Home Company Services Case Studies Blog Contact Us

GOLANG • WEB DEVELOPMENT

12 Best Practices Every Golang Developer Must Follow

BY REDDY SAI • JULY 9, 2018

READ 5 MINUTES

This blog is aimed to act as a practical guide of Golang to implement the Go best practices and design patterns which could help the developers achieve the greater results. You must have already seen in the Go Tutorial and Effective Go.

Newsletter

Join the weekly newsletter and never miss out on new tips, tutorials, and more.

First name

Last name

E-mail

SUBSCRIBE

Search For

Search ...

Case Studies

detailed samples.

These are the Go best practices or techniques that has consistently shown results for superior those achieved great results

Listing down the few of best techniques to write Go code that is simple, readable, Maintainable.

- Use gofmt
- Avoid nesting by handling errors first
- Error Strings
- Handling Errors
- Avoid Repetition When Possible
- Variable Name Declaration
- Type Switch To Handle Special Cases

Recent Posts

- Important Code Needs To Be Served First
- Import Dot
- Document Your Code
- Comment Sentences

Use gofmt

Run gofmt on your code to automatically fix the majority of mechanical style issues. Almost all Go coders in the world uses gofmt.
gofmt will read the go program and it will show you the properly aligned output after indentation, vertical alignment and even it can re-format the comments too.

Commands and options

gofmt filename – This will print re-formatted code.
gofmt -w filename – This will

Archives

Select Month

Categories

Big Data (3)
Blockchain (24)
CEO Diaries (15)
Cryptocurrency (4)
DevOps (27)
General (151)
Geographical Information System(1)
IOT (1)
Mobile App Development (41)
Corporate Events (13)
Tech News (8)

the rewrite rule to the source before reformatting.

gofmt /path/to/ package – This will format the whole package.

Here is a small example for gofmt

filename: demo.go

```
1  package main
2          import "fmt"
3  // this is demo to format code
4            // with gofmt commar
5  var a int=10;
6             var b int=15;
7                          var
8       func print(){
9                  fmt.Println("
10                       fmt.Prir
11
12
13                        }
```

Passing a Command: $ **gofmt demo.go**

Web Development (317)

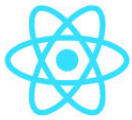
Weekly Roundup (1)

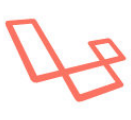
Our Technologies


Python


Symfony


AngularJS


ReactJS


Laravel


Golang


PHP


IOS


Android


HTML5


Django


MySQL


JQuery


NodeJs

```
2  import "fmt"
3  // this is demo to format code
4  // with gofmt command
5  var a int = 10
6  var b int = 15
7  var c string =  “Welcome to Ag
8  func print() {
9          fmt.Println("Value for
10         fmt.Println(a)
11         fmt.Println((b))
12         fmt.Println(c)
13 }
```


Avoid Nesting By Handling Errors First

Instead of giving multiple or nested conditions, We can break the condition if we ought to face error while processing and proceed further with coding.

```
1 err := request()
2 if err != nil {
```

```
6 }
```

Instead you can do like this:

```
1 err := request()
2 if err != nil {
3  // handling error
4  return // or continue, etc.
5 }
```

// proceed to further

Less nesting means less cognitive load on the reader.

If the **if statement** has an initialization statement such as:

```
1 if x, err := f(); err != nil {
2    // handling error
3    return
4 } else {
5    // use x
6 }
```

```
1 x, err := f()
2 if err != nil {
3     // handling error
4     return
5 }
6 // use x
```

Error Strings

Error strings should not be capitalized (unless beginning with proper nouns or acronyms).

For Example:

Instead of thics command fmt.Errorf("Something went wrong") we can move with this fmt.Errorf("something went wrong")

Handling Errors

Do not discard errors using _ variables. If a function returns an error, check to make sure whether the function succeeded or not.

situation occurs.

Don't use panic errors

Don't use panic for normal error handling. In that case you can use error and multiple return values.

Avoid Repetition When Possible

suppose if you want use structures in controllers, as well as models, Create one common file and there you can create the structures.

Variable Name Declaration

Variable names in Go should be short rather than long. This is especially true for local variables with limited scope.

Prefer `c` **to** `lineCount`

Prefer `i` **to** `sliceIndex`

descriptive.

- For a method receiver, one or two letters are sufficient.
- Common variables such as loop indices and readers can be a single letter (i, r).
- More unusual things and global variables need more descriptive names.

