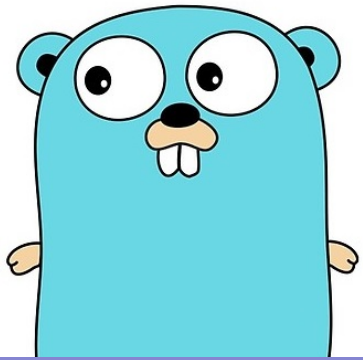
Go - Golang

Aktas, Bonini, Bulut, Kocher

24. Jänner 2020



Inhaltsverzeichnis

Einstieg

- 2007 wurde der Grundstein für Go gelegt
- Unzufriedenheit über gängige Programmiersprachen (C++, Java, ...)
- Entwicklung mit Ausblick auf zukünftige Technologien
- Release von Version 1.0 am 28.3.2012

Signifikante Unterschiede

- Syntax grundsätzlich an C-Familie orientiert
- Wirthsche Sprachen (Pascal, Modula, Oberon)
- Runde Klammern bei Bedingungen nicht zwingend notwendig
- Semikolon nicht zwingend notwendig
- Keine while-Schleife, nur for-Schleife

Hallo Welt!

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Print("Hallo Welt")
7  }
```

Vorzüge

- Simplizität und Multifunktionalität
- Keine Sprache weist alle 3 Vorzüge auf:
  - Effiziente Code-Kompilierung
  - Schnelle Codeausführung
  - Einfacher Programmierungsprozess
- Kombination wichtiger Features aus unterschiedlichen Sprachfamilien

Wichtige Besonderheiten

- GOPATH-Umgebung
  - $GOPATH/bin
  - $GOPATH/pkg
  - $GOPATH/src
- Modularer Aufbau mit Packages
  - Quelldateien lassen sich modular über Verzeichnisse organisieren
  - Name des Verzeichnisses = Name des Pakets
- Einheitliche Code-Formatierung
  - Konventionen für die Formatierung des Codes
  - *gofmt*

Wichtige Besonderheiten

- Relative Importe als Standard
  - Pakete und Dateien sind relativ zum Verzeichnis $GOPATH/src
  - Go kompiliert importierte Elemente nicht, wenn diese nicht genutzt werden
- Multiple Rückgabewerte für Funktionen und Methoden
  - $func(x\ Node, y\ List)(i\ int, j\ int)$
  - $func(x\ Node, y\ List)(int, int)$
  - $func(...int)$
  - $func(int...)$

Unternehmen, die Go benützen...

- 2061 Unternehmen verwenden Go (laut *stackshare.io*, 23.01.2020)

Performance

- Allgemein sehr gute Performance
- Go wird kompiliert, nicht interpretiert
- Statische Typisierung statt dynamischer Typisierung
- Nur die nötigsten Dependencies werden in die Binary inkludiert

Warum ist Go langsamer als es sein könnte?

- Neue Sprache, mit noch nicht ganz optimierten Libraries
- Die Codeoptimierung ist noch nicht ganz ausgereift
- Der Garbage Collector verbraucht viele Ressourcen
- Go ist memory-safe

Pointer

- Daten können effizient und einfach an eine Funktion übergeben werden
- Pointer sind sehr ähnlich zu C Pointern
- Pointer zu Funktionen mit Umwegen möglich
- Pointerarithmetik ist nicht möglich

Goroutines und Channels

- Goroutines sind Funktionen die gleichzeitig mit anderen Funktionen ablaufen
- Goroutines werden auf mehrere Threads verteilt
- Channels sind Verbindungen zwischen Goroutines
- Parallelisieren von Rechnungen über Signale durch Channel leicht möglich

The Computer Language Benchmarks Game (Stand 22.01.2020)

- Nachkommastellen von Pi berechnen (pidigits):

source	secs	mem
Go	2,10	8448
Java	3,07	39320
C++	1,89	4552
Python3	3,47	10356

- Hashtable erstellen und updaten (k-nucleotide):

source	secs	mem
Go	11,77	160184
Java	9,33	447976
C++	3,90	156216
Python3	72,24	199856

Syntax

