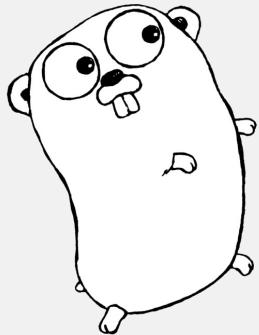


a p p  
m o t  
i o n



# Go

Boring and Fast

# Was ist Go?



- Entwickelt von Google
- Garbage-Collected
- “Wie könnte C++ heute aussehen?”

# Wer benutzt Go?



Google

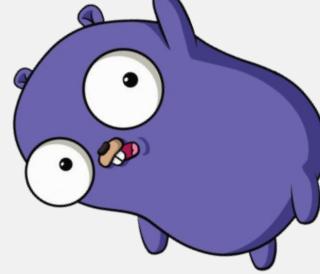


kubernetes

twitch

Uber

# Todo App - Aspekte

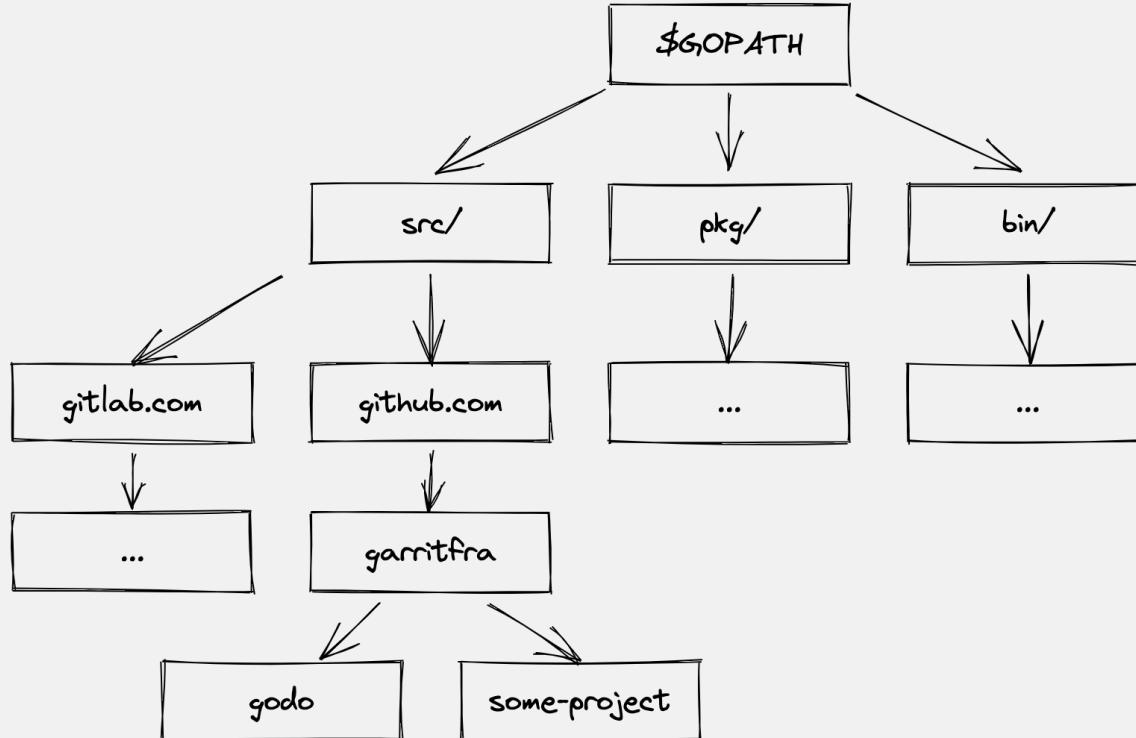


- Projekt Setup
- HTTP-Server
- GraphQL
- Datenbank (SQLite)
- Frontend

# Demo Time!



## Setup - \$GOPATH



## Setup - Hello World!

```
package main

import "fmt"

func main() {
    fmt.Println("Hello World!")
}
```

## HTTP-Server

```
● ● ●  
package main  
  
import "github.com/gin-gonic/gin"  
  
func main() {  
    r := gin.Default()  
    r.GET("/ping", func(c *gin.Context) {  
        c.JSON(200, gin.H{  
            "message": "pong",  
        })  
    })  
    r.Run() // listen and serve on 0.0.0.0:8080  
}
```

- API vergleichbar mit Express.js
- Nutzt Standard-Bibliothek (net/http)

# GraphQL

```
type Todo {  
  id: ID!  
  text: String!  
  done: Boolean!  
}
```

todo.graphql

```
go generate ./...
```

- `gqlgen` Library
- Code Generierung
- Setup sehr straight-forward

```
type Todo struct {  
  ID   string `json:"id"  
  Text string `json:"text"  
  Done bool  `json:"done"  
}
```