Type Switch To Handle Special Cases

Suppose if you're not sure about what the **interface{}** type could be, you can use a type switch:

For Example:

```
1 func Write(v interface{}) {
2  switch v.(type) {
3  case string:
4    s := v.(string)
5    fmt.Printf("%T\n",s)
6  case int:
7    i := v.(int)
```

```
}
```

Related: Message Queues in Golang Via RabbitMQ

Type Switch With The Short Variable Declaration

Declare a variable and it will have the type of each `case`:

For Example:

```
1 func Write(v interface{}) {
2  switch x := v.(type) {
3  case string:
4    fmt.Printf("%T\n",x)
5  case int:
6    fmt.Printf("%T\n",x)
7  }
8 }
```

If you have the importannt information like License information, build tags, package documentation then describe it earlier.
We can separate the Import statements, related groups by blank lines.
The standard library packages are in the first group.

```
1 import (
2     "fmt"
3     "io"
4     "log"
5
6   "golang.org/x/net/websocket"
7 )
```

The rest of the code starting with the most significant types and ending
with helper function and types.

Import Dot

The **import . form** can be used to test the circular dependencies.

```
1 package foo_test
2 import (
3 "bar/testutil" // also imports "
4 . "foo"
5 )
```

In this case, the test file cannot be in package **foo** because it uses bar/testutil, which imports foo. So we use the `import .` form to let the file pretend to be part of the package for even though it is not. Except for this one case, do not use `import .` in your programs.

It makes the programs much harder to read because it is unclear when a name like Quux is a top-level identifier in the current package or in an imported package.

Related: How To Deploy a Golang Web Application with Docker

Package name, with the associated documentation before.

```
// Package playground registers an HTTP handler at "/compile" that
// proxies requests to the golang.org playground service.
package playground
```

Exported identifiers appear in godoc, and they should be documented correctly.

```
1 // Author represents the person
2 type Author struct {
3 Elem []Elem
4 }
5 // TextElem returns the first te
6 // This is used to display the c
7 // without the contact details.
8 func (p *Author) TextElem() (ele
```

Comment Sentences

Comments documenting declarations should be full sentences, even if that seems a

documentation. Comments should begin with the name of the thing being described and end in a period:

```
1 // Request represents a request
2 type Request struct { ...
3
4 // Encode writes the JSON encod
5 func Encode(w io.Writer, req *Re
```

Hope these Go best practices will help you to improve the quality of your code. We have listed out some more best practices in various technologies which can be found in our largest blog repository. For more inquires reach us at info@agiratech.com

We offer Golang development services for building world-class enterprise apps. We have expertise in building most complex software solutions using Google's Go language. Chat with us now and hire golang developers within 72 hours.

Reddy Sai

Young Senior Software Developer having 3+years of experience in Web development, skilled in Golang, Ruby, AngularJS, Ruby on Rails & NodeJS. With a handful of skills, he keeps thriving through all the Edge of Web development to become a perfect back end developer. Besides, this young techie will always buzz up & surprise us with his beautiful Movie collections.

DEVELOPER

BEST PRACTICES FOR GOLANG DEVELOPERS

DOCUMENTING GO CODE

GO BEST PRACTICES

GO WEB APPLICATION STRUCTURE

GOLANG BEST PRACTICES

GOLANG BEST PRACTICES GITHUB

GOLANG GLOBAL VARIABLES BEST PRACTICES

GOLANG GOROUTINE BEST PRACTICES

GOLANG PROJECT STRUCTURE BEST PRACTICE

GOLANG SKELETON

HOW TO STRUCTURE GO PROJECTS

One thought on “12 Best Practices Every Golang Developer Must Follow”

Guilherme Ferreira says:

In the bold sentence: ” These are the Go best practices or techniques that has consistently shown results for superior those achieved great results ”

REPLY

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name Email Website

Post C

Latest Updates

Weekly Roundup: Node.Js v12.20.0, Python3.8 Released, Trending Articles & Latest Tech updates

OCTOBER 25, 2019

Ingenious Software Deliverables by Agira Technologies Leaves a Mark at GoodFirms

OCTOBER 24, 2019

What's New with Python 3.8? Latest Features and Updates

OCTOBER 22, 2019

Join Our Team to Explore all Our Career Opportunities. Hit Here !

Do you have a project in mind?

Let's talk about what we can build together

E-mail

Subject

Your Message

Não sou um robô

reCAPTCHA
Privacidade - Termos

SEND

Latest Tech updates

Ingenious Software Deliverables by Agira Technologies Leaves a Mark at GoodFirms

What's New with Python 3.8? Latest Features and Updates

Go vs Python: Is Golang an Alternative to Python?

8 Essential Skills That Every Angular Web Developer Must Have

Laravel Development
Symfony Development
JavaScript Development
Mean Stack Development
Mobile Development

Hire Angular
Hire JavaScript
Hire Symfony
Hire Ruby on Rails
Hire Golang Developers
Hire Mean stack

Blog
why choose us