



GO WEB DEVELOPMENT 101



3 MONTHS to be productive

74% write **API SERVICES**

45% write **WEB APPS**

* Go Developer Survey 2020 Results





HELLO GOPHER

```
package main

import "fmt"

func main() {
    fmt.Println("Hello Gopher!")
}
```

Run Code

```
go build hellogopher.go // 1. Compile code
./hellogopher           // 2. Run binary
```

```
go run hellogopher.go   // Compile code + run binary
```



DEVELOPMENT



JetBrains
GoLand



Visual
Studio Code



Vim Go



5 FACTS ABOUT GO

- 1 static type system
- 2 garbage collection
- 3 no inheritance
- 4 built-in concurrency
- 5 native execution

Linux, Win, z/OS, 386, amd64, ARM, wasm, ...





VARIABLES, SLICES, LOOPS

```
1 // Variable
2 var kitty string = "Kitty"
3 bella := "Bella"
4
5 // Array (fixed length)
6 namesArray := [3]string{kitty, bella, "Cleo"}
7
8 // Slice (variable length)
9 namesSlice := make([]string, 2)
10 namesSlice[0] = kitty
11
12 // Loop
13 for i, name := range namesSlice {
14     fmt.Println("Hello " + name + "!")
15 }
```



STRUCT

```
1 type Cat struct {
2     Name string
3 }
4
5 func main() {
6     c := Cat{Name: "Kitty"}
7     fmt.Println("Hello " + c.Name + "!")
8 }
```



ERRORS

```
1 // Error as return value
2 func (c Cat) feed(food string) error {
3     if c.Name == "Kitty" && food != "Steak Tartare" {
4         return errors.New("Won't eat!")
5     }
6     return nil
7 }
8
9 func main() {
10    c := Cat{Name: "Kitty"}
11
12    // Handle error
13    err := c.feed("Caviar")
14    if err != nil {
15        fmt.Printf("%v won't eat it.", c.Name)
16    }
17 }
```



THE CATS APP



Abyssinian

The Abyssinian is easy to care for, and a joy to have in your home. They're affectionate cats and love both people and other animals.



Aegean

Native to the Greek islands known as the Cyclades in the Aegean Sea, these are natural cats, meaning they developed without humans getting involved in their breeding. As a breed, Aegean Cats are rare, although they are numerous on their home islands. They are generally friendly toward people and can be excellent cats for families with children.



CATS APP API



SET UP CATS APP

```
# Create directory  
mkdir cats  
cd cats
```

```
# Enable dependency tracking  
go mod init devopscon.com/cats
```

```
# Create go file  
touch main.go
```



CAT API SIMPLE

```
1 func catAPIHandler(w http.ResponseWriter, r *http.Request)
2     fmt.Fprintf(w, "Meow!")
3     w.WriteHeader(http.StatusOK)
4 }
5
6 func main() {
7     http.HandleFunc("/api/cats", catAPIHandler)
8     http.ListenAndServe(":8080", nil)
9 }
```



CAT API WITH JSON

```
1 type Cat struct {
2     Name string
3 }
4
5 func catAPIHandler(w http.ResponseWriter, r *http.Request)
6     cats := make([]Cat, 1)
7     cats[0] = Cat{Name: "Ginger"}
8     json.NewEncoder(w).Encode(c)
9 }
10
11 func main() {
12     http.HandleFunc("/api/cats", catAPIHandler)
13     http.ListenAndServe(":8080", nil)
14 }
```



CAT API WITH JSON

```
1 # Query cat API
2 curl -s http://localhost:8080/api/cats | jq
3 [
4   {
5     "Name": "Ginger"
6   }
7 ]
```



CAT API WITH JSON

```
1 type Cat struct {
2     Name string `json:"name"`
3 }
4
5 func catAPIHandler(w http.ResponseWriter, r *http.Request)
6     cats := make([]Cat, 1)
7     cats[0] = Cat{Name: "Ginger"}
8     json.NewEncoder(w).Encode(cats)
9 }
10
11 func main() {
12     http.HandleFunc("/api/cats", catAPIHandler)
13     http.ListenAndServe(":8080", nil)
14 }
```



TEST CAT API `cat_api_test.go`

```
1 func TestCatAPIHandler(t *testing.T) {
2     // 1. Create test request
3     req, _ := http.NewRequest("GET", "/api/cats", nil)
4
5     // 2. Create recorder (which satisfies http.ResponseWriter)
6     recorder := httptest.NewRecorder()
7
8     // 3. Invoke handler
9     catAPIHandler(recorder, req)
10
11    // 4. Check the response
12    if recorder.Code != http.StatusOK {
13        t.Errorf("Wrong status: got %v expected %v", recorder.Code, http.Sta
14    }
15 }
```



RUN TEST

```
1 # Run tests
2 go test -v ./...
3 === RUN    TestCatAPIHandler
4 --- PASS: TestCatAPIHandler (0.00s)
5 PASS
6 coverage: 50.0% of statements
7 ok      devopscon.com/cats      0.127s  coverage: 50.0% of statem
```



BUILD WITH Makefile

```
1 .DEFAULT_GOAL := build
2 BIN_FILE=cats
3
4 build:
5     go build -o dist/"${BIN_FILE}"
6
7 test:
8     go test -v ./...
9
10 run:
11     ./"${BIN_FILE}"
12
13 clean:
14     go clean
```



DEMO



CATS APP WEB



CATS HANDLER BASIC TEMPLATE SETUP

```
1 func indexHandler(w http.ResponseWriter, r *http.Request) {  
2     var tpl = template.Must(template.ParseFiles("index.html"))  
3     tpl.Execute(w, nil)  
4 }  
5  
6 func main() {  
7     http.HandleFunc("/", indexHandler)  
8     http.ListenAndServe(":8080", nil)  
9 }
```



