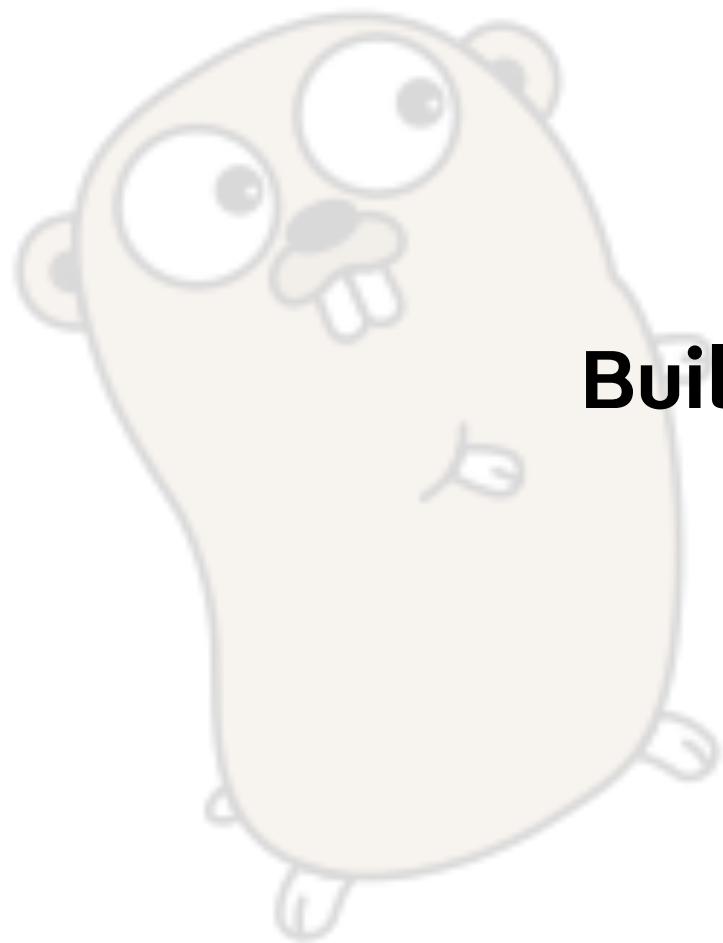


Go net/http




Build and test basic web app in Go

GO



golang : net/http

create main.go in folder chapter14-1 :



```
import (  
    "fmt"  
    "net/http"  
)  
  
func main() {  
    http.HandleFunc(  
        "/",  
        func(w http.ResponseWriter, r *http.Request) {  
            fmt.Fprintln(w, "Hello, World!")  
        })  
  
    http.ListenAndServe(":3000", nil)  
}
```

run -> no error -> push to your git repository



golang : net/http

create main.go in folder chapter14-1 :

```
func main() {  
    http.HandleFunc(  
        "/",  
        func(w http.ResponseWriter, r *http.Request) {  
            fmt.Fprintln(w, "Hello, World!")  
        })  
  
    barHandler := func(w http.ResponseWriter, r *http.Request) {  
        fmt.Fprintln(w, "Hello Bar!")  
    }  
    http.HandleFunc("/bar", barHandler)  
  
    http.ListenAndServe(":3000", nil)  
}
```

run -> no error -> push to your git repository



golang : net/http

create main.go in folder chapter14-1 :

```
type HomePageHandler struct{}

func (h *HomePageHandler) ServeHTTP(w http.ResponseWriter, r
*http.Request) {
    fmt.Fprintln(w, "Hello, World!")
}

func main() {
    ...
    http.Handle("/home", &HomePageHandler{})

    http.ListenAndServe(":3000", nil)
}
```

run -> no error -> push to your git repository



Go net/http



Query String

GO



golang : net/http

create main.go in folder chapter14-2 :

```
type HomePageHandler struct{}

func (h *HomePageHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    name := r.URL.Query().Get("name")
    if name == "" {
        name = "World"
    }
    fmt.Fprintf(w, "Hello, %s!", name)
}

func main() {
    http.Handle("/", &HomePageHandler{})

    http.ListenAndServe(":3000", nil)
}
```

run -> no error -> push to your git repository



Go net/http

working with json




```
url: localhost:3000/  
method: post  
content type: application/json  
body: {"first_name": "espresso",  
      "last_name": "longshot",  
      "email": "espresso@speedy.coffee"}  
}
```



