GO

— for beginner —

acoshift (2017)

WHY GO ?

- Open Source — https://github.com/golang/go
- Fast — Learn, Dev, Compile, Deploy, Run
- Designed for Modern Hardware
- Concurrency (goroutines, channels, select)

Install Go

https://golang.org/

Mac — $ brew install go

Windows — C:\> choco install golang

Linux — 😉

```go
package main

import (
  "fmt"
)

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

```
$ go run main.go
Hello, Gopher.
```

```go
package main

import (
  "log"
)

func main() {
  log.Println("Hello, Gopher.")
}
```

```
$ go run main.go
2017/07/02 20:54:42 Hello, Gopher.
```

POINTER

Address	Value
0x10000000	
0x10000001	
0x10000002	
0x10000003	

POINTER

a = 10

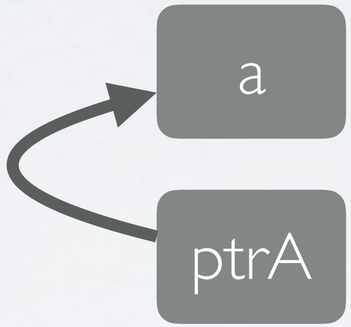
a

Address	Value
0x10000000	10
0x10000001	
0x10000002	
0x10000003	

POINTER

ptrA = &a

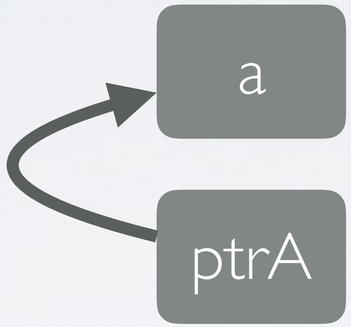
ptrA = 0x10000000



Address	Value
0x10000000	10
0x10000001	0x10000000
0x10000002	
0x10000003	

POINTER

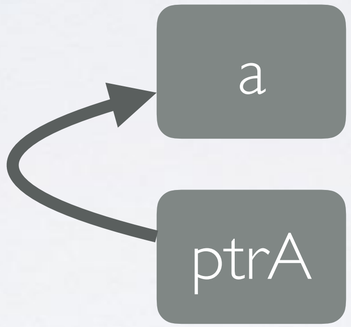
ptrA => 0x10000000
*ptrA => *(0x10000000)
*ptrA => 10



Address	Value
0x10000000	10
0x10000001	0x10000000
0x10000002	
0x10000003	

POINTER

*ptrA = 20



Address	Value
0x10000000	20
0x10000001	0x10000000
0x10000002	
0x10000003	

FUNCTION

```go
package main

import "fmt"

func main() {
  a, b := 1, 2
  r := add(a, b)
  fmt.Println(r)
}

func add(x, y int) int {
  p := 1
  return x + y + p
}
```

main()

a	1
b	2

FUNCTION

```go
package main

import "fmt"

func main() {
  a, b := 1, 2
  r := add(a, b)
  fmt.Println(r)
}

func add(x, y int) int {
  p := 1
  return x + y + p
}
```

```
add(a, b)
```

a	1
b	2
add.return	0
add.p	1
add.x	1
add.y	2

FUNCTION

```go
package main

import "fmt"

func main() {
  a, b := 1, 2
  r := add(a, b)
  fmt.Println(r)
}

func add(x, y int) int {
  p := 1
  return x + y + p
}
```

```
add(a, b)
```

a	1
b	2
add.return	4
add.p	1
add.x	1
add.y	2

FUNCTION

```go
package main

import "fmt"

func main() {
  a, b := 1, 2
  r := add(a, b)
  fmt.Println(r)
}

func add(x, y int) int {
  p := 1
  return x + y + p
}
```

```go
r := add(a, b)
```

a	1
b	2
r	4

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT		
0x01	0x01		
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x01	
0x01	0x01		
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x01	
0x01	0x01	0x02	
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x01	0x06
0x01	0x01	0x02	
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x01	0x06
0x01	0x01	0x02	
0x02	LIT	0x01	
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x01	0x06
0x01	0x01	0x03	
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x04	0x06
0x01	0x01		
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x04	
0x01	0x01		
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

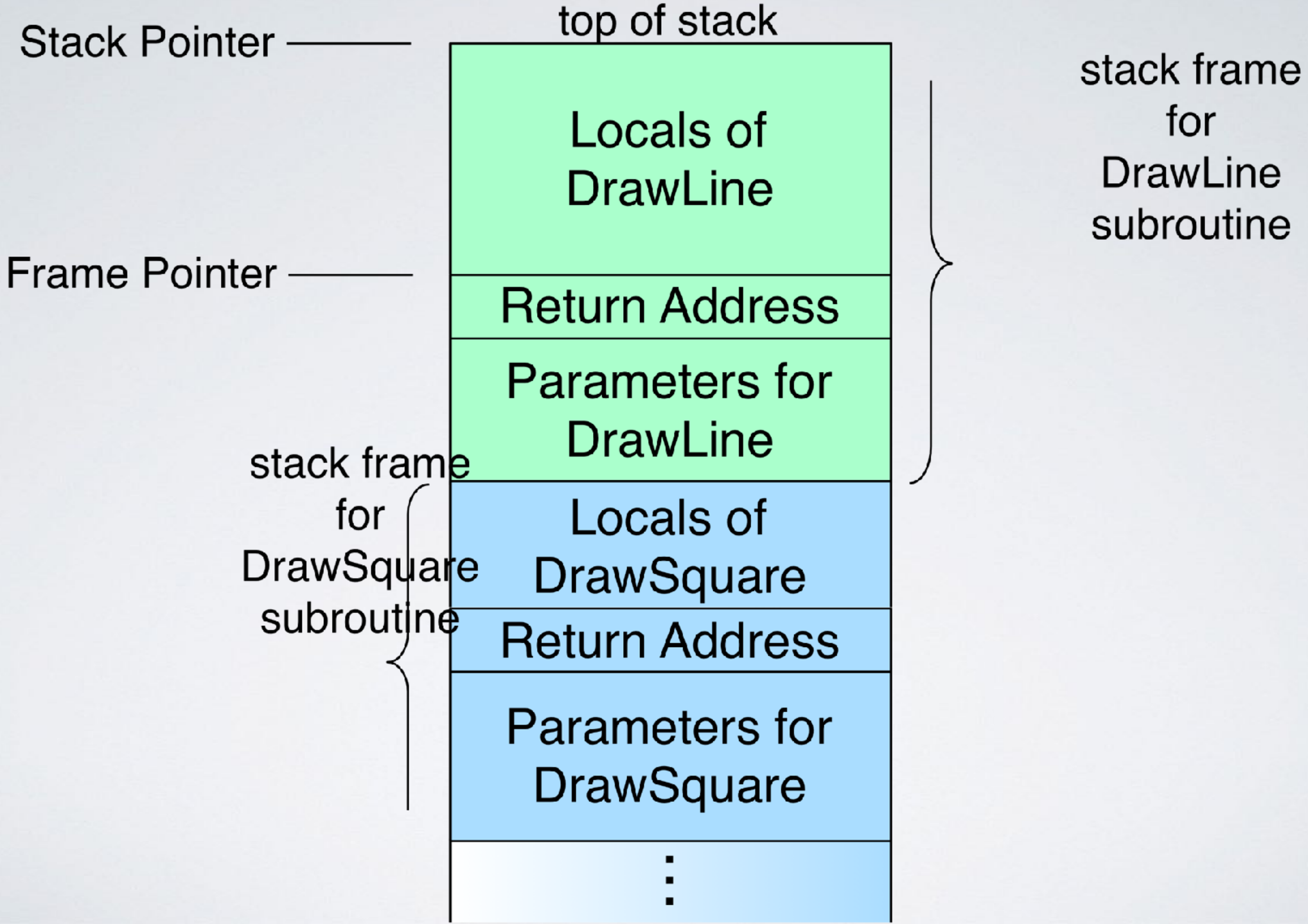
```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT	0x04	
0x01	0x01	0x0F	
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x00		