```
1  package main
2
3  import "fmt"
4
5  func main() {
6    fmt.Println ( "Hello world!" )
7    fmt.Println ( "Ciao", "mondo!" ) // Italienisch
8
9    for i := 1; i < 3; i++ {
10     if i % 2 == 0 {
11       fmt.Println ( "Merhaba" + "dünya!"); /* Türkisch */
12     }
13   }
14
15   goto a
16   for true { fmt.Println("Stop me") }
17   a:
18 }
```

Gegenüberstellung Go / Java

```go
package main

import ( "fmt"; "io/ioutil" )

func main() {
  dat, err := ioutil.ReadFile("dat.txt")
  if err != nil {
    panic(err)
  }
  fmt.Print(string(dat))
}
```

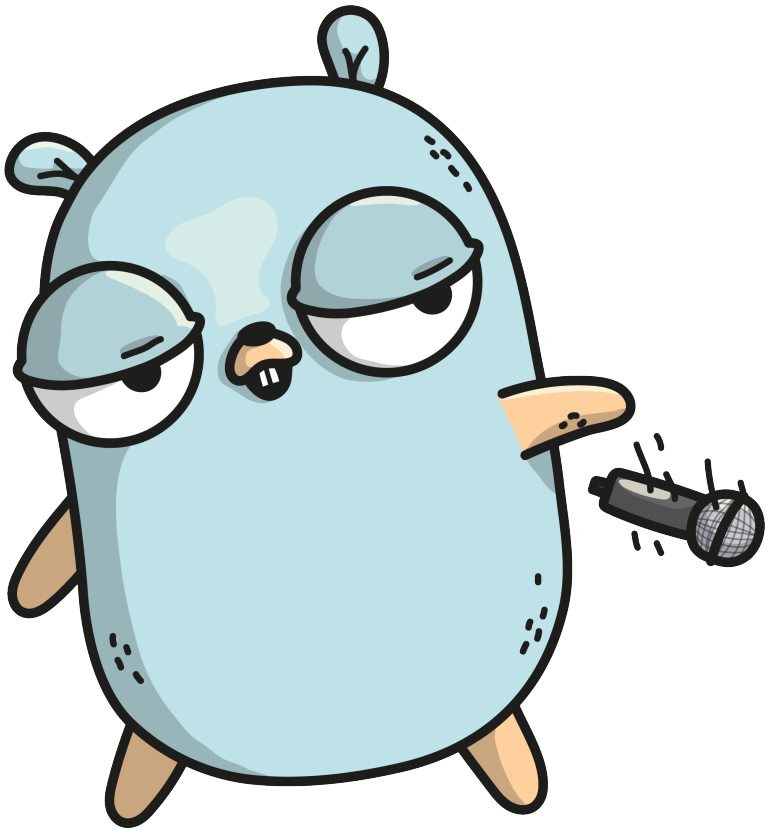
```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class Golang {
  public static void main(String args[]) throws IOException {
    File file = new File("dat.txt");

    BufferedReader br =
      new BufferedReader(new FileReader(file));

    String dat;
    while ((dat = br.readLine()) != null) {
      System.out.println(dat);
    }
  }
}
```

Schwachpunkte

- Es fehlen generische Typen
- Objekt-Orientierung ist nur bedingt möglich
- Interfaces besitzen ein anderes Konzept als in Sprachen wie Java
- Vorhandene Library klein im Vergleich zu C++, Java und Co.
- Wenige Experten, die die Programmiersprache beherrschen

Schlussfolgerung

- Mögliche Alternative zu etablierten Sprachen
- Bessere Performance als viele gängige Sprachen
- Schnell erlernbar für Einsteiger in die Programmiersprache
- Vor allem für den Business- und Serverbereich interessant



Vielen Dank für

Eure Aufmerksamkeit!

Quellen (Stand 23.01.2020)

- Textquellen:
  - https://benchmarksgame-team.pages.debian.net/benchmarksgame/
  - golang.org/doc/
  - https://golangbot.com/goroutines/
  - https://github.com/golang/go
  - https://stackshare.io/go
  - https://www.ionos.de/digitalguide/server/knowhow/golang/
- Bildquellen (Reihenfolge nach Auftreten):
  - https://www.redbubble.com/de/people/vladocar/works/27088115-go-golang-gopher?p=poster
  - https://upload.wikimedia.org/wikipedia/commons/0/05/Go_Logo_Blue.svg
  - https://stackshare.io/go
  - https://womenwhogo.threadless.com/designs/gopher-drop/