

Error Handling in Go

Holistic view, based on personal experience

Vitalii Lakusta
Starship Technologies
lakusta96@gmail.com
June 29, 2020

Recommended Articles on Go Error Handling

1. <https://blog.golang.org/error-handling-and-go>
2. <https://blog.golang.org/errors-are-values>
3. <https://blog.golang.org/go1.13-errors>
4. <https://blog.golang.org/defer-panic-and-recover>
5. https://golang.org/doc/faq#nil_error

Why do we need error handling?

- Invoke other biz logic when error happens. Gracefully handle errors.
- Discover errors before your users report to you!
- Give correct feedback on the error to the user/client (e.g. 404 Not Found instead of 500 Internal Server Error).
- Debug effectively. Get to the root cause of error asap.
- Get vital statistics of errors fast.
 - When was the error first seen?
 - Last seen?
 - What is its frequency?
 - How many users does it affect?
 - In which release/build this error first appeared?
 - Does it happen in sandbox or production?

Tools related to error handling

- **Logs** (stdout, stderr).
- Error tracking software (Sentry, Rollbar)
- Slack (or other comms tool you use)
 - When error happens, you see it in Slack immediately
- **Distributed Tracing**
 - To get the full chain of microservice direct calls & side-effects related to this error.
 - Add traceID to errors & logs.
 - Keywords: Jaeger tracing, OpenTracing, OpenTelemetry
- **Metrics Monitoring** (Prometheus, InfluxDB, Grafana)
 - To see in Grafana immediately health of the service, rate of errors.
 - Alerting rules for errors.

Discover Fast - Example of Error Stats in Sentry

All Environments

LAST 24 HOURS



LAST 30 DAYS



FIRST SEEN

When: 3 months ago

Apr 6, 2020 5:17:35 PM EEST

Release: [bb4d0070d871](#)

LAST SEEN

When: 7 hours ago

Jun 29, 2020 11:06:15 AM EEST

Release: [a43655594bc1](#)

Linked Issues



[Link GitHub Issue](#)



Tags

environment [sandbox](#) 95%



sentry APP 10:29 AM

MongoError

interrupted at shutdown

environment

sandbox



SERVER-SERVICE-MERCHANT-A via Merchant Repo Errors - Slack

#merchant-errors> | Today at 10:28 AM

Resolve...

Ignore

Select Assignee...



Considerations when handling errors

- Logs formatting. JSON is a good start. Check out zap log library <https://github.com/uber-go/zap> for json-formatted logs.
- Make sure your logs are searchable fast. Make sure you can add structured key-value info to logs, and you can search by them fast.
- Which layer should log & report the error?
 - Rule of thumb - aim for service layer, as it's the place which fulfills use-cases, and can be accessed via different interface (API, CLI, cron job etc.)
- Propagating trace ids into logs.
- Error noise. What to do when there are too many errors in the service?
- Error logging/reporting duplication. Same error, reported 3 or 4 times.

Errors in Go. TL;DR

```
type error interface {  
    Error() string  
}
```

Errors in Go. TL;DR

`// errorString is a trivial implementation of error.`

```
type errorString struct {
```

```
    s string
```

```
}
```

```
func (e *errorString) Error() string {
```

```
    return e.s
```

```
}
```


Errors in Go. TL;DR

// New returns an error that formats as the given text.

```
func New(text string) error {  
    return &errorString{text}  
}
```

Errors in Go. TL;DR

```
func Sqrt(f float64) (float64, error) {  
    if f < 0 {  
        return 0, errors.New("math: sqrt negative number")  
    }  
  
    // implementation  
}
```

Errors in Go. TL;DR

```
f, err := Sqrt(-1)

if err != nil {
    fmt.Println(err)

    // handle error here.

    // 1. Errors are values. No panic, 99.999% of the time.
    // 2. Errors flow in Go is understandable and boring...by design!
    // 3. DBC (design by contract)! Make it clear to the client
    //     what kind of errors it can expect from you.
}
```

Errors in Go. TL;DR

```
if f < 0 {  
    return 0, fmt.Errorf("math: square root of negative number  
                           %g", f)  
}
```

Errors in Go. TL;DR

```
type NegativeSqrtError float64
```

```
func (f NegativeSqrtError) Error() string {  
    return fmt.Sprintf("math: square root of negative number %g",  
        float64(f))  
}
```

Errors in Go. TL;DR

```
type SyntaxError struct {  
    msg      string // description of error  
    Offset int64  // error occurred after reading Offset bytes  
}
```

```
func (e *SyntaxError) Error() string { return e.msg }
```

Errors in Go. TL;DR

```
if err := dec.Decode(&val); err != nil {  
    if serr, ok := err.(*json.SyntaxError); ok {  
        line, col := findLine(f, serr.Offset)  
        return fmt.Errorf("%s:%d:%d: %v", f.Name(), line, col,  
            err)  
    }  
    return err  
}
```

Errors in Go. TL;DR

```
package net
```

```
type Error interface {
```

```
    error
```

```
    Timeout() bool    // Is the error a timeout?
```

```
    Temporary() bool // Is the error temporary?
```

```
}
```


Errors in Go. TL;DR

```
if nerr, ok := err.(net.Error); ok && nerr.Temporary() {  
    time.Sleep(1e9)  
    continue  
}  
  
if err != nil {  
    log.Fatal(err)  
}
```