FUNCTION

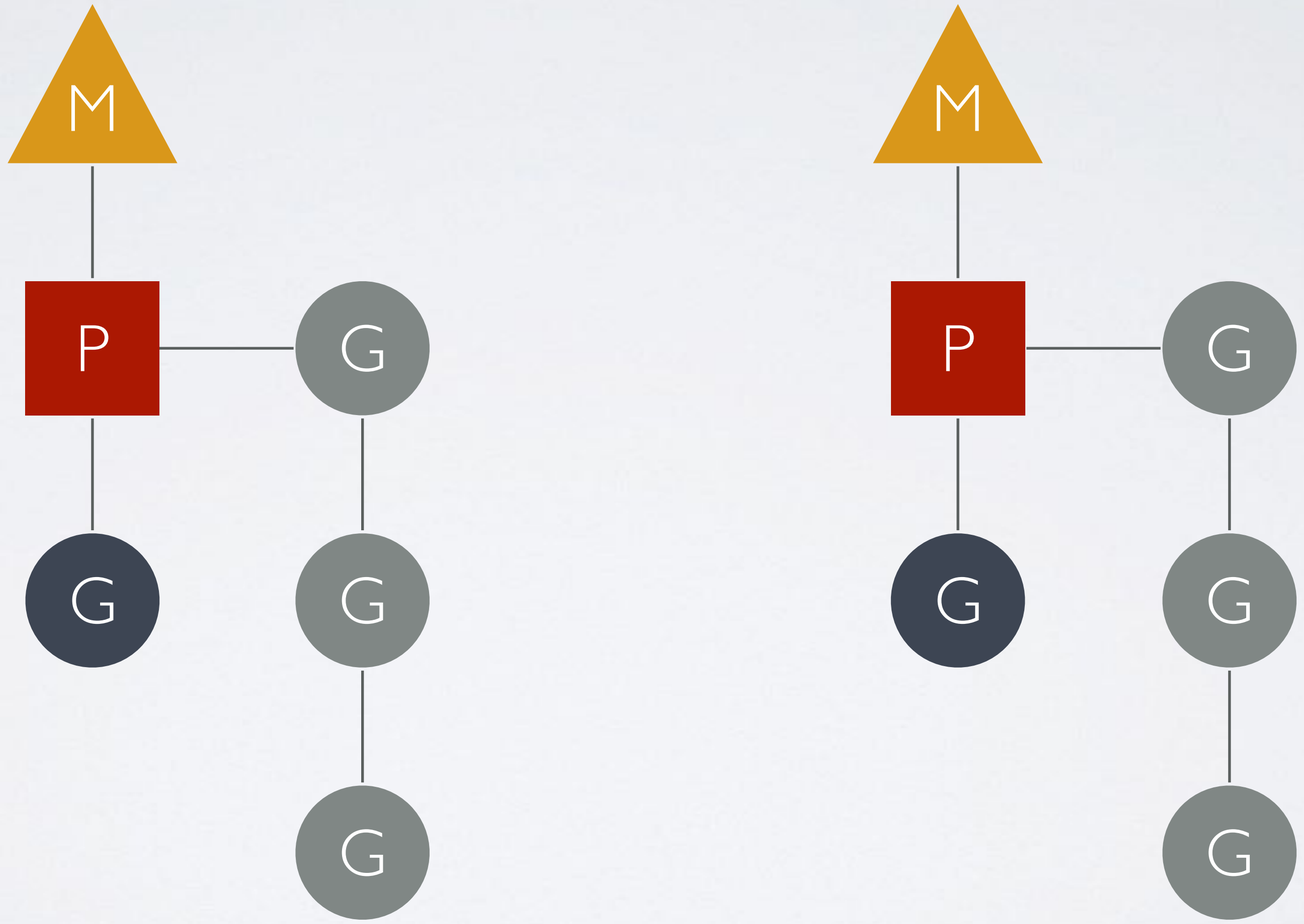
```
LIT 1 ; a
LIT 2 ; b
CALL :ADD
LIT :r
!
HALT
:ADD
LIT 1 ; p
+
+
EXIT
:r
```

0x00	LIT		
0x01	0x01		
0x02	LIT		
0x03	0x02		
0x04	CALL		
0x05	0x0A		
0x06	LIT		
0x07	0x0F		
0x08	!		
0x09	HALT		
0x0A	LIT		
0x0B	0x01		
0x0C	+		
0x0D	+		
0x0E	EXIT		
0x0F	0x04		


top of stack
Stack Pointer
Locals of DrawLine
Frame Pointer
Return Address
Parameters for DrawLine
stack frame for DrawLine subroutine
stack frame for DrawSquare subroutine
Locals of DrawSquare
Return Address
Parameters for DrawSquare

