Golang
Chemnitzer Linux Tage

Thomas Güttler, Syself

Sonntag, 10:00 - Raum V3 - Dauer 60 Min.

[me: Next]

Über mich

Thomas Güttler

BASIC/Pascal: ..1995

Studium: 1996-2001: C, C++, Prolog, Java, Visual Basic

Beruflich: Ab 2001 Python, Bash und etwas C.

Bis 2023: Python, Django, HTML/CSS/(JS), Linux-Server.

Ab 2023: Kubernetes & Go (Syself, Cluster API Provider Hetzner)

thomas-guettler.de

Go vs Golang

Kompiliert

… Executable

Anders als Python, Ruby, Java.

Mit Gargabe Collection

Anders als C, C++, Rust

Aber: Kein auto-close fd

Static Typing

→ Autocomplete ❤️

Modernes C

Hello Word ≈ 2 Mbyte

Inkl. Garbage Collector.

Entwickelt von Google

Vorteile, und Nachteile

Einfach

Keine “fancy” Features

Kein OOP, Keine Vererbung

“Langweilig”

```
result, err := doSomething(ctx, input)
if err != nil {
    ...
```

Keine Exceptions.

Nicht nur “happy path”

30% des Quelltexts sind Fehlerbehandlung.

Wrap Errors

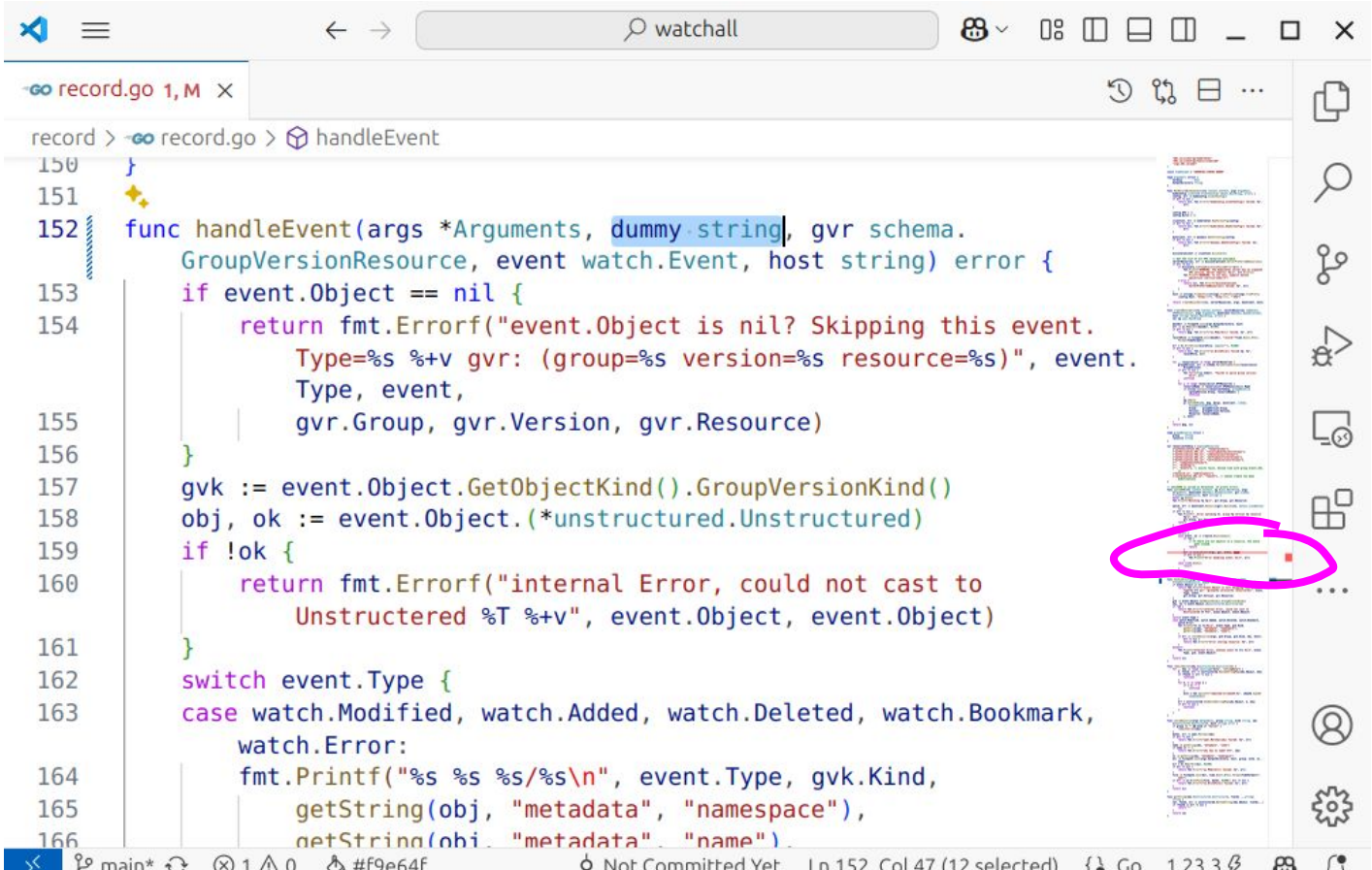
```
result, err := doSomething(ctx, input)
if err != nil {
    return fmt.Errorf("doSomething failed: %w", err)
}
```

