Learn the Go Programming Language

For experienced developers or those of an adventurous nature

gotutorial.net
@GoTutorialNet

Matt Nunogawa
@amattn

Lesson 08

Making a Testable HTTP Web Server

v0.2 draft

We're only* going to use the standard library

```
net/http
net/http/httptest
html/template
database/sql

bytes
fmt
io/ioutil
log
strconv
strings
time

github.com/lib/pq/
```

* PostgreSQL driver, not the standard library, but we don't call it directly.

How To Follow Along

```
# This repo is pdfs as well as code, so it’s a bit
large.

git clone https://github.com/amattn/gotutorial.git
cd gotutorial

git checkout tags/b01
git checkout tags/b02
git checkout tags/b03
...

# after any checkout you can:
go test
go build && ./gtls
```

Build 01

- This is just hello world.

NET/HTTP

Routers & Handlers

- At a basic level, we use net/http to route requests to handlers
- The simplest possible way to do is with the built-in pattern matching and HandleFunc

BUILD 02

- HTTP Hello World
- Route using http.HandleFunc()
- Simplest Posible Pattern Matching & One handler
    - “/“
    - inline function literal

Build 03

- Well behaved web servers don’t hard code config
- Let’s implement some simple flag parsing

Reasonable CLI Flags

```
// command line flag variables
var (
    show_h       bool
    show_help    bool
    show_version bool
    listen_addr  string
)
// sensible defaults
const (
    DEFAULT_LISTEN_ADDRESS = ":8080"
)
func init() {
    flag.BoolVar(&show_h, "h", false, "show help message and exit(0)")
    flag.BoolVar(&show_help, "help", false, "show help message and exit(0)")
    flag.BoolVar(&show_version, "version", false, "show version info and exit(0)")
    flag.StringVar(&listen_addr, "addr", DEFAULT_LISTEN_ADDRESS, "addr that our webserver listens on")
}
func main() {
    log.Println("gtls", Version(), "build", BuildNumber())
    flag.Parse()
    if show_version {
        os.Exit(0)
    }
    if show_h || show_help {
        flag.Usage()
        os.Exit(0)
    }

    ...
```

BUILD 04

- Some refactoring
- Route using http.Handle() & a dedicated handler
- Add some primitive logging

BUILD 05

- More refactoring of routing
  - Route using http.ListenAndServe & a dedicated routing struct
  - router calls handlers
  - think about centralizing logging
- and a primitive test script

Handlers & DefaultServeMux

- DefaultServeMux

```
http.ListenAndServe(listen_addr, nil)
```

  - When you pass nil as above, that means you are using DefaultServeMux as your root handler/router.
  - You must use at least one of http.Handle() or http.HandleFunc()

- CustomHandler

```
http.ListenAndServe(listen_addr, logging_handler)
```

  - When you pass your own object, that means your object is the root handler.
  - http.Handle() or http.HandleFunc() are never used

Lots of 3rd Party routers

- gorilla/mux
- https://github.com/bmizerany/pat
- collectivehealth/eprouter

Build 06

- More refactoring of routing
- Simplify child handlers with an interface
- centralize logging in router

Build 07

- Update our router
    - route to Admin or LinksHandler as appropriate
    - modify our interface to support custom response headers
- Make an ultra-primitive “In-Memory DB” to store our short links. (aka map[string]string)
- Make a LinksHandler handle shortlink urls

Build 07

- Update our router
  - route to Admin or LinksHandler as appropriate

```
// command line flag variables
type LoggingRouter struct {
    adminHandler *AdminHandler
    linksHandler *LinksHandler
}

func (router *LoggingRouter) ServeHTTP(w http.ResponseWriter, req *http.Request) {

    <snip>

    if url == "/" {
        code = http.StatusOK
        responseBytes = []byte("Welcome to gtls”)
    } else if strings.HasPrefix(url, "/admin/") {
        // use the admin handler
        code, extra_headers, responseBytes = router.adminHandler.Respond(req)
    } else {
        // use the shortlink handler
        code, extra_headers, responseBytes = router.linksHandler.Respond(req)
    }

    <snip>
```

NET/HTTP/HTTPTEST

Build 08

- Unit Testing!

```
httptest.NewServer(router)
reflect.DeepEqual(expected, candidate)
```

- Currently just checking response status code

Build 08

```
func TestWorkingShortlinks(t *testing.T) {
    working_paths := []string{"a","b","c"}

    for i, subpath := range working_paths {
        final_url := test_server.URL + "/" + subpath
        res, err := http.Get(final_url)
        if err != nil {
            log.Fatal(1626235976, i, subpath, final_url, err)
        }

        assertEqual(t, 200, res.StatusCode, i, final_url)
    }
}
```

currently, only checking status code

Build 09

- New Abstract BaseResponder
- prototype an ugly POST route/handler
  - you would never route it like this in real life

BUILD 10

- Working Post route

- Add a shortlink

```
curl --data "code=123&url=http://google.com" http://
localhost:8080/admin/post
```

- Test a shortlink

```
curl -I http://localhost:8080/123
```

HTML/TEMPLATE

Basic Template Usage

- Make
- Parse (“compile”)
- Execute

Build 11

- Refactor Router to cleanup admin handler
- Refactor admin handler to use html/template after adding a new shortlink

DATABASE/SQL

How to use database/sql

- Find a driver
- connect
- prepare
- exec or query

Build 12

- Use a real database

Further Reading

- http://go-database-sql.org
- https://code.google.com/p/go-wiki/wiki/SQLInterface
- http://golang.org/pkg/database/sql/
- http://gophercon.sourcegraph.com/post/83852708856/building-database-applications-with-database-sql

Build 13

- Slightly more complicated html/template example
- List all shortlinks

`http://localhost:8080/admin/list`

LARGER FRAMEWORKS

- revel
- martini
- beego
- GoCraft
- Echo
- Goji
- Gin

More Reading

- http://golang.org/doc/articles/wiki/
- http://codegangsta.gitbooks.io/building-web-apps-with-go/
- https://www.gitbook.com/book/astaxie/build-web-application-with-golang/details
- https://www.google.com/search?q=web+apps+in+go

Thank You, Credits & License

http://gotutorial.net
@GoTutorialNet

- I owe many many, thanks to the many authors of Go and to Rob Pike in particular.

Matt Nunogawa
@amattn