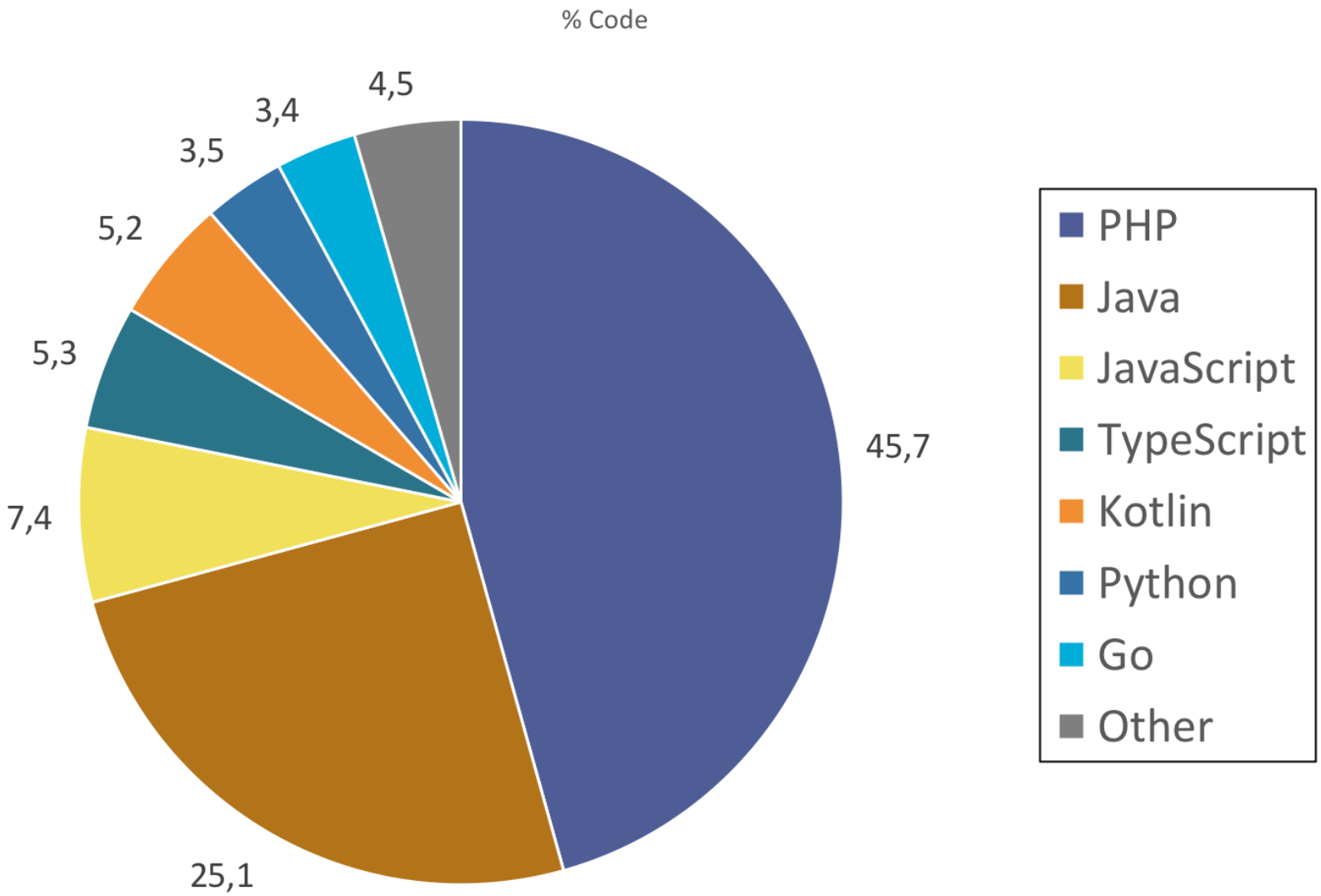
Go at trivago

27 January 2020

Martin Mai
Site Reliability Engineer
trivago N.V.