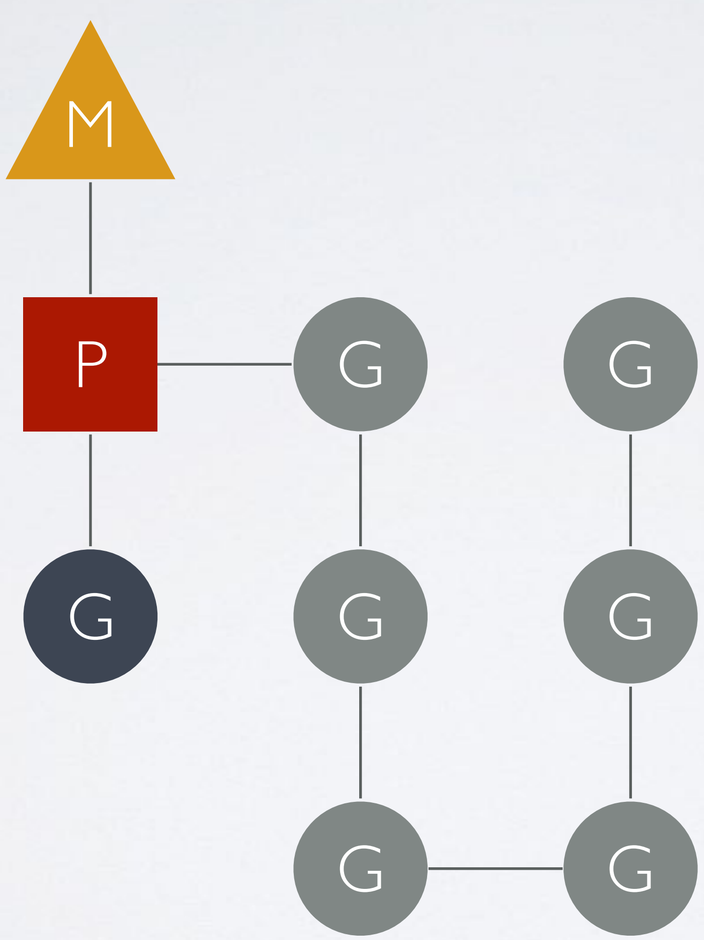
GOROUTINES != THREADS

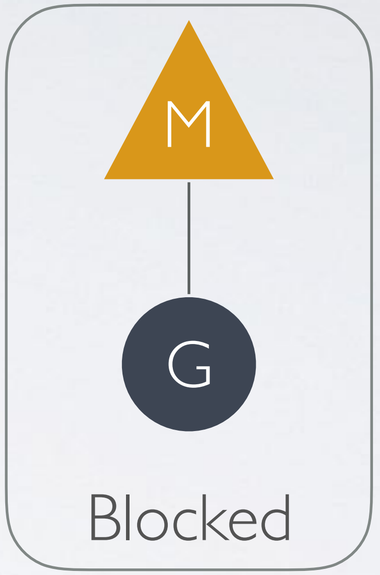
GOMAXPROCS=2



GOROUTINES != THREADS

GOMAXPROCS=2



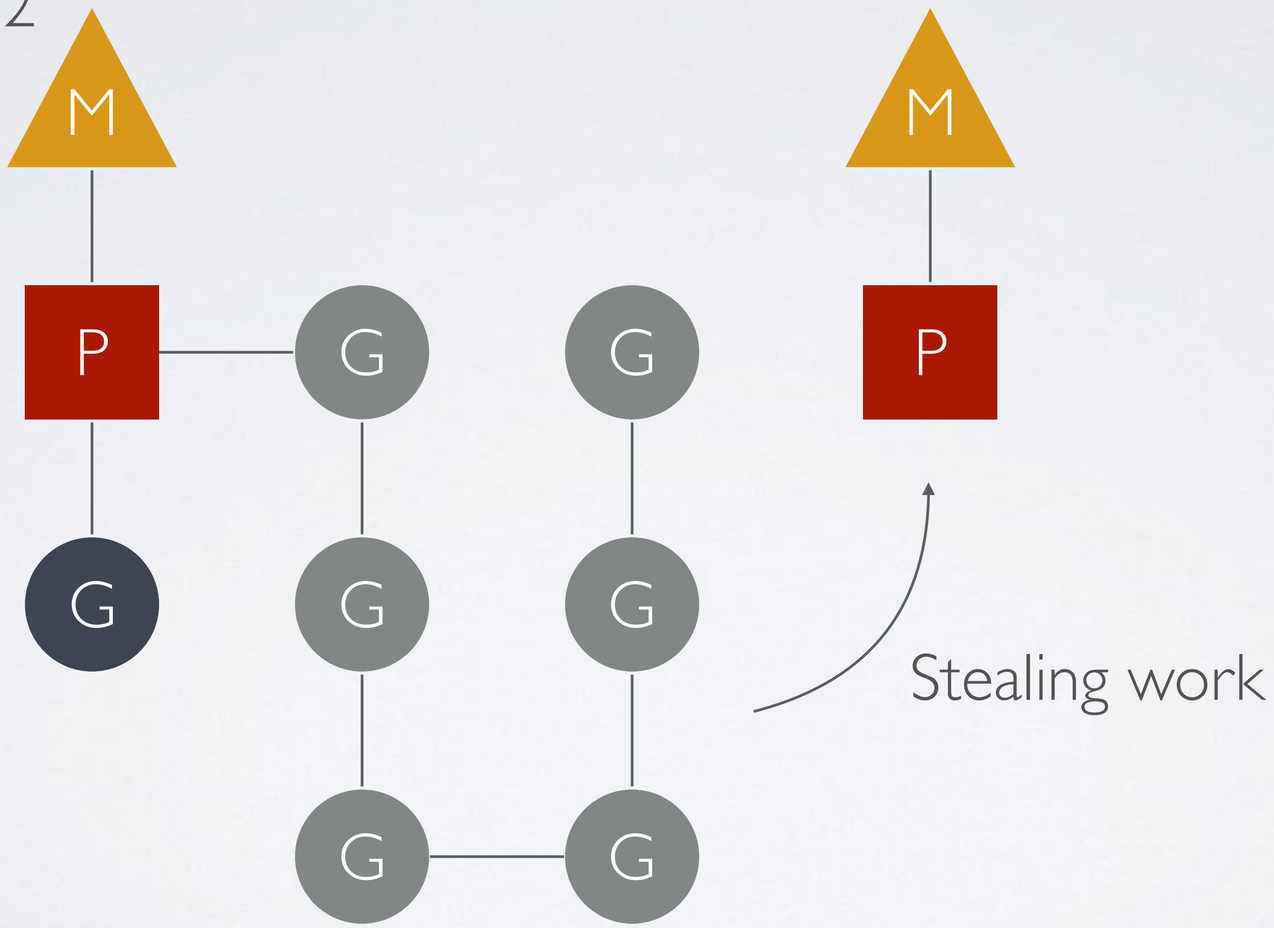


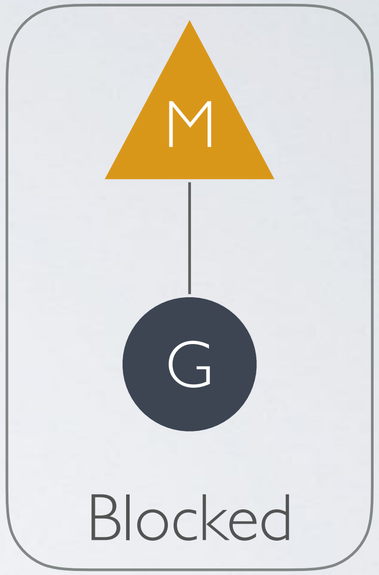
Blocked thread
- syscall (read, write file)