Error handling

Eat that frog

Day1 vs Day2

Go Please: gopls (Language Server)



Tiobe Index

1	1		Python	23.85%	+8.22%
2	3	▲	C++	11.08%	+0.37%
3	4	▲	Java	10.36%	+1.41%
4	2	▼	C	9.53%	-1.64%
5	5		C#	4.87%	-2.67%
6	6		JavaScript	3.46%	+0.08%
7	8	▲	Go	2.78%	+1.22%
8	7	▼	SQL	2.57%	+0.65%
9	10	▲	Visual Basic	2.52%	+1.09%
10	15	⇈	Delphi/Object Pascal	2.15%	+0.94%

Concurrency

Go Proverbs #1

Don't communicate by sharing memory, share memory by communicating.

→Avoid Mutex

→ Use goroutines & Channels

ctx: context.Context

```
select {
case <-time.After(10 * time.Second):
    …
case <-ctx.Done():
    …
}
```

context.WithTimeout()

```
ctx, cancel := context.WithCancelCause(parentContext)

...

cancel(myError)
```

… aber

Concurrency, nur falls sinnvoll

Public

private

Auto-detect
unused. Nice!

“exported”

“unexported”

gofmt

`golangci-lint`

“fast” → In IDE

“all” → In CI

Generics

Packages: slices, maps

Selber schreiben? Eher nicht.

Copilot (AI)...

… fancy Autocomplete

Custom Proxies

… nur wenige Zeilen Go Code.

```go
package main

import (
    "net/http"
    "net/http/httputil"
    "net/url"
)

func main() {
    // Target server to proxy requests to
    targetURL, _ := url.Parse("https://example.com")

    // Create a ReverseProxy
    proxy := httputil.NewSingleHostReverseProxy(targetURL)

    // Handle HTTP requests using the reverse proxy
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        proxy.ServeHTTP(w, r)
    })

    // Start the server
    http.ListenAndServe(":8080", nil)
}
```

“No is temporary, yes is forever.”

Modules
&
Packages

github.com/guettli/**MODULE/PACKAGE**

Packages direkt ausführbar (vgl npx)

```
go run github.com/guettli/check-conditions@latest all
```

```
go install github.com/guettli/check-conditions@latest all
```

```
import (
    "fmt"

    "github.com/guettli/mymodule/mypackage"
)

func main() {
    mypackage.MyFunc()
    …
```

Common Packages

testify

cobra (Kommandozeilenparameter)

controller-runtime (Kubernetes)

go-cmp Diff()

templ (html Templating)

fyne (native GUI)

sqlc (type safe SQL)

kubernetes-sigs/yaml

Go eher nicht für …

Mikrocontroller (2 Mbyte)

ML (Python)

Einfache Scripte: Bash Strict Mode

Admin UI für eigenes SQL Schema: Django+Python

❤️ Du bist nicht alleine ❤️

Answering Machines:

Google, Bing, Stackoverflow, ChatGPT, Copilot …

Q & A:

reddit.com/r/golang/

stackoverflow.com/questions/tagged/go

Podcasts:

cupogo.dev/

(changelog.com/gotime)

Spielwiese

```
❯mkdir playground

❯code playground

user@pc:playground❯ mkdir foo

user@pc:playground❯ code foo/foo.go
```

Spielwiese

```
user@pc:playground> go mod init \
   github.com/YOU/playground

user@pc:playground> go mod tidy

user@pc:playground> go run ./foo
```

```go
package main

import (
    "fmt"

    "github.com/google/go-cmp/cmp"
)

func main() {
    fmt.Println(cmp.Diff("Hello, World!", "Hello, Go!"))
}
```

```
user@pc:playground> go run ./foo


   string(
-    "Hello, World!",
+    "Hello, Go!",
   )
```

```
❯ cat go.mod

module github.com/YOU/playground

go 1.23.3

require github.com/google/go-cmp v0.7.0
```

for Loop

```
for j := 0; j < 3; j++ {
    fmt.Println(j)
}

for i := range 3 {
    fmt.Println("range", i)
}


for {
    fmt.Println("loop")
    break
}
```

switch

```
switch time.Now().Weekday() {
case time.Saturday, time.Sunday:
    fmt.Println("It's the weekend")
default:
    fmt.Println("It's a weekday")
}
```

slice: Capacity & Length

```
// s is nil
var s []string

// length 0, capacity 3
s = make([]string, 0, 3)

// length 3, capacity 3
s = make([]string, 3)
```

