Go net/http

Build and test basic web app in 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 :

```go
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

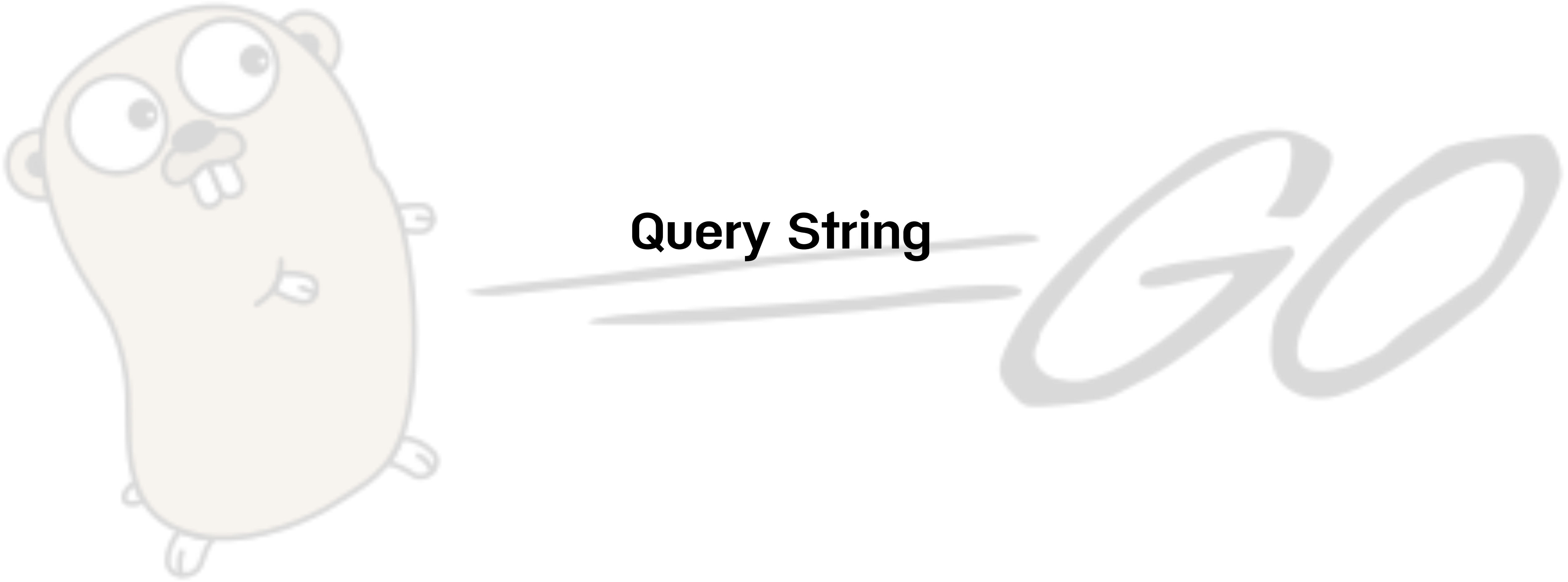
golang : net/http

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

golang : net/http

create main.go in folder chapter14-2 :