Slides

github.com/mrkanister/slides (https://github.com/mrkanister/slides)


% Code
4,5
3,4
3,5
5,2
5,3
7,4
25,1
45,7
PHP
Java
JavaScript
TypeScript
Kotlin
Python
Go
Other

Error Handling

```
9  func main() {
10      file, err := os.Open("some-file")
11      if err != nil {
12          log.Fatal()
13      }
14      defer file.Close()
15
16      var data []int
17      if err := json.NewDecoder(file).Decode(&data); err != nil {
18          log.Fatal(err)
19      }
20
21      // ...
22  }
```

Error Handling

```
9  func main() {
10     file, err := os.Open("some-file")
11     if err != nil {
12         log.Fatal()
13     }
14     defer file.Close()
15
16     var data []int
17     if err := json.NewDecoder(file).Decode(&data); err != nil {
18         log.Fatal(err)
19     }
20
21     // ...
22 }
```

Wait, are our deferred functions run when we call `log.Fatal()`?

Error Handling

```
package log

// Fatal is equivalent to Print() followed by a call to os.Exit(1).
func Fatal(v ...interface{}) {
    std.Output(2, fmt.Sprint(v...))
    os.Exit(1)
}
```

```
package os

// Exit causes the current program to exit with the given status code.
// Conventionally, code zero indicates success, non-zero an error.
// The program terminates immediately; deferred functions are not run.
//
// For portability, the status code should be in the range [0, 125].
func Exit(code int)
```

Not exactly the answer we were looking for ... what would you do?

Error Handling

```
10  func main() {
11      if err := run(); err != nil {
12          log.Fatal(err)
13      }
14  }
15
16  func run() error {
17      file, err := os.Open("some-file")
18      if err != nil {
19          return err
20      }
21      defer file.Close()
22
23      var data []int
24      if err := json.NewDecoder(file).Decode(&data); err != nil {
25          return fmt.Errorf("decode data: %v", err)
26      }
27
28      // ...
29      return nil
30  }
```

Versioning Injection

- version flag

```
$ git --version
git version 2.25.0
```

- version command

```
$ go version
go version go1.13.6 darwin/amd64
```

- version command with flag

```
$ kubectl version --short
Client Version: v1.14.8
Server Version: v1.15.4-gke.22
```

So many options to display, but which version to use?

Versioning Injection

Maybe Git has an answer.

```
$ git describe --help
--always
    Show uniquely abbreviated commit object as fallback.

--dirty
    If the working tree has local modification "-dirty" is appended.

--tags
    Use any tag found in refs/tags namespace.
```

Let's combine them:

```
$ git describe --always --dirty --tags
v0.1.0-91-g8a2d63d
```

Nice! But how do we inject the version when running our application?

Versioning Injection

From `run`

```
$ go help run
For more about build flags, see 'go help build'.
```

over `build`

```
$ go help build
-ldflags '[pattern=]arg list'
    arguments to pass on each go tool link invocation.
```

to golang.org/cmd/link (https://golang.org/cmd/link)

```
Flags:
-X importpath.name=value
    Set the value of the string variable in importpath named `name` to `value`.
```

I think we found it!

Version Injection

Let's try it:

```
1  package main
2
3  import "fmt"
4
5  var VersionString string
6
7  func main() {
8      fmt.Println(VersionString)
9  }
```

```
$ go run .

$ go run -ldflags "-X main.VersionString=v0.1.0" .
v0.1.0

$ go run -ldflags "-X main.VersionString=$(git describe --always --dirty --tags)" .
v0.1.0-91-g8a2d63d
```

Voila!

Version Injection

```
9  var VersionString string
10
11  func main() {
12      mux := http.NewServeMux()
13      // setup routes ...
14
15      mux.HandleFunc("/version", func(w http.ResponseWriter, r *http.Request) {
16          fmt.Fprintln(w, VersionString)
17      })
18
19      // define timeouts in production!
20      log.Fatal(http.ListenAndServe(":8080", mux))
21  }
```

```
$ go run -ldflags "-X main.VersionString=$(git describe --always --dirty --tags)" . &
[1] 87782

$ curl localhost:8080/version
v0.1.0-91-g8a2d63d
```

Project Structure

```
├── bin/
│   └── staticcheck
├── build/
│   ├── mdclient
│   └── mdserver
├── cmd/
│   ├── mdclient/
│   └── mdserver/
├── doc/
├── k8s/
├── pkg/
│   ├── domain/
│   ├── gateway/
│   ├── handler/
│   └── storage/
├── Dockerfile
├── Makefile
├── README.md
├── RUNBOOK.md
├── go.mod
├── go.sum
└── tools.go
```