golang : net/http

create main.go in folder chapter14-3 :



```
func main() {  
    http.Handle("/", &HomePageHandler{})  
  
    http.ListenAndServe(":3000", nil)  
}  
  
type User struct {  
    FirstName string  
    LastName  string  
    Email     string  
    CreatedAt time.Time  
}
```

run -> no error -> push to your git repository



golang : net/http

create main.go in folder chapter14-3 :

```
type HomePageHandler struct{}

func (h *HomePageHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    user := new(User)
    json.NewDecoder(r.Body).Decode(user)
    user.CreatedAt = time.Now()

    w.Header().Set("Content-Type", "application/json")
    w.WriteHeader(http.StatusCreated)

    data, _ := json.Marshal(user)
    w.Write(data)
}
```

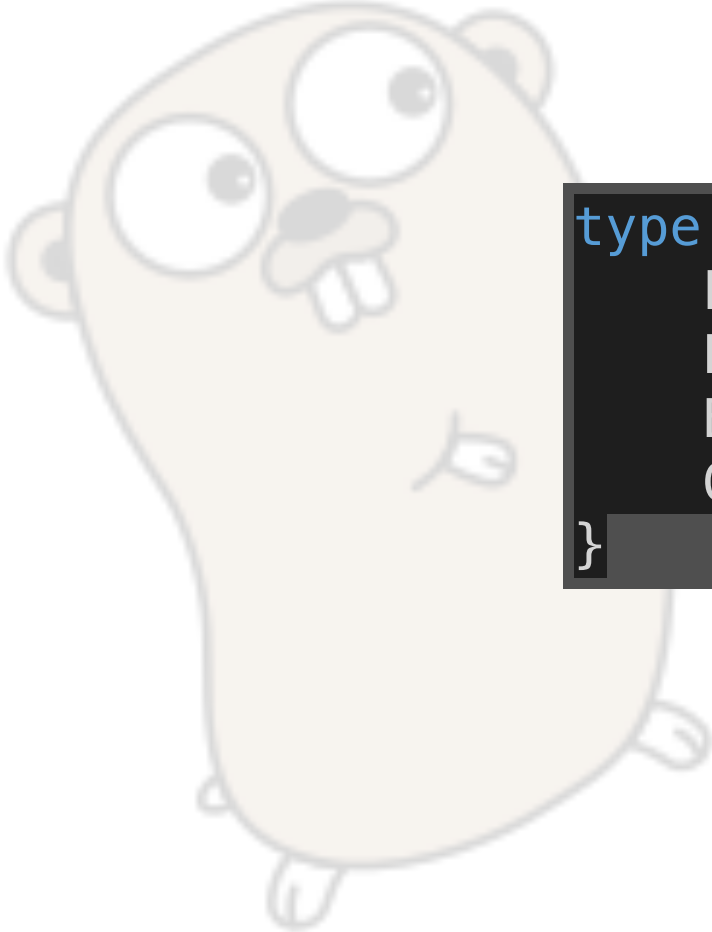
run -> no error -> push to your git repository



golang : net/http

create main.go in folder chapter14-3 :

mapping json to properties



```
type User struct {  
    FirstName string    `json:"first_name"`  
    LastName  string    `json:"last_name"`  
    Email     string    `json:"email"`  
    CreatedAt time.Time `json:"created_at"`  
}
```