Append to Slice

```
s = append(s, "d")
```

Keine List Comprehension

```
func double(intSlice []int) []int {
    res := make([]int, 0, len(intSlice))
    for _, v := range intSlice {
        res = append(res, v*2)
    }
    return res
}
```

```go
m := make(map[string]int)

m["k1"] = 7
m["k2"] = 13

v, ok := m["k1"]

for k, v := range m {
    fmt.Println("key:", k, "value:", v)
}
```

```
func plus(a int, b int) int {
    return a + b
}

res := plus(1, 2)
```

```go
type person struct {
    name string
    age  int
}

func newPerson(name string) *person {
    p := person{name: name}
    p.age = 42
    return &p
}
```

Interface: Liste von Methoden

```
type geometry interface {
    area() float64
    perim() float64
}
```

```
type rect struct {
    width, height float64
}

func (r rect) area() float64 {
    return r.width * r.height
}
func (r rect) perim() float64 {
    return 2*r.width + 2*r.height
}
```

```
type circle struct {
    radius float64
}

// The implementation for `circle`s.
func (c circle) area() float64 {
    return math.Pi * c.radius * c.radius
}

func (c circle) perim() float64 {
    return 2 * math.Pi * c.radius
}
```

```
func f(arg int) (int, error) {
    if arg == 42 {
        return -1, fmt.Errorf("can't work with 42")
    }

    return arg + 3, nil
}
```

```go
var (
    ErrOutOfTea = fmt.Errorf("no more tea available")
    ErrPower    = fmt.Errorf("can't boil water")
)

func MakeTea(arg int) error {
    if arg > 5 {
        return fmt.Errorf("making tea: %w", ErrPower)
    } else if arg > 3 {
        return fmt.Errorf("making tea: %w", ErrOutOfTea)
    }
    return nil
}
```

Channels

```
queue := make(chan string, 2)
queue <- "one"
queue <- "two"
close(queue)

for elem := range queue {
    fmt.Println(elem)
}
```

```
func worker(id int, jobs <-chan int, results chan<- int) {
    for j := range jobs {
        fmt.Println("worker", id, "started  job", j)
        time.Sleep(time.Second)
        fmt.Println("worker", id, "finished job", j)
        results <- j * 2
    }
}
```

```
func main() {
    const numJobs = 5
    jobs := make(chan int, numJobs)
    results := make(chan int, numJobs)

    for w := 1; w <= 3; w++ {
        go worker(w, jobs, results)
    }

    for j := 1; j <= numJobs; j++ {
        jobs <- j
    }
    close(jobs)

    for a := 1; a <= numJobs; a++ {
        <-results
    }
}
```

```go
func Do() error {
    // Open the file for reading
    file, err := os.Open("example.txt")
    if err != nil {
        return fmt.Errorf("could not open file: %w", err)
    }
    // Ensure the file is closed when the function ends
    defer file.Close()

    // Create a new scanner to read the file line by line
    scanner := bufio.NewScanner(file)
    for scanner.Scan() {
        fmt.Println(scanner.Text())
    }

    if err := scanner.Err(); err != nil {
        return fmt.Errorf("could not read file: %w", err)
    }

    return nil
}
```

```
data, err := os.ReadFile("example.txt")
if err != nil {
    retur fmt.Errorf("could not read file: %w", err)
}
```

```
panic(1)
```

```
.....
panic: 1
goroutine 1 [running]:
main.foo(...)
    /home/guettli/projects/gobyexample/tmp/main.go:16
main.main()
    /home/guettli/projects/gobyexample/tmp/main.go:12 +0xbf
exit status 2
```

```
import (
    "embed"
)

//go:embed folder/single_file.txt
var fileString string
```

```go
package main

import (
    "testing"

    "github.com/stretchr/testify/require"
)

func Add(a, b int) int {
    return a + b
}
func TestAdd(t *testing.T) {
    result := Add(3, 5)
    require.Equal(t, 8, result)
}
```

Go By Example

Für Vortag: https://thomas-guettler.de/go/

Viel Spaß mit Go!