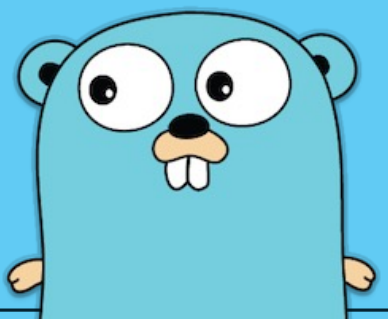
Fortgeschrittene Programmierkonzepte anhand einer einfachen eDSL in GO

Kevin Becker



Agenda

1. Was bedeutet ,,eDSL“?
2. Werkzeuge der Metaprogrammierung
3. Welche Werkzeuge besitzt GO?
4. Code Implementierung
5. Fazit: eignet sich GO für eDSL-Programmierung?

Was bedeutet „eDSL“?

eDSL == “embedded domain-specific Language“

1. embedded: eingebettet, integriert
2. domain-specific: einer bestimmten Domäne angehörig
3. Language: Sprache mit eigenem Grammatiksystem bzw. Syntax

Was bedeutet „eDSL“?

Beispiel DSL 1: HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

Was bedeutet „eDSL“?

Beispiel DSL 2: SQL

```sql
SELECT C.customer_name,
       C.customer_id,
       O.order_id,
       O.order_date,
       P.product_name,
       P.product_price,
       SUM(OL.quantity * OL.unit_price) AS total_order_cost
FROM customers C
JOIN orders O ON C.customer_id = O.customer_id
JOIN order_lines OL ON O.order_id = OL.order_id
JOIN products P ON OL.product_id = P.product_id
WHERE C.customer_state IN ('CA', 'NY')
GROUP BY C.customer_name,
         O.order_id,
         P.product_name,
         P.product_price
HAVING total_order_cost > 1000
ORDER BY O.order_date DESC;
```

Was bedeutet „eDSL“?

Beispiel eDSL 1: HTML in GO mit html/template

```go
package main

import (
    "html/template"
    "os"
)

type PageData struct {
    Title   string
    Message string
}

func main() {
    tmpl := template.Must(template.New("index").Parse(`
    <!DOCTYPE html>
    <html>
    <head>
        <title>{{.Title}}</title>
    </head>
    <body>
        <h1>{{.Message}}</h1>
    </body>
    </html>
`))

    // Daten für das Template
    data := PageData{
        Title:   "Meine Go HTML-Seite",
        Message: "Hallo, das ist eine eDSL in Go mit HTML Templates!",
    }

    // Template rendern
    tmpl.Execute(os.Stdout, data)

}
```

Was bedeutet „eDSL“?

Beispiel eDSL 2: SQL in C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    char query[1024];
} SQLQuery;

#define SELECT(columns) "SELECT " columns
#define FROM(table) " FROM " table
#define WHERE(condition) " WHERE " condition
#define AND " AND "
#define OR " OR "

SQLQuery sql_build(const char* select, const char* from, const char* where) {
    SQLQuery q;
    snprintf(q.query, sizeof(q.query), "%s%s%s;", select, from, where);
    return q;
}

int main() {
    SQLQuery q = sql_build(
        SELECT("*"),
        FROM("users"),
        WHERE("age > 18") AND WHERE("country = 'DE'")
    );

    printf("Generated SQL Query: %s\n", q.query);
    return 0;
}
```

Werkzeuge der Metaprogrammierung

Metaprogrammierung == Fähigkeit einer Programmiersprache, Code zu generieren und eigene Strukturen zu analysieren & modifizieren

Ziel: Effiziente Code-Generierung und Entwicklung von eDSLs

=> Welche Werkzeuge bietet GO?

Werkzeuge der Metaprogrammierung

1. Interfaces und Polymorphie

```
10  type Expression interface {
11      eval(num float64) float64
12      derive() Expression
13      latex() string
14  }
```

Werkzeuge der Metaprogrammierung

2. Rekursion

```
72  type Add struct {
73      left, right Expression
74  }
75
76  func (a Add) eval(num float64) float64 {
77      return a.left.eval(num) + a.right.eval(num)
78  }
```

Werkzeuge der Metaprogrammierung

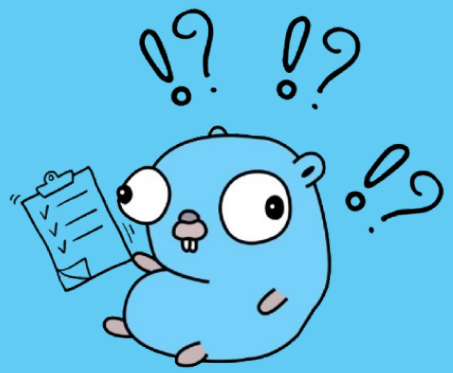
3. String Manipulation