CATS HANDLER WITH MULTIPLEXER

```
1 func indexHandler(w http.ResponseWriter, r *http.Request) {
2     var tpl = template.Must(template.ParseFiles("index.html"))
3     tpl.Execute(w, nil)
4 }
5
6 func main() {
7     mux := http.NewServeMux()
8     mux.HandleFunc("/", indexHandler)
9
10    http.ListenAndServe(":8080", mux)
11 }
```



SERVE FILES FROM FILESYSTEM

```
1 func main() {  
2     mux := http.NewServeMux()  
3     mux.HandleFunc("/", indexHandler)  
4  
5     // Serve files  
6     fs := http.FileServer(http.Dir("assets"))  
7     mux.Handle("/assets/", http.StripPrefix("/assets/", fs))  
8  
9     http.ListenAndServe(":8080", mux)  
10 }
```



SERVE FILES EMBEDDED IN BINARY

```
1 //go:embed assets
2 var assets embed.FS
3
4 func main() {
5     mux := http.NewServeMux()
6     mux.HandleFunc("/", indexHandler)
7
8     // Serve files
9     mux.Handle("/assets/", http.FileServer(http.FS(assets)))
10
11    http.ListenAndServe(":8080", mux)
12 }
```

HANDLER AND HANDLEFUNC



http

package

standard library



Version: go1.16.5

LATEST

Published: Jun 3, 2021

License: BSD-3-Clause

Jump to ...



Documentation

Overview

Index

Constants

Variables

Functions

Types

 ▶ type Client

 type CloseNotifier

 ▶ type ConnState

 ▶ type Cookie

 type CookieJar

 ▶ type Dir

 type File

 ▶ type FileSystem

func (*ServeMux) Handle

```
func (mux *ServeMux) Handle(pattern string, handler Handler)
```

Handle registers the handler for the given pattern. If a handler already exists for pattern, Handle panics.

▶ Example

func (*ServeMux) HandleFunc

```
func (mux *ServeMux) HandleFunc(pattern string, handler func(ResponseWriter, *Request))
```

HandleFunc registers the handler function for the given pattern.

func (*ServeMux) Handler

added in go1.1

```
func (mux *ServeMux) Handler(r *Request) (h Handler, pattern string)
```

Handler returns the handler to use for the given request, consulting r.Method, r.Host, and r.URL.Path. It always returns a non-nil handler. If the path is not in its canonical form, the handler will be an internally-generated handler that redirects to the canonical path. If the host contains a port, it is ignored when matching handlers.



CATS HANDLER TEMPLATE WITH DATA

main.go

```
type Cat struct {
    Name string
}

func indexHandler(w ResponseWriter, r *Request) {
    // Create cat slice
    cat := make([]Cat, 1)

    // Add cat ginger
    cat[0] = Cat{Name: "Ginger"}

    // Render template
    tpl.Execute(w, cat)
}
```

index.html

```
<body>
    <h1>Cats App</h1>
    {{ range. }}
        <h2>{{ .Name }}</h2>
    {{ end }}
</body>
```





CATS HANDLER WITH CATS API

Query Cats API

```
GET https://api.thecatapi.com/v1/breeds?limit=5
```

```
[  
  {  
    "id": "abys",  
    "name": "Abyssinian",  
    "image": {  
      "url": "https://cdn2.thecatapi.com/images/0XYvRd7oD.jpg  
    }  
  },  
  {  
    "id": "aefe",  
    "name": "Aegean",  
    "image": {  
      "url": "https://cdn2.thecatapi.com/images/ozEvzdVM-.jpg  
    }  
  },  
  ...  
]
```

Map JSON

```
type Cat struct {  
  ID   string `json:"id"  
  Name string `json:"name"  
  Image struct {  
    URL string `json:"url"  
  } `json:"image"  
}
```





CATS HANDLER WITH CATS API

```
1 func indexHandler(w http.ResponseWriter, r *http.Request) {
2     resp, err := http.Get("https://api.thecatapi.com/v1/breeds?limit=5")
3     if err != nil {
4         http.Error(w, "Cats API error", http.StatusInternalServerError)
5         return
6     }
7
8     // Create cat slice
9     cat := make([]Cat, 5)
10
11    // Read and parse body
12    defer resp.Body.Close()
13    body, _ := ioutil.ReadAll(resp.Body)
14    json.Unmarshal(body, &cat)
15
16    tpl.Execute(w, cat)
17 }
```



MIDDLEWARE

- # cross cutting functionality for all requests
(e.g. logging, authentication)
- # create a chain of handlers
router => middleware handler => application handler
- # satisfy the interface http.Handler



MIDDLEWARE TO LOG REQUESTS

```
1 func loggingMiddleware(next http.Handler) http.Handler {  
2     return http.HandlerFunc(func(w http.ResponseWriter, r *http.Reque  
3         log.Printf("%v requested URL %v", r.Host, r.URL)  
4         next.ServeHTTP(w, r)  
5     ))  
6 }  
7  
8 func main() {  
9     mux := http.NewServeMux()  
10    mux.HandleFunc("/", indexHandler)  
11  
12    http.ListenAndServe(":8080", loggingMiddleware(mux))  
13 }
```



DEMO



BASICS

Dev Setup
Variables,
Slices, Loops
Struct
Errors

CATS APP API

Handler (with JSON)
Http Test
Build

CATS APP WEB

Multiplexer
(Router)
File Server
Template
Middleware





JAN STAMER
Solution Architect
jan.stamer@comdirect.de