run -> no error -> push to your git repository



Go net/http



test web

GO



golang : net/http

create homePageHandler_test.go in folder chapter14-4 :

```
package home

import (
    "net/http"
    "net/http/httptest"
    "testing"
)

func TestHomePageHandler(t *testing.T) {
    res := httptest.NewRecorder()
    req, _ := http.NewRequest("GET", "/", nil)
    HomePageHandler(res, req)

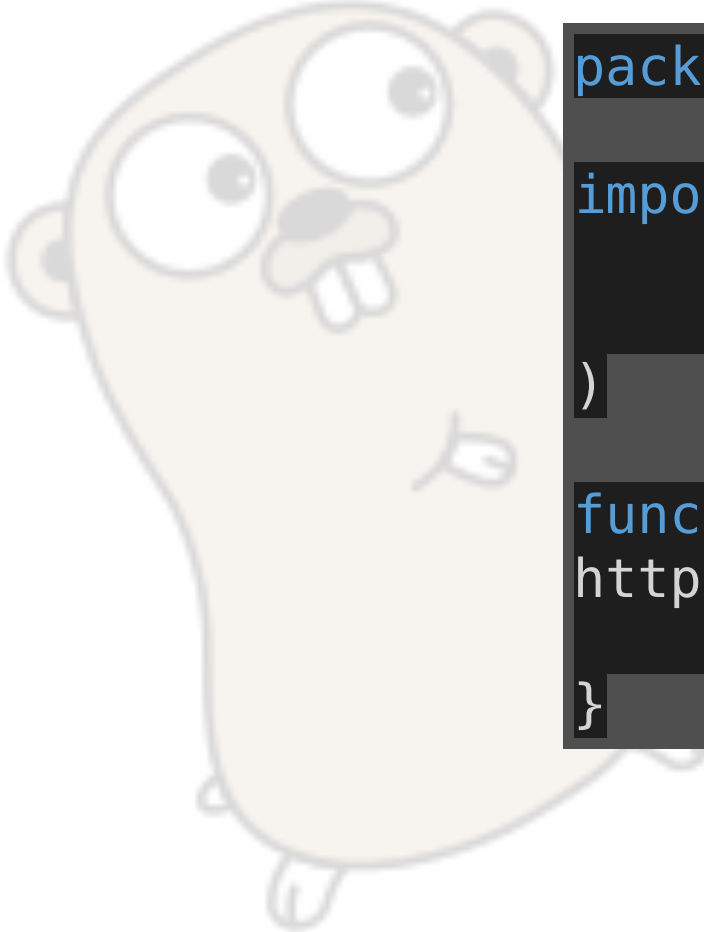
    if res.Code != 200 {
        t.Fatalf("Expected status to == 200, but got %d", res.Code)
    }
}
```

run -> no error -> push to your git repository



golang : net/http

create homePageHandler.go in folder chapter14-4 :



```
package home

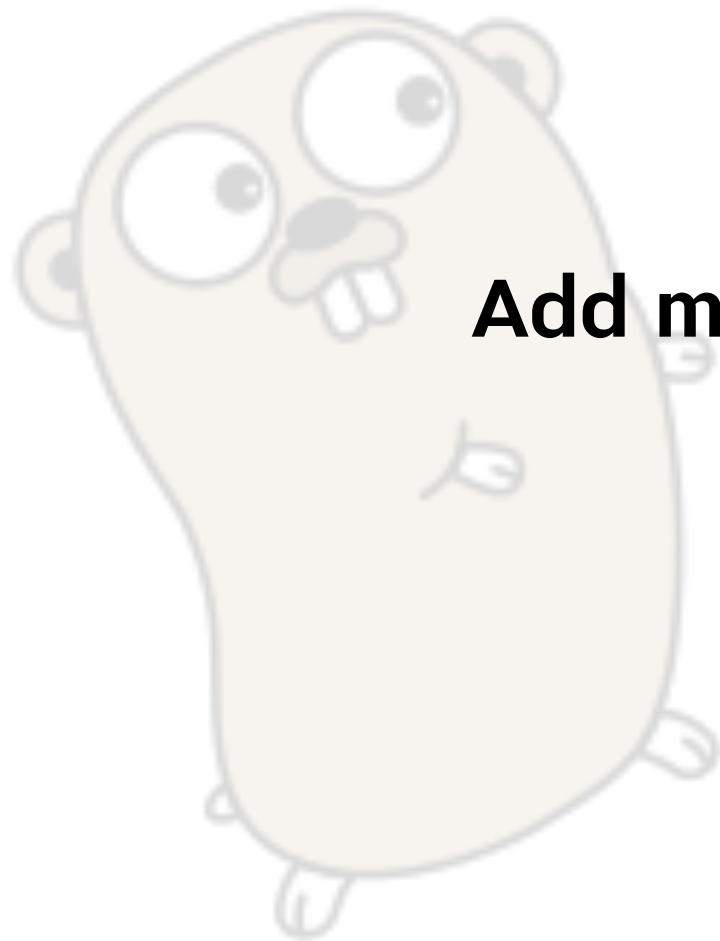
import (
    "fmt"
    "net/http"
)

func HomePageHandler(res
http.ResponseWriter, req *http.Request) {
    fmt.Fprint(res, "Hello, World!")
}
```