Project Structure

```
├── cmd/            # main packages
│   ├── mdclient/
│   └── mdserver/
├── doc/            # documentation
├── pkg/            # libraries
│   ├── domain/
│   ├── gateway/
│   ├── handler/
│   └── storage/
├── README.md
├── go.mod          # dependencies
└── go.sum
```

- Basic layout, no magic
- GopherCon 2018: How Do You Structure Your Go Apps (https://www.youtube.com/watch?v=oL6JBUk6tj0)
- GopherCon 2019: How Uber Goes (https://www.youtube.com/watch?v=nLskCRJOdxM)
- Uncle Bob: The Clean Architecture (https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html)

Project Structure

```
├── build/          # build artifacts (.gitignore)
│   ├── mdclient
│   └── mdserver
└── Makefile
```

```
VERSION := $(shell git describe --always --dirty --tags)
LDFLAGS := -ldflags '-X main.VersionString=$(VERSION)'

build-%:
    go build $(LDFLAGS) -o build/$* ./cmd/$*

# Enable auto-completion for all available main packages.
COMMANDS := $(patsubst cmd/%/.,%,$(wildcard cmd/*/.))
$(addprefix build-,$(COMMANDS)):
```

```
$ make build-  <tab><tab>
build-%    build-mdclient    build-mdserver
$ make build-
```

Project Structure

```
├── bin/           # tool dependencies (.gitignore)
│   └── staticcheck
├── Makefile
└── tools.go       # versioning
```

```
tools: export GOBIN := $(PWD)/bin
tools:
    go install honnef.co/go/tools/cmd/staticcheck

check: export PATH := $(PWD)/bin:$(PATH)
check:
    staticcheck ./...
```

```
// +build tools

package main

import (
    _ "honnef.co/go/tools/cmd/staticcheck"
)
```

cmd/go: accept main packages as dependencies in go.mod files #32504

(https://github.com/golang/go/issues/32504)

Project Structure

```
├── k8s/            # Kubernetes manifests
├── Dockerfile
└── RUNBOOK.md
```

```
1  FROM golang:1.13.6 as build
2  WORKDIR /go/src
3  COPY . .
4  RUN CGO_ENABLED=0 make build-mdserver
5
6  FROM scratch
7  COPY --from=build /go/src/build/mdserver /entrypoint
8  ENTRYPOINT ["/entrypoint"]
```

- Minimal Docker image size
- No dependencies
- Reduced attack surface
- Happy Site Reliability Engineer :)

Unit Tests

```
1  package test
2
3  import (
4      "testing"
5  )
6
7  func Test_myFunc(t *testing.T) {
8      tests := []struct {
9          name string
10          arg  int
11          want int
12      }{
13          {"test 1",
14              123, 456,
15          },
16      }
17      for _, tt := range tests {
18          t.Run(tt.name, func(t *testing.T) {
19              // ...
20          })
21      }
22  }
```

Unit Tests

```
tests := []struct {
    name string
    arg  int
    want int
}{
    {"test 1",
        123, 456,
    },
}
```

vs.

```
tests := map[string]struct {
    arg  int
    want int
}{
    "test 1": {
        123, 456,
    },
}
```

Unit Tests

```
tests := map[string]struct {
    arg  int
    want int
}{
    "test 1": {
        123, 456,
    },
}
```

- Test names have to be unique.
- Test names have a well defined place.
- Test order is unspecified!
- Check out github.com/stretchr/testify (https://github.com/stretchr/testify)

Go 1.14 Sneak-Peak

1. Jaana B. Dogan (@rakyll): Inlined defers in Go (https://rakyll.org/inlined-defers/)

```
mu.Lock()
defer mu.Unlock()
```

2. cmd/go: add GOINSECURE for insecure dependencies #32966 (https://github.com/golang/go/issues/32966)

- Go 1.13: Explicitly fetch insecure dependency

```
# Makefile
deps:
    go get -insecure $(REPO)@$(VERSION)

$ make deps build-mdserver
```

- Go 1.14: Allow insecure dependency host

```
# Makefile
export GOINSECURE = <Git host>

$ make build-mdserver
```

Thank you

Martin Mai
Site Reliability Engineer
trivago N.V.
martin.mai@trivago.com (mailto:martin.mai@trivago.com)
https://github.com/mrkanister (https://github.com/mrkanister)