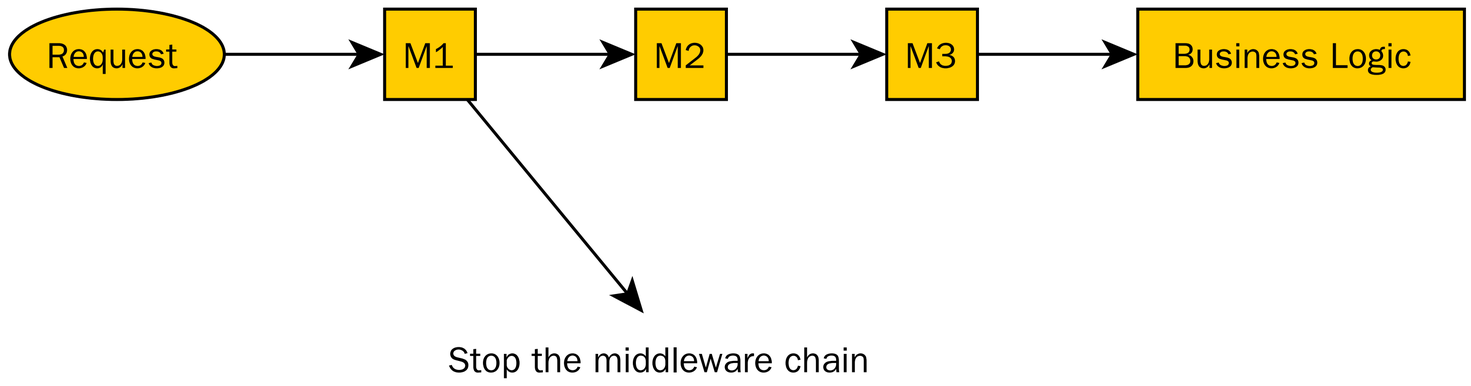
Middleware at Spacelift

Wojciech Barczynski | VPE | Spacelift

Middleware

Provides common services and capabilitie.

Middleware



Middleware in Golang

```
func exampleMiddleware(next http.Handler) http.Handler {
  return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request
    log.Print("Middleware before")
    next.ServeHTTP(w, r)
    log.Print("Middleware after")
  })
}

func main() {
  hello := http.HandlerFunc(func(w http.ResponseWriter, r *http.Reque
    io.WriteString(w, "Hello World!")
  })
  err := http.ListenAndServe(":8080", exampleMiddleware(hello))
  log.Fatal(err)
}
```

Example 1

Middleware in Golang

```go
func main() {
  r := chi.NewRouter()

  // our middleware
  r.Use(middleware.Logger)

  r.Get("/", func(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("welcome"))
  })
  http.ListenAndServe(":3000", r)
}
```

Example 2 / chi docs

A missing piece

How to pass:

