICKFIX

Cmd	**Abrevatiation**	**Description**
:cprevious	:cp	change to previous error
:cnext	:cn	Change to next error
:cclose		Close quickfix view

If Errors in LOCATION LISTS

Cmd	Abreviations	Description
:lnext		Change to the next line
:lprevious		Change to the previous line
:lclose		Close view

NAVIGATION BETWEEN FUNCTIONS

! Install it adding **ctrlp** Plugin to your plug directives in .vimrc:

- **Plug 'ctrlpvim/ctrlp.vim'j**
- Run **:PlugInstall** to install it.

:GoDecls - Shows a list of all declarations in the file
:GoDeclsDir - Shows a list of all declarations in all files of the actual directory
(Use ↑↓ to navigate and ENTER to jump to them)

]] - Jumps to the next function
[[- Jumps to the previous function
3]] - if you are at the top, you will jump to the 3th function
v]] - it will select the next function

DOCUMENTATION

:GoDoc - K - shows the documentation about the topic under the cursor

:GoInfo - print the function signature in the status line
★ Let it happens automatically with a delay of only 100ms:

- **let g:go_auto_type_info = 1**
- **set updatetime=100**

:GoSameIds - highlights all matching identifiers
:GoSameIdsClear - Clears the highlighting function
★ Make it happen automatically:

- **let:go_auto_sameids = 1**

DEPENDENCIES AND FILES

:GoFiles - shows the files that make a package
:GoDeps - shows the dependencies of a file.

GURU

References
:GoReferrers - Finds the references to the selected identifier, scanning all necessary packages within workspace.

Definition, Methods and Struct-Fields
:GoDescribe - Shows the definition of the field, the method set and the struct's fields

Interfaces
:GoImplements - Displays a quickfix list showing our selected type and the interface it implements.

Errors thrown
:GoWhicherrs - returns a list containing all errors that can be returned.

Channels tracking
:GoChannelPeers - Displays allocation, sender and receivers of a channel

Call Targets
:GoCallees - shows the possible call targets of the selected function call.
:GoCallers - shows the callers of this function
:GoCallstack - shows an arbitrary path from the root of the call graph to the function containing the selection.

Scope
:GoGuruScope github.com/fatih/vim-go-tutorial - Sets the scope to this prj
:GoGuruScope … - sets the scope to be GOPATH
:GoGuruScope github.com/… golang.org/x/tools - sets multiple scopes
:GoGuruScope encoding/… -encoding/xml - whit -(neg)sign exclude pkgs.
:GoGuruScope "" - clears the scope

Set the scope in .vimrc
let g:go_guru_scope = ["github.com/fatih/vim-go-tutorial"]
let g:go_guru_scope = ["..."]
let g:go_guru_scope = ["github.com/...", "golang.org/x/tools"]
let g:go_guru_scope = ["encoding/...", "-encoding/xml"]

REFACTOR

Rename
⭐ **:GoRename newname** - identifier under cursor and all references all renamed to "newname".

Extract
Has to be done manually :-(
:GoFreevars - shows the variables that are dependent of a selected block of code.

CODE GENERATION
:GoImpl - Creates the implementation stuff for the interface that you type for the classifier under the cursor
:GoImpl io.ReadWriteCloser - Creates the io.ReadWriterClose implementation stuff for the definer under the cursor
:GoImpl b *B fmt.Stringer - Creates the fmt.Stringer implementation stuff of the Stringer interface for the b definer of type *B.

SHARE IT
:GoPlay - publics the code on https://play.golang.org and creates a link, which you can share. Select only a piece of code to share is possible.