exclude:
- channel operations
- network operations
- sleeping

GOROUTINES != THREADS

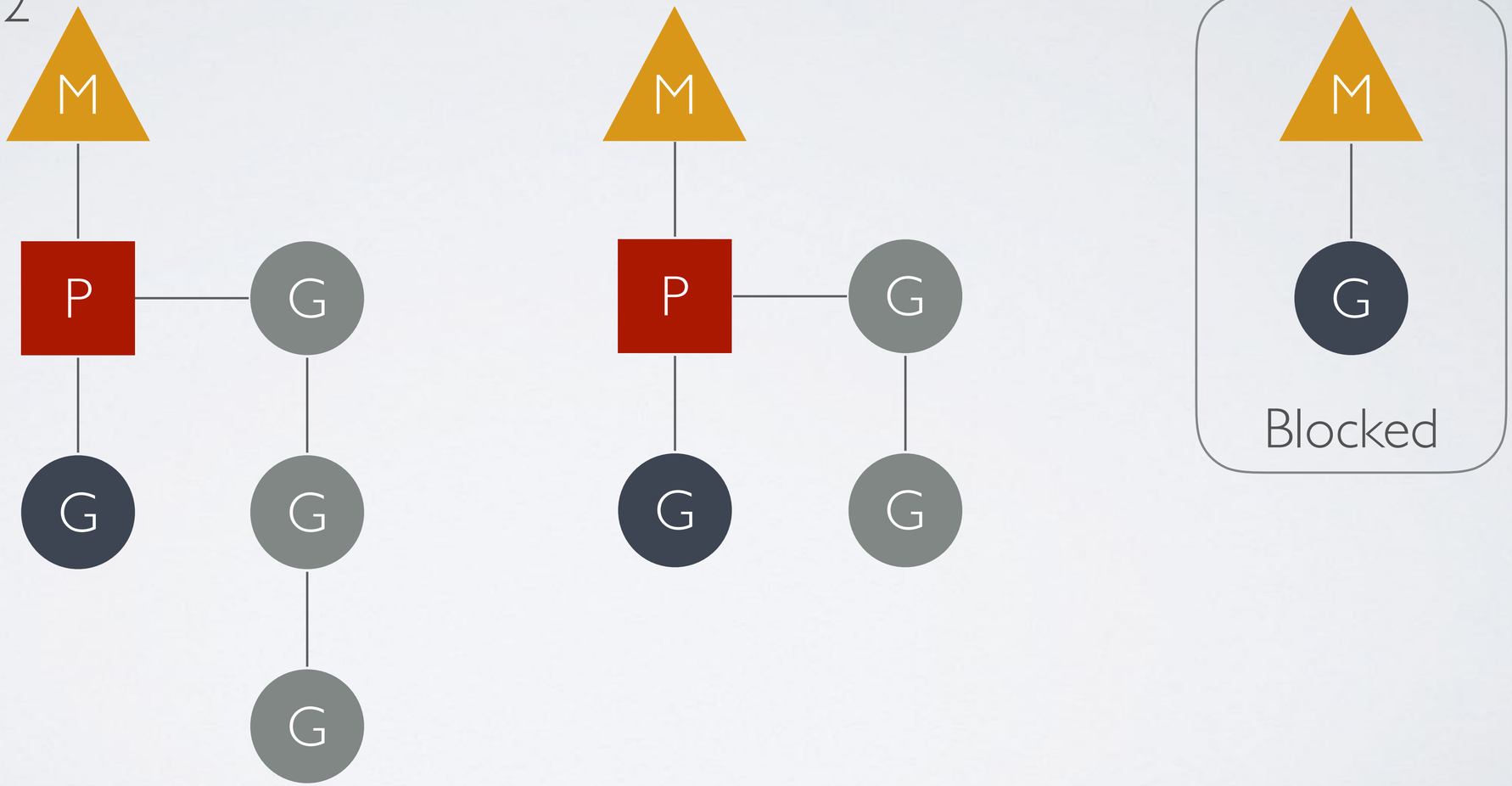
GOMAXPROCS=2



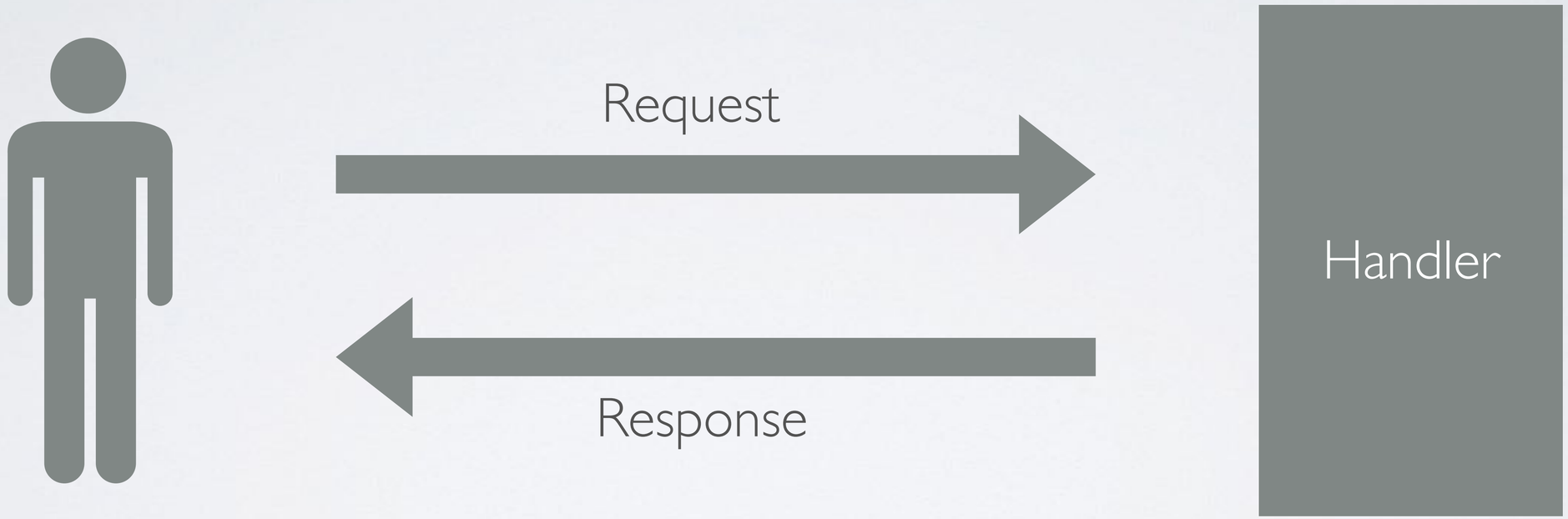


GOROUTINES != THREADS

GOMAXPROCS=2



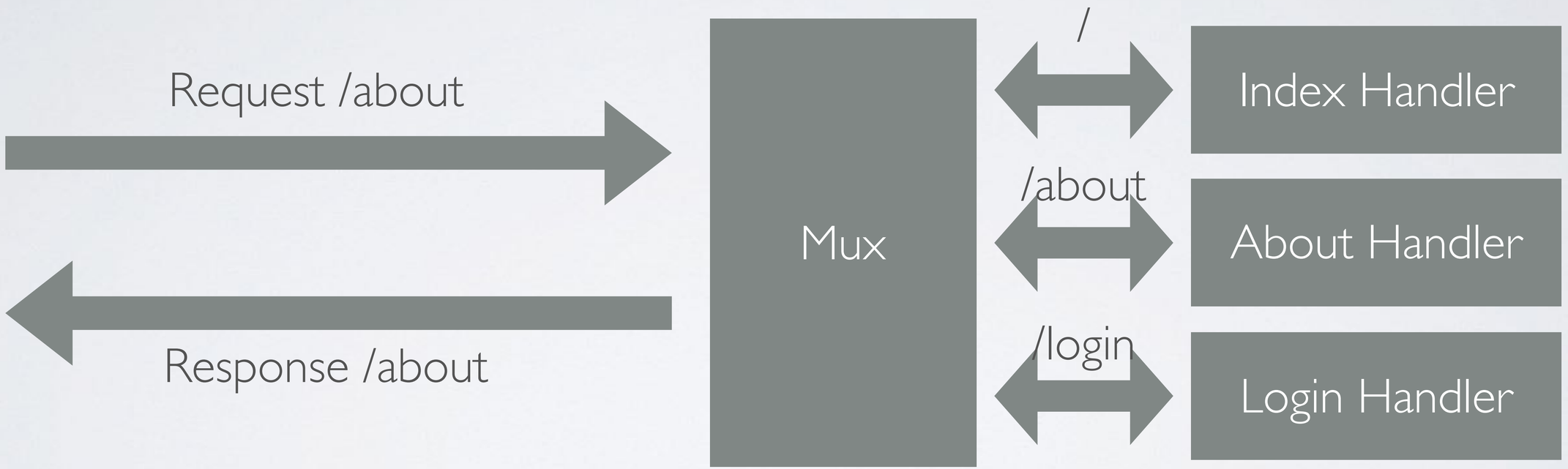
HANDLER



```
type Handler interface {
   ServeHTTP(ResponseWriter, *Request)
}
```

```
type HandlerFunc func(ResponseWriter, *Request)



func (f HandlerFunc) ServeHTTP(w ResponseWriter, r *Request) {
  f(w, r)
}
```

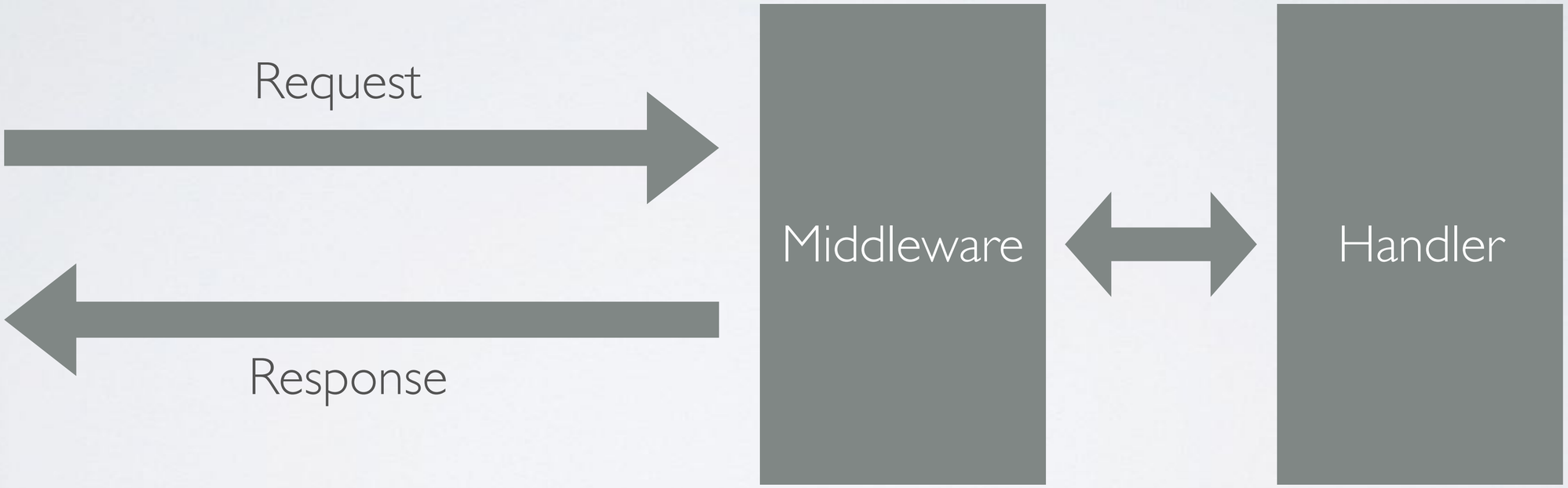
MULTIPLEXER (MUX)



```go
func mux(w http.ResponseWriter, r *http.Request) {
  switch r.URL.Path {
  case "/about":
    aboutHandler(w, r)
  case "/login":
    loginHandler(w, r)
  default:
    indexHandler(w, r)
  }
}

func indexHandler(w http.ResponseWriter, r *http.Request) {
  w.Write([]byte("Index Page"))
}

func aboutHandler(w http.ResponseWriter, r *http.Request) {
  w.Write([]byte("About Page"))
}

func loginHandler(w http.ResponseWriter, r *http.Request) {
  w.Write([]byte("Login Page"))
}
```

```
mux := http.NewServeMux()
mux.HandleFunc("/", indexHandler)
mux.HandleFunc("/about", aboutHandler)
mux.HandleFunc("/login", loginHandler)
```