Errors in Go. TL;DR

```
type appError struct {  
    Error    error  
    Message  string  
    Code     int  
}
```

```
type appHandler func(http.ResponseWriter, *http.Request) *appError
```

```
func (fn appHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {  
    if e := fn(w, r); e != nil { // e is *appError, not os.Error.  
        c := appengine.NewContext(r)  
        c.Errorf("%v", e.Error)  
        http.Error(w, e.Message, e.Code)  
    }  
}
```

Errors in Go. TL;DR

```
func viewRecord(w http.ResponseWriter, r *http.Request) *appError {
    c := appengine.NewContext(r)
    key := datastore.NewKey(c, "Record", r.FormValue("id"), 0, nil)
    record := new(Record)
    if err := datastore.Get(c, key, record); err != nil {
        return &appError{err, "Record not found", 404}
    }
    if err := viewTemplate.Execute(w, record); err != nil {
        return &appError{err, "Can't display record", 500}
    }
    return nil
}
```

```
func init() {
    http.Handle("/view", appHandler(viewRecord))
}
```

Error Logging/Reporting Duplication

```
func someFunction() error {  
    if err != nil {  
        // log error  
        // return err  
    }  
}
```

```
func someOtherFunction() error {  
    val, err := someFunction()  
    if err != nil {  
        // log error  
        // return err  
    }  
}
```

Where to Handler Errors

▶ bootstrap

▶ domain

▼ infra

▶ kafka

▶ mongo

▶ rehttp

▶ server

▶ mocks

▶ service

Wrapping Error Logging into Interface

```
type CommonLogger interface {  
    Debug(...interface{})  
    Info(...interface{})  
    Warn(...interface{})  
    Error(...interface{})  
    Panic(...interface{})  
    Fatal(...interface{})  
}
```

```
type CommonLoggerFmt interface {  
    CommonLogger  
  
    Debugf(template string, args ...interface{})  
    Infof(template string, args ...interface{})  
    Warnf(template string, args ...interface{})  
    Errorf(template string, args ...interface{})  
    Panicf(template string, args ...interface{})  
    Fatalf(template string, args ...interface{})  
}
```

Wrapping Error Logging into Interface

```
type Logger interface {  
    logger.CommonLoggerFmt  
    With(args ...interface{}) Logger  
}
```

```
if err != nil {  
    s.With("span", utils.SpanID(ctx)).Error(err)  
    return nil, err  
}
```

Passing CrossCutting as dep in your service layer

```
type CrossCutting interface {  
    Logger  
    CollectMetric(name string, value float64, tags ...string)  
    CurrentTime() time.Time  
}
```

- + Easy to mock in tests
- + Easy to swap any other logger later on (and swap simpler logger in tests as well)

Embedding Build into Binary

```
ARG GIT_COMMIT
ARG BUILD_NUMBER
RUN go build -ldflags="-X 'main.CommitHash=${GIT_COMMIT}' -X 'main.BuildNumber=${BUILD_NUMBER}'" -o orderthrottling-bin -i cmd/orderthrottling/main.go
```

```
// These variables are baked in via ldflags during binary build as:
// -ldflags="-X 'main.CommitHash=41aa0cb4b3781da46' -X 'main.BuildNumber=45'"
var CommitHash = "development"
var BuildNumber = "development"

func main() {
    log.Println("CommitHash ", CommitHash)
    log.Println("BuildNumber ", BuildNumber)
    app := bootstrap.InitApplication(CommitHash, BuildNumber)

    if err := app.Start(); err != nil {
        app.Error(err)
    }

    app.Cleanup()
}
```

Working with Errors Before Go 1.13

Sometimes we compare an error to a known *sentinel* value, to see if a specific error has occurred.

```
var ErrNotFound = errors.New("not found")

if err == ErrNotFound {
    // something wasn't found
}
```

Working with Errors Before Go 1.13

```
if err != nil {  
    return fmt.Errorf("decompress %v: %v", name, err)  
}
```

```
type QueryError struct {  
    Query string  
    Err   error  
}
```

```
if e, ok := err.(*QueryError); ok && e.Err == ErrPermission {  
    // query failed because of a permission problem  
}
```

Working with Errors in Go 1.13

```
func (e *QueryError) Unwrap() error { return e.Err }
```

```
// Similar to:  
//   if err == ErrNotFound { ... }  
if errors.Is(err, ErrNotFound) {  
    // something wasn't found  
}
```

```
// Similar to:  
//   if e, ok := err.(*QueryError); ok { ... }  
var e *QueryError  
if errors.As(err, &e) {  
    // err is a *QueryError, and e is set to the error's value  
}
```

Working with Errors in Go 1.13

```
if e, ok := err.(*QueryError); ok && e.Err == ErrPermission {  
    // query failed because of a permission problem  
}
```

AFTER GO 1.13

```
if errors.Is(err, ErrPermission) {  
    // err, or some error that it wraps, is a permission problem  
}
```

Working with Errors in Go 1.13

```
if err != nil {  
    return fmt.Errorf("decompress %v: %v", name, err)  
}
```

AFTER GO 1.13

```
if err != nil {  
    // Return an error which unwraps to err.  
    return fmt.Errorf("decompress %v: %w", name, err)  
}
```

```
err := fmt.Errorf("access denied: %w", ErrPermission)  
...  
if errors.Is(err, ErrPermission) ...
```

Thank you!

Q & A

Vitalii Lakusta
Starship Technologies
lakusta96@gmail.com
June 29, 2020