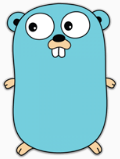
Programming in Go

Matt Holiday
Christmas 2020



Homework

Homework #5

Exercise 7.11 from *GOPL:* web front-end for a database

Except now we will

- Write a program to generate traffic against our first solution
- Show running the server with `-race`
- Solve the race conditions

See my solution at: https://github.com/matt4biz/go-class-exer-7.11

Homework #5

1. Change the DB type

```go
type database struct {
    mu sync.Mutex
    db map[string]int
}
```

2. Use a pointer receiver in all methods

3. Lock the mutex (and defer unlock) in all methods

```go
package main

import ("fmt"; "net/http"; "os"; "testing"; "time")

type sku struct {
    item  string
    price string
}

var items = []sku{
    {"shoes", "46"},
    {"socks", "6"},
    {"sandals", "27"},
    {"clogs", "36"},
    {"pants", "30"},
    {"shorts", "20"},
}
```

```go
func doQuery(cmd, parms string) {
    resp, err := http.Get("http://localhost:8080/" + cmd + "?" + parms)

    if err == nil {
        defer resp.Body.Close()
        fmt.Fprintf(os.Stderr, "got %s = %d (no err)\n", parms, resp.StatusCode)
    } else if resp != nil {
        defer resp.Body.Close()
        fmt.Fprintf(os.Stderr, "got %s = %d (%v)\n", parms, resp.StatusCode, err)
    } else {
        fmt.Fprintf(os.Stderr, "got err %v\n", err)
    }
}
```

```go
func runAdds() {
	for {
		for _, s := range items {
			doQuery("create", "item="+s.item+"&price="+s.price)
		}
	}
}

func runUpdates() {
	for {
		for _, s := range items {
			doQuery("update", "item="+s.item+"&price="+s.price)
		}
	}
}
```

```go
func runDrops() {
    for {
        for _, s := range items {
            doQuery("create", "item="+s.item)
        }
    }
}

func TestServer(t *testing.T) {
    go runServer()    // code from old main
    go runAdds()
    go runDrops()
    go runUpdates()

    time.Sleep(5 * time.Second)
}
```