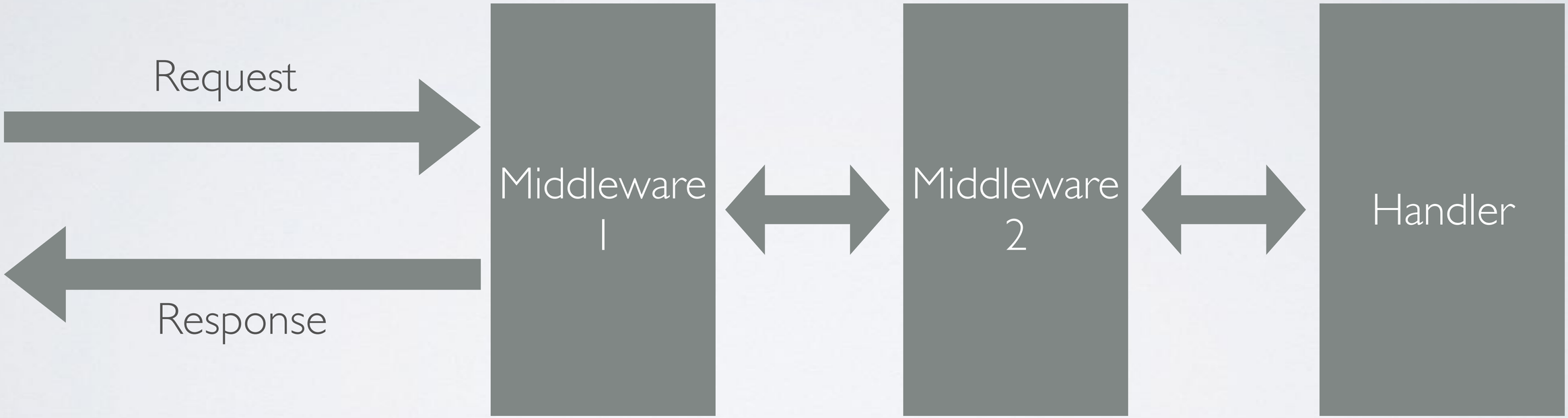
MIDDLEWARE



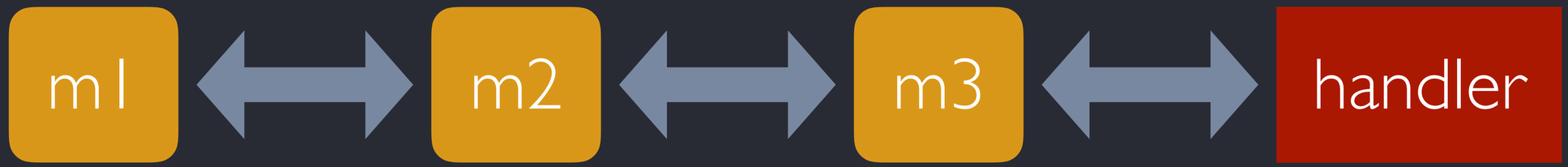
```go
func main() {
  h := logger(http.HandlerFunc(indexHandler))
  http.ListenAndServe(":8080", h)
}

func logger(h http.Handler) http.Handler {
  return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    log.Printf("request: %s\n", r.RequestURI)
    h.ServeHTTP(w, r)
  })
}

func indexHandler(w http.ResponseWriter, r *http.Request) {
  w.Write([]byte("Index Page"))
}
```

```
type Middleware func(http.Handler) http.Handler
```

MIDDLEWARES



```
h := m1(m2(m3(handler)))
```



```go
func Chain(hs ...Middleware) Middleware {
  return func(h http.Handler) http.Handler {
    for i := len(hs); i > 0; i-- {
      h = hs[i-1](h)
    }
    return h
  }
}
```

```
h := Chain(
    m1,
    m2,
    m3,
)(handler)
```

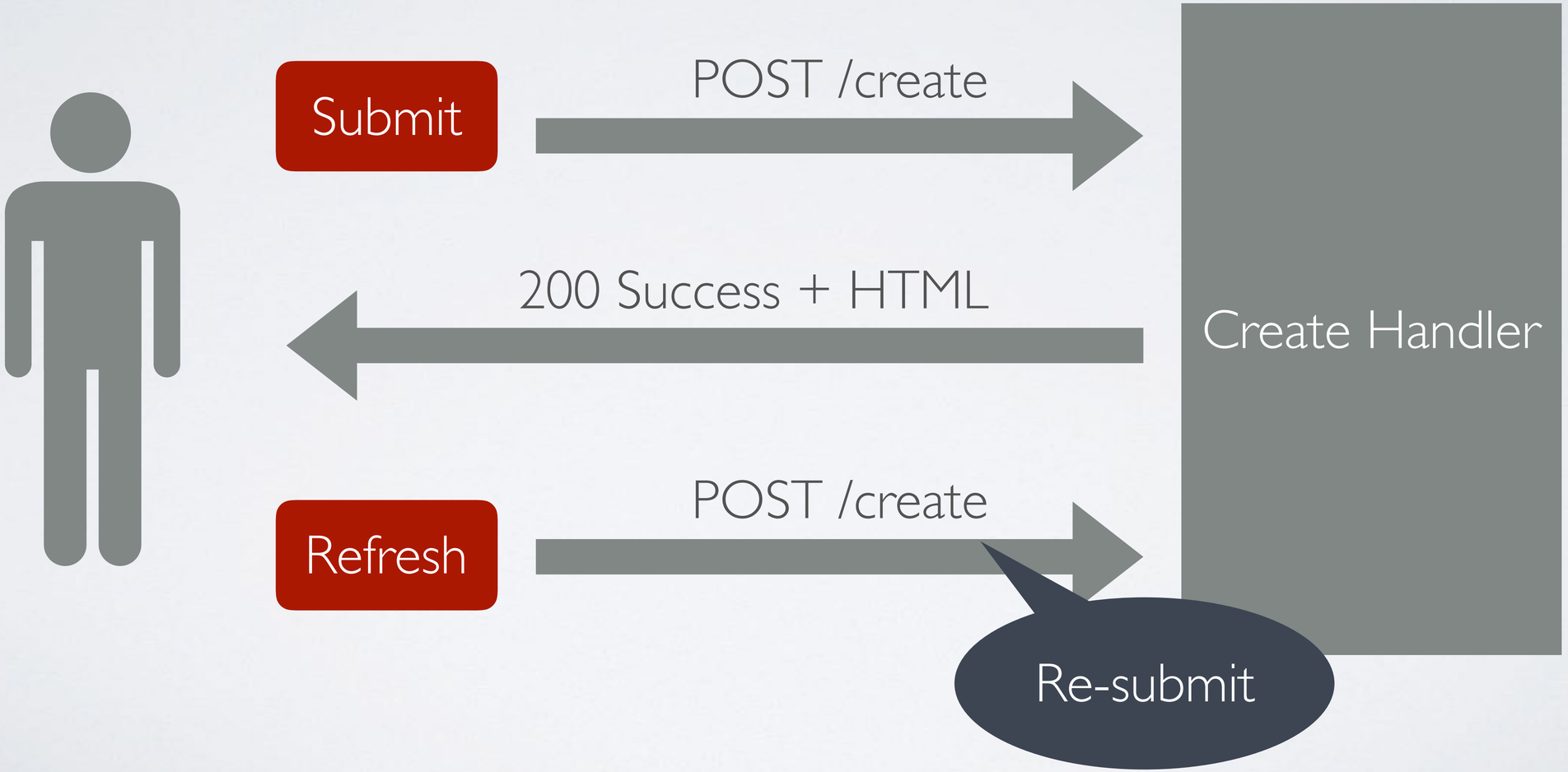
CONFIG MIDDLEWARE

```go
func allowRoleAdmin(h http.Handler) http.Handler {
  return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    reqRole := r.Header.Get("Role")
    if reqRole != "admin" {
      http.Error(w, "Forbidded", http.StatusForbidden)
      return
    }
    h.ServeHTTP(w, r)
  })
}

func allowRoleStaff(h http.Handler) http.Handler {
  return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    reqRole := r.Header.Get("Role")
    if reqRole != "staff" {
      http.Error(w, "Forbidded", http.StatusForbidden)
      return
    }
    h.ServeHTTP(w, r)
  })
}
```