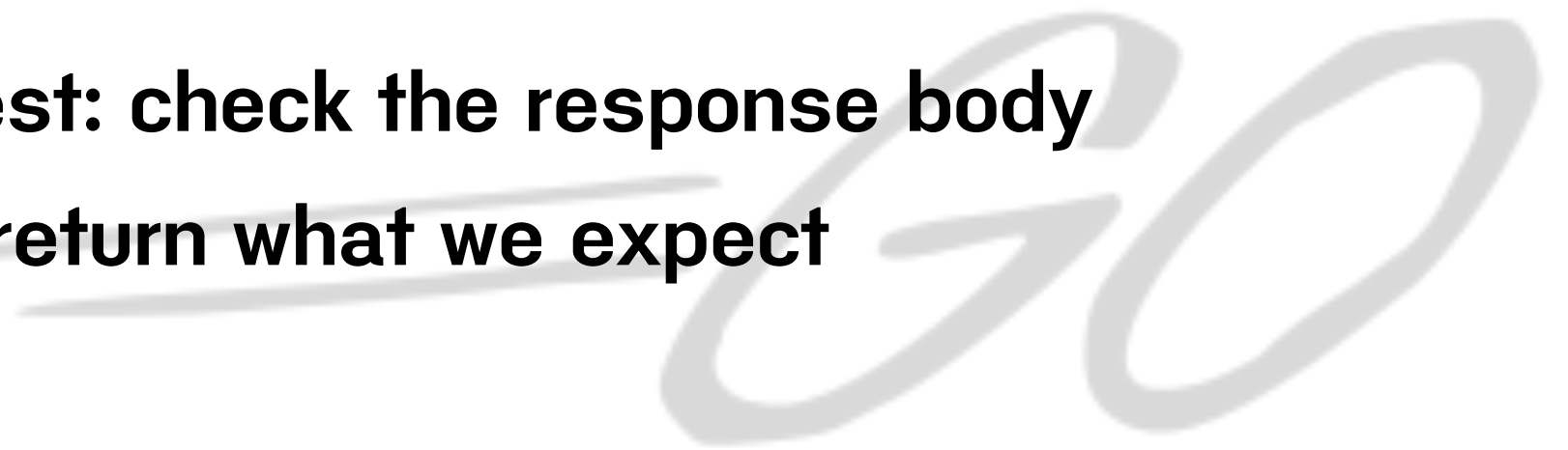
run -> no error -> push to your git repository



Go net/http



**Add more test: check the response body
that return what we expect**



golang : net/http

create homePageHandler_test.go in folder chapter14-5 :

```
func TestJsonHandler(t *testing.T) {  
    res := httptest.NewRecorder()  
    req, _ := http.NewRequest("POST", "/json",  
        strings.NewReader(`{"first_name":  
            "espresso", "last_name": "longshot",  
            "email": "espresso@speedy.coffee"}`))  
  
    home := HomePageHandler{}  
    home.ServeHTTP(res, req)  
  
    if res.Code != 201 {  
        t.Fatalf("Expected status to == 201, but got %d", res.Code)  
    }  
  
    user := new(User)  
    json.NewDecoder(res.Body).Decode(user)  
  
    if user.FirstName != "espresso" {  
        t.Fatalf("Expected user first to == espresso, but got %s",  
            user.FirstName)  
    }  
}
```

run -> no error -> push to your git repository



golang : net/http

create homePageHandler.go in folder chapter14-4 :

```
type User struct {
    FirstName string `json:"first_name"`
    LastName  string `json:"last_name"`
    Email     string `json:"email"`
    CreatedAt time.Time `json:"created_at"`
}

type HomePageHandler struct{}

func (h *HomePageHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    user := new(User)
    json.NewDecoder(r.Body).Decode(user)
    user.CreatedAt = time.Now()

    w.Header().Set("Content-Type", "application/json")
    w.WriteHeader(http.StatusCreated)

    data, _ := json.Marshal(user)
    w.Write(data)
}
```

run -> no error -> push to your git repository