- Request-scoped values
- Cancellation signals
- Deadlines
- ... and more

go.dev/blog/context

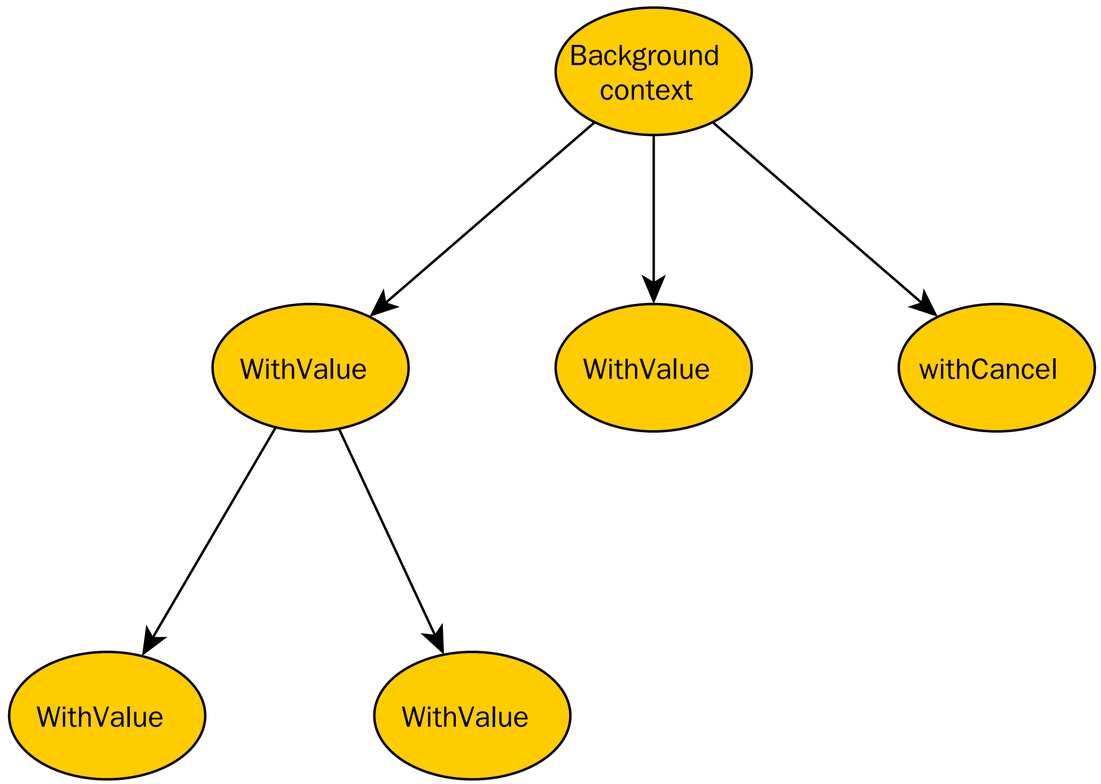
Context

```
type Context interface {
    // Done returns a channel that is closed when this Context is can
    // or times out.
    Done() <-chan struct{}

    // Err indicates why this context was canceled, after the Done ch
    // is closed.
    Err() error

    // Deadline returns the time when this Context will be canceled,
    Deadline() (deadline time.Time, ok bool)

    // Value returns the value associated with key or nil if none.
    Value(key interface{}) interface{}
```

Context

```
ctx = context.WithValue(ctx, "myKey", "myValue")

ctx.Value("myKey")
```

Context


Background
context
WithValue
WithValue
withCancel
WithValue
WithValue

Context

- spcontext is a heart of our middleware

github.com/spacelift-io/spcontext

- fields
- tracing and span management
- logging (bugsnag handler)
- small utilities

github.com/spacelift-io/spcontext

Step 1 - create:

```
ctx := spcontext.New(log.NewJSONLogger(os.Stdout), DatadogTracer)
```

github.com/spacelift-io/spcontext

Step 2 - preparation:

```
r.Use(wrap(spcontext.ContextInjector(ctx)))
```

spcontext.ContextInjector

github.com/spacelift-io/spcontext

Step 3 - how to use it

```
r.Get("/", func(w http.ResponseWriter, r *http.Request) {
        ctx := spcontext.FromStdContext(r.Context())
}
```

github.com/spacelift-io/spcontext

Step 3 - tracing <3

```
ctx, span := ctx.StartSpan(
            spcontext.WithOperation("middleware.account"),
            spcontext.WithResource(subdomain),
            spcontext.WithTags("myapp.account.subdomain", subdomain),

            defer func() { span.Close(err) }()
```

spcontext.StartSpan

github.com/spacelift-io/spcontext

Step 3 - pushing along the middleware chain

```
func exampleMiddleware(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Reque

        next(w, r.WithContext(ctx))

    })
}
```

Summary

- Give spcontext a shot.
- You might find the code patterns useful for your project.

ps. We are hiring – spacelift.io/careers

Questions?