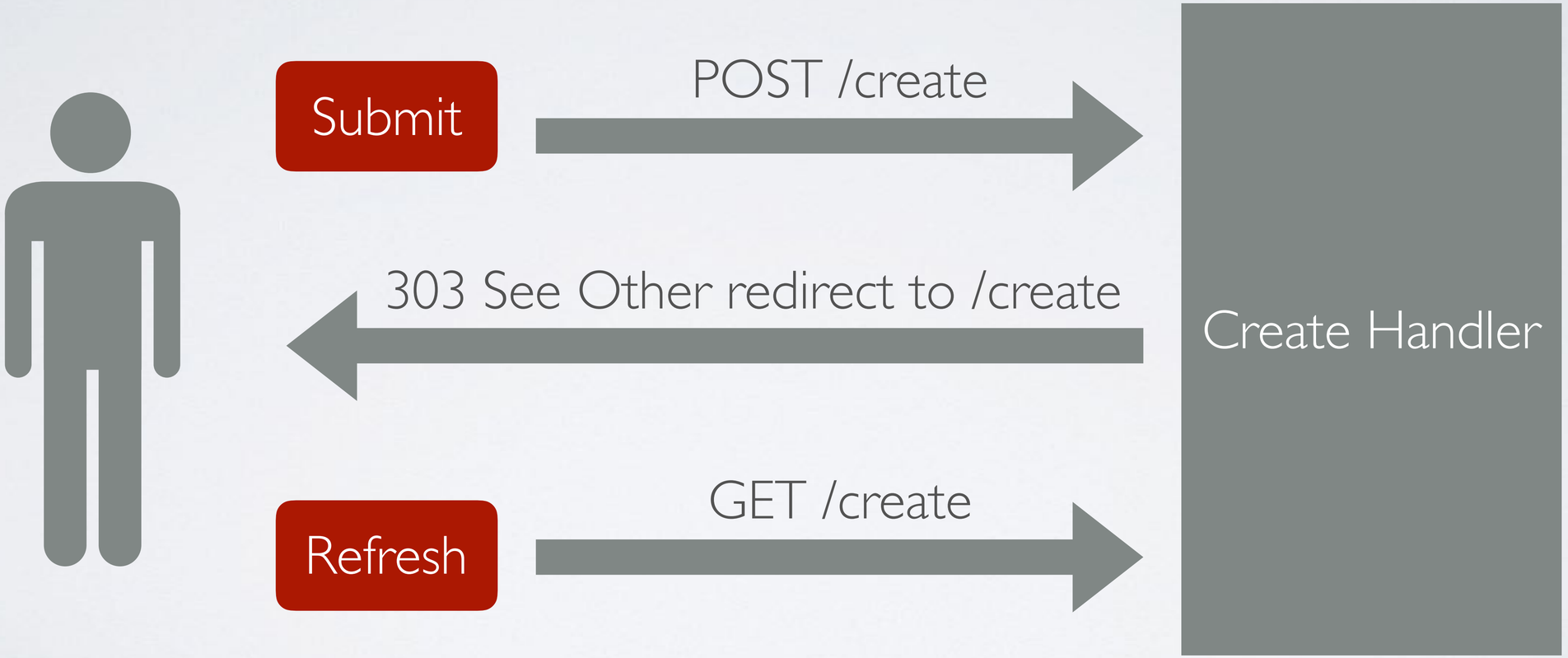
```go
func main() {
  h := allowRole("admin")(handler)
  http.ListenAndServe(":8080", h)
}

func allowRole(role string) Middleware {
  return func(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
      reqRole := r.Header.Get("Role")
      if reqRole != role {
        http.Error(w, "Forbidden", http.StatusForbidden)
        return
      }
      h.ServeHTTP(w, r)
    })
  }
}
```

```go
func allowRoles(roles ...string) Middleware {
  allow := make(map[string]struct{})
  for _, role := range roles {
    allow[role] = struct{}{}
  }

  return func(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
      reqRole := r.Header.Get("Role")
      if _, ok := allow[reqRole]; !ok {
        http.Error(w, "Forbidden", http.StatusForbidden)
        return
      }
      h.ServeHTTP(w, r)
    })
  }
}
```

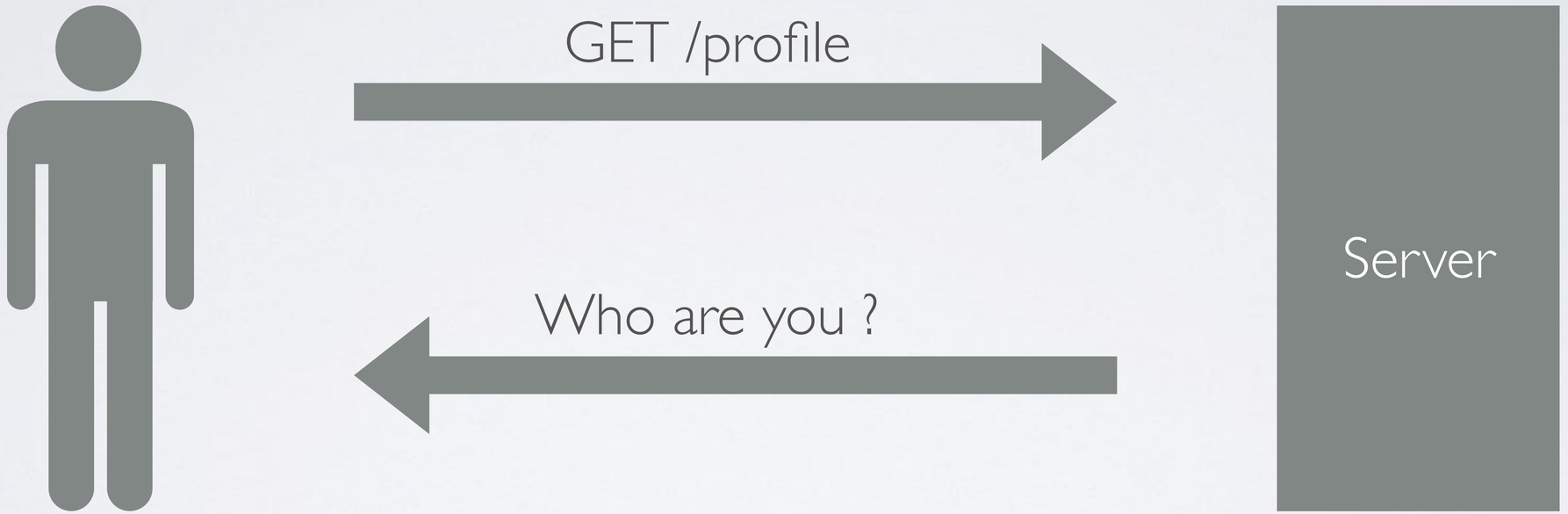
POST-REDIRECT-GET (PRG)



POST-REDIRECT-GET (PRG)