```go
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 :

```go
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 :

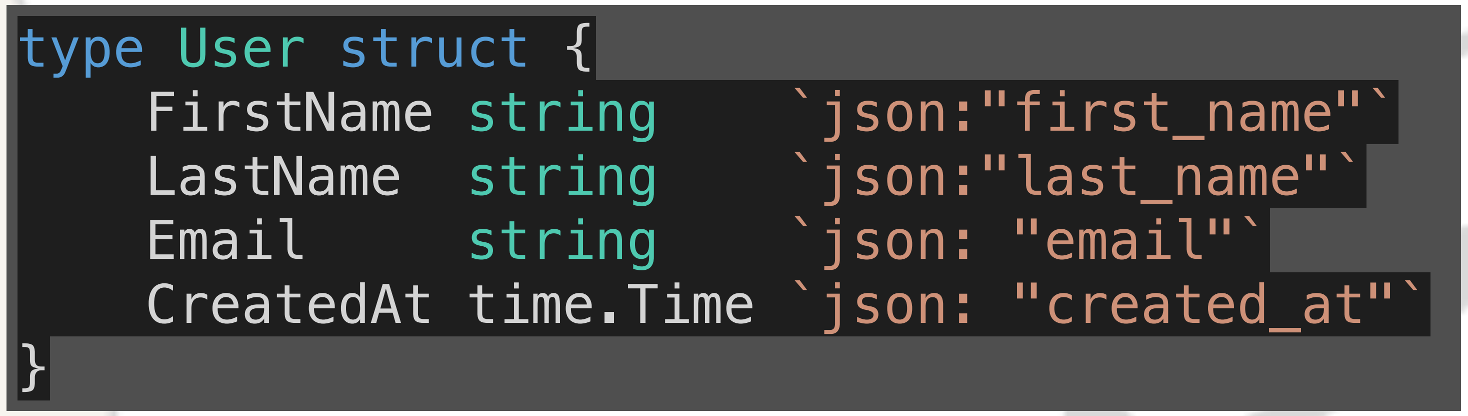
```go
type HomePageHandler struct{}

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

golang : net/http

create homePageHandler_test.go in folder chapter14-4 :

```go
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

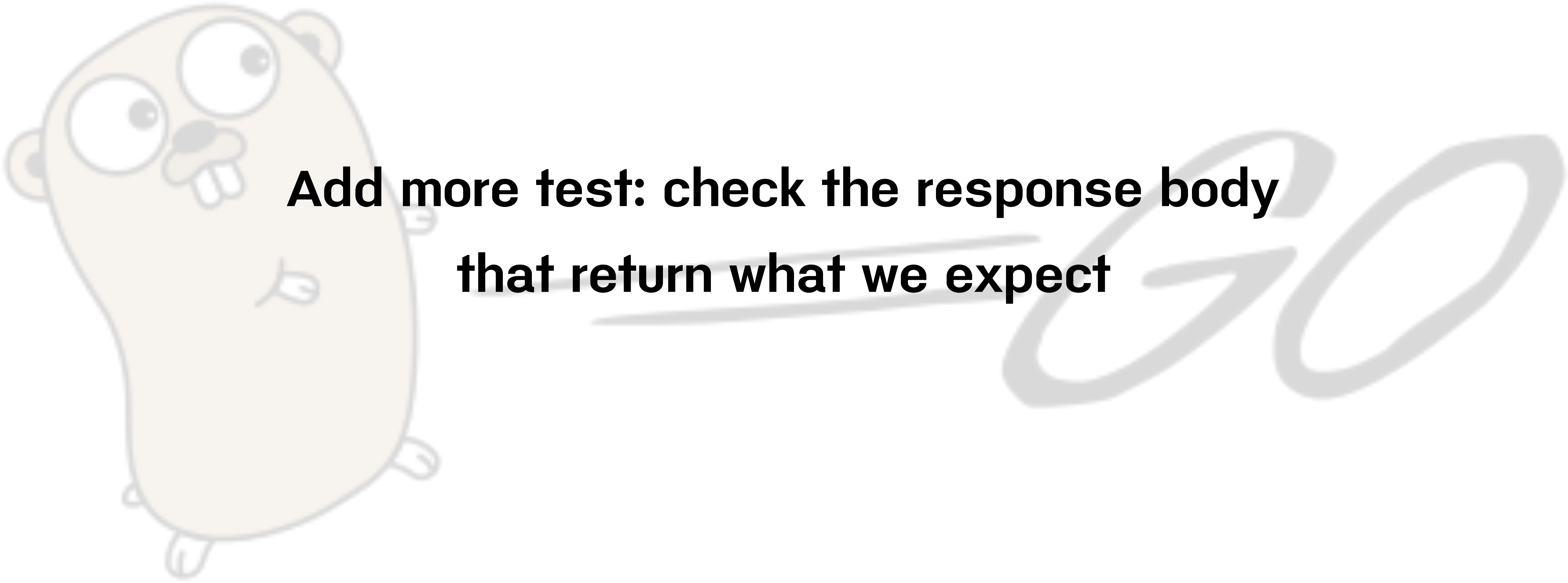
golang : net/http

create homePageHandler.go in folder chapter14-4 :