```
33    func (f Func) latex() string {
34        return fmt.Sprintf("\\( f(x) = %s \\)", f.fn.latex())
35    }
```

Werkzeuge der Metaprogrammierung

Welche nützlichen Features fehlen in GO?

- ➔ **Methoden- & Operatorenüberladung**
- ➔ **Makros**
- ➔ **Templates**



Code Implementierung

1. Aufgabenstellung: eDSL für Mathematische Ausdrücke

- Insgesamt 9 Structs und 1 Interface
- 3 Basisfunktionen: eval(), derive(), latex()

- Structs: Func, Var, Const, Add, Sub, Mult, Div, Pow, Sqr
- Interface: Expression

Code Implementierung

Spieleregeln für unsere Funktionen-eDSL:

1. **Basistyp Func{} definieren**
2. **Min. 1 Ausdruck definieren (Add, Sub, Mult …)**
3. **Const{c} für Konstante c oder Var{} für x-Wert im Ausdruck nutzen**
4. **.eval(num), .latex() oder .derive() benutzen**

Code Implementierung

2. Aufgabenstellung: eDSL für SVG-Vektorengrafiken

- Insgesamt 6 Structs und 1 Interface
- Zwei Funktionen: toSVG() und saveSVG()

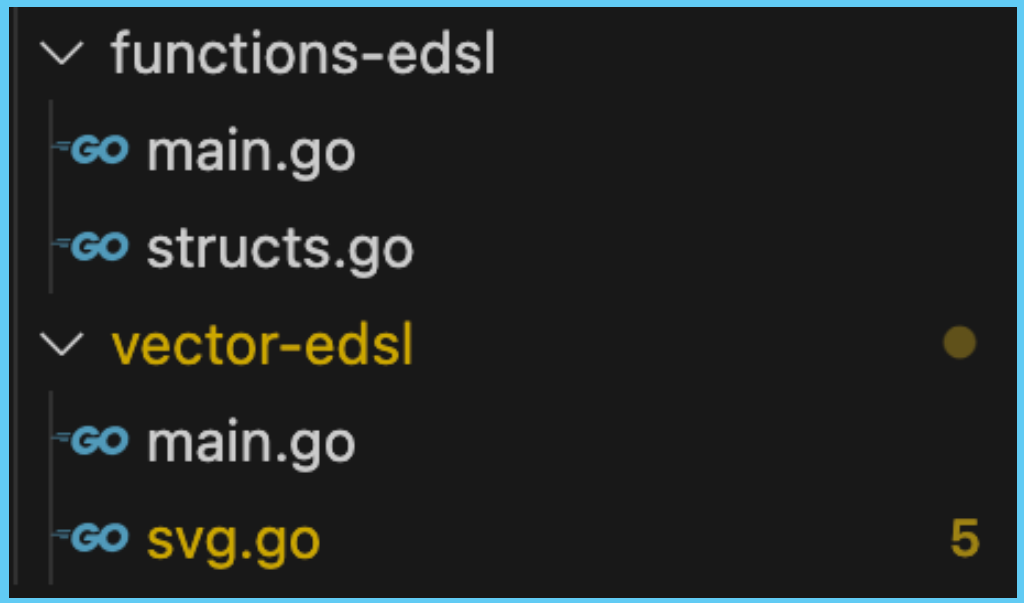
- Structs: svg, rect, circle, line, text, ellipse
- Interface: Element

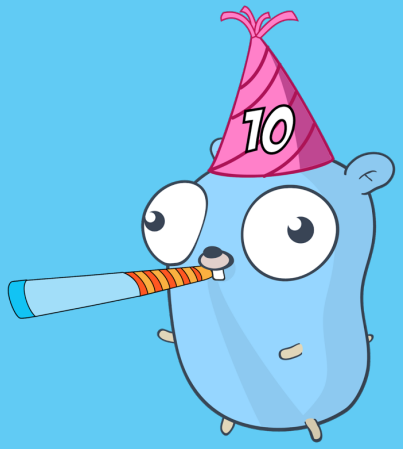
Code Implementierung

Spieleregeln der Vektoren-eDSL:

1. **Basistyp svg{} definieren**
2. **Width, height und []Elements definieren**
3. **Für jedes Element die required Attributes initialisieren**
4. **Entweder fmt.Println(svg.toSVG()) für Konsolenausgabe oder .saveSVG() für SVG-Datei**

Code Implementierung





Fazit: eignet sich GO für eDSL-Programmierung?

Vorteile

+ Structs & Interfaces
+ Durch Standardbibliotheken möglich
+ Klar strukturierte API

Nachteile

- „verbose“
- Laufzeit basierend
- Nur begrenzt möglich (Fehlende Features)

Quellen

https://www.revelo.com/blog/metaprogramming

https://missing.csail.mit.edu/2020/metaprogramming/

https://www.w3schools.com/html/html5_svg.asp

https://de.wikipedia.org/wiki/Dom%C3%A4nenspezifische_Sprache

https://learn-haskell.blog/03-html/03-edsls.html