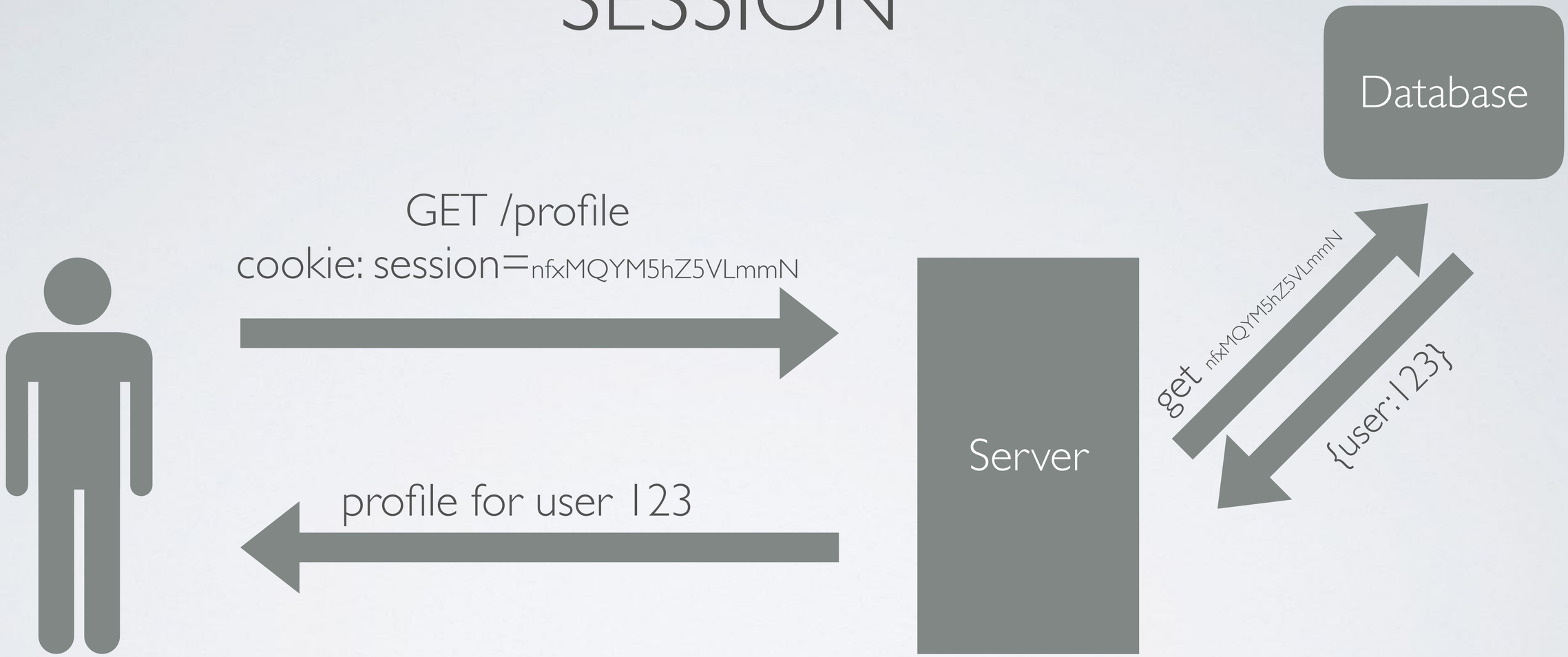


COOKIE

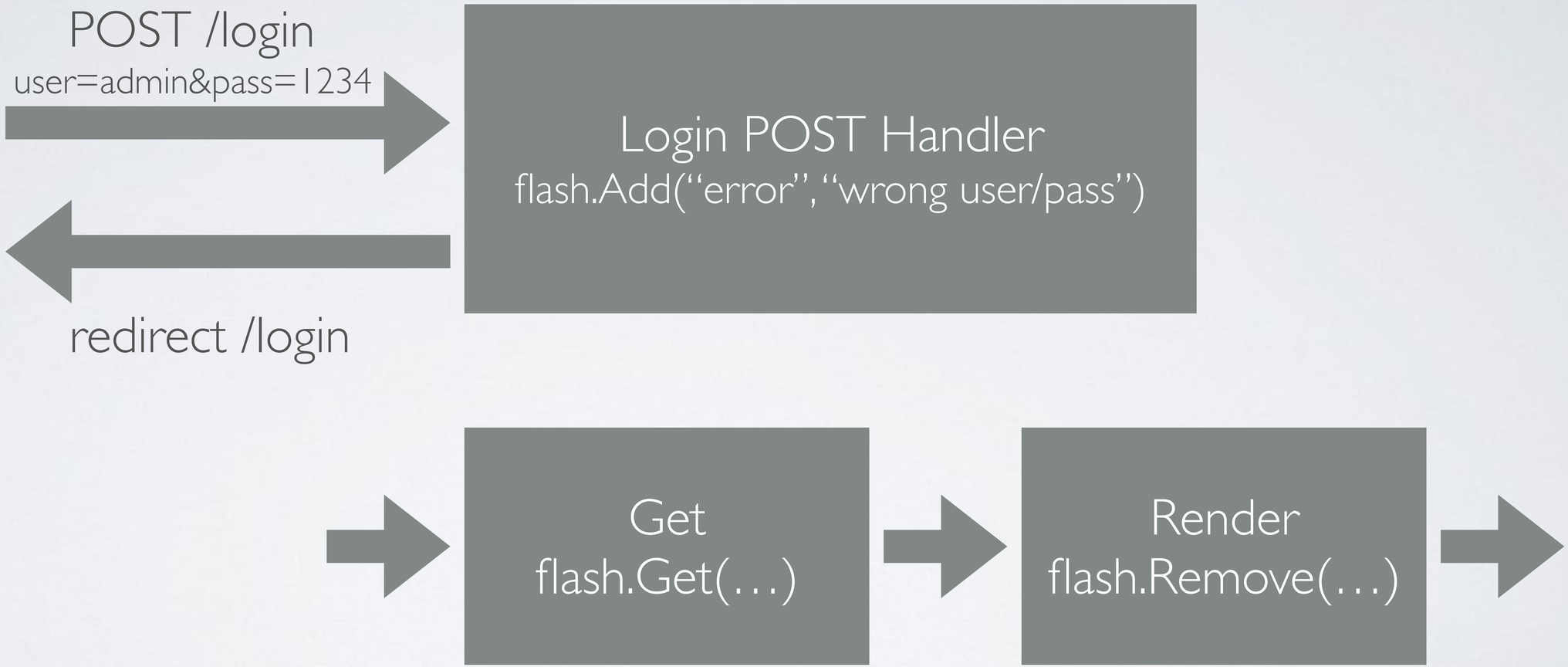


COOKIE

SESSION


Database
GET /profile
cookie: session=nfxMQYM5hZ5VLmmN
Server
get nfxMQYM5hZ5VLmmN
{user:123}
profile for user 123

FLASH MESSAGE



CSRF

- Verifying Same Origin with Standard Headers
  - Origin == `scheme://host:port`
  - Referer starts with `scheme://host:port/`
  - Host == Origin/Referer
- CSRF Token
  - Per-Session Token
  - Per-Request Token
- Double Submit Cookie

not Referrer

tailing slash or use url.Parse

DEPLOYMENT

- Heroku
- Linux

HEROKU

- get port from os.Getenv("PORT")
- heroku create gonews
- heroku buildpacks:set heroku/go
- heroku config:set GOVERSION=go1.8.3
- heroku config:set GO_INSTALL_PACKAGE_SPEC=github.com/acoshift/gonews/cmd/gonews
- git push heroku master

WHAT'S NEXT

- SQL Database
- WaitGroup
- Semaphore
- Context
- Logging
- Email
- CORS
- Web Socket
- SSE
- Caching
- Reverse Proxy
- Image resize/crop
- Testing
- Build tags
- OAuth 2
- OTP
- Reflection
- GRPC
- Microservices
- API Gateway
- Profiling (pprof)
- App Engine
- Docker
- Kubernetes

Q&A