```go
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 :

```go
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 chapter14-5 :

```go
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

golang : net/http

create home.go in folder chapter14-6 :

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

func UsersHandle(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Users Page")
}

func main() {
    http.HandleFunc("/", HomePageHandle)
    http.HandleFunc(“/users", UsersHandle)
    http.ListenAndServe(":3000", nil)
}
```

What’s wrong with it?

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

golang : net/http

create home_test.go in folder chapter14-6 :

```go
func Test_Get(t *testing.T) {
    res := httptest.NewRecorder()
    req, _ := http.NewRequest("GET", "/", nil)
    HomePageHandle(res, req)

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

func Test_Post(t *testing.T) {
    res := httptest.NewRecorder()
    req, _ := http.NewRequest("POST", "/", nil)
    HomePageHandle(res, req)

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

What's wrong with it?

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

golang : net/http

github.com/gorilla/mux

```
go get -u github.com/gorilla/mux
```

golang : gorilla/mux

```go
func HomePageHandle(w http.ResponseWriter, r *http.Request) {
    vars := mux.Vars(r)
    name := vars["name"]

    if vars["name"] == "" {
        name = "World"
    }
    fmt.Fprintf(w, "Hello, %s!", name)
}

func UsersHandle(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Users Page")
}

func NewRouter() http.Handler {
    r := mux.NewRouter()
    r.HandleFunc("/{name}", HomePageHandle).Methods("GET")
    r.HandleFunc("/users", UsersHandle).Methods("GET")
    return r
}

func main() {
    http.ListenAndServe(":3000", NewRouter())
}
```

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

golang : gorilla/mux

```
func Test_Get(t *testing.T) {
	ts := httptest.NewServer(NewRouter())
	defer ts.Close()

	res, _ := http.Get(ts.URL + "/espresso")

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

func Test_Post(t *testing.T) {
	ts := httptest.NewServer(NewRouter())
	defer ts.Close()

	res, _ := http.Post(ts.URL + "/", "", nil)

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

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

golang : flag

copy home.go in folder chapter14-6 and paste in chapter14-7 :

```
func main() {
    var port string

    flag.StringVar(&port, "port", ":3000",
    “default port: 3000”)
    flag.Parse()

    http.ListenAndServe(port, NewRouter())
}
```

Windows Build:

```
>go build -o main.exe
```

Windows Run:

```
>main.exe -port=:8080
>.\main.exe -port=:8080
```

Linux, Mac Build:

```
>go build -o main
```

Linux, Mac Run:

```
>./main -port=:8080
```

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

golang : middleware

copy home.go in folder chapter14-6 and paste in chapter14-8 :

```go
func HomePageHandle(w http.ResponseWriter, r *http.Request) {
    start := time.Now()
    fmt.Printf("Start at %v", start)

    name := r.URL.Query().Get("name")
    if name == "" {
        name = "World"
    }
    fmt.Fprintf(w, "Hello, %s!", name)
    fmt.Printf("Completed in %v", time.Since(start))
}

func UsersHandle(w http.ResponseWriter, r *http.Request) {
    start := time.Now()
    fmt.Printf("Start at %v", start)

    fmt.Fprintf(w, "Users Page")

    fmt.Printf("Completed in %v", time.Since(start))
}
```

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

golang : middleware

Remove boilerplate with middleware

```go
func loggingMiddleware(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        start := time.Now()
        fmt.Printf("Start at %v", start)

        next.ServeHTTP(w, r)

        fmt.Printf("Completed in %v", time.Since(start))
    })
}

func NewRouter() http.Handler {
    r := mux.NewRouter()
    r.HandleFunc("/", HomePageHandle).Methods("GET")
    r.HandleFunc(“/users", UsersHandle).Methods("GET")
    r.Use(loggingMiddleware)
    return r
}
```

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