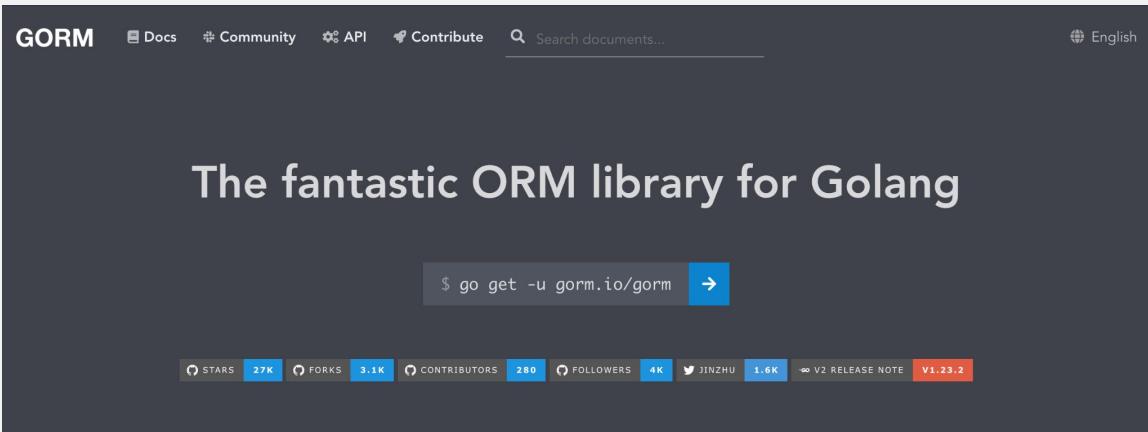
todo.go

## GraphQL - Resolver

```
● ● ●  
func (r *mutationResolver) UpdateTodo(  
    ctx context.Context,  
    input model.UpdateTodoInput  
) (*model.Todo, error) {  
    var todo *model.Todo  
  
    result := r.DB.First(&todo).Where("id", input.ID)  
  
    if result.Error != nil {  
        return nil, result.Error  
    }  
  
    if input.Done != nil {  
        todo.Done = *input.Done  
    }  
  
    result = r.DB.Save(&todo)  
  
    return todo, result.Error  
}
```

- C-ähnliche Patterns
- Error Handling

# Datenbank



The screenshot shows the GitHub repository page for GORM. At the top, there's a navigation bar with links for Docs, Community, API, Contribute, and a search bar. On the right, it says "English". Below the header, the title "The fantastic ORM library for Golang" is displayed in large white text. Underneath the title is a button with the command "\$ go get -u gorm.io/gorm" followed by a blue arrow button. At the bottom of the main dark section, there are social icons for GitHub, Forks (27K), Contributors (3.1K), Followers (4K), Twitter (JINZHU 1.6K), and a V2 RELEASE NOTE link. A red "V1.23.2" badge is also present. The footer contains three bullet points: "Full-Featured ORM", "Associations (has one, has many, belongs to, many to many, polymorphism, single-table inheritance)", and "Hooks (before/after create/save/update/delete/find)".

- Reiches Set an Features
- Sehr leichtes Setup
- Unterstützt mehrere DBs (SQLite, Postgres, MySQL, ...)

## Datenbank - Models

```
type Todo struct {
    gorm.Model
    ID   string `json:"id" gorm:"column:id"`
    Text string `json:"text" gorm:"column:text"`
    Done bool   `json:"done" gorm:"column:done"`
}
```

- Generierte GraphQL Models können erweitert werden
- Auto-Migrations
- DB-Instanz hängt an Resolver Context

## Todo App - Fazit



- Sehr schnelle Entwicklung
- Keine versteckten Überraschungen
- Syntax erfrischend einfach
- Einheitliches Tooling (Formatting, Docs, ...)
- \$GOPATH ist gewöhnungsbedürftig

## Go - Erwähnenswert



- Goroutines
- Generics sind noch sehr neu (Type-System sehr flexibel)
- Tooling legt Wert auf Korrektheit (öffentliche Funktionen müssen dokumentiert sein)

## Wann Go?



- Performance ist kritisch
- Häufig wechselnde Devs (schnelles Onboarding)
- Langfristige Projekte

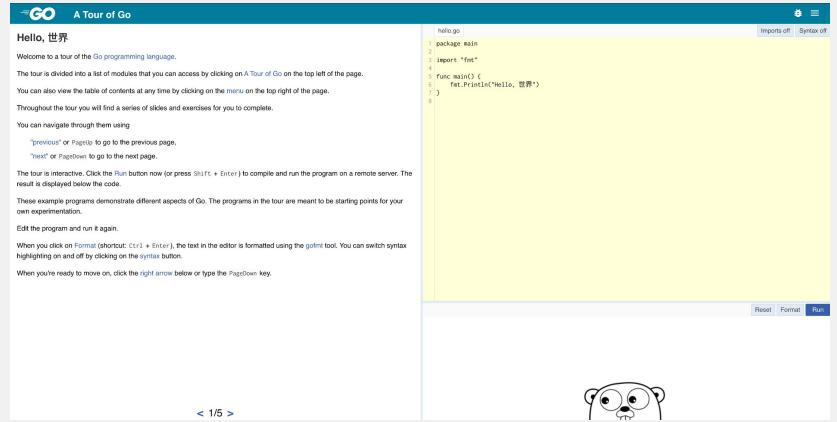
## Wann nicht Go?\*

\*für Web-Applikationen



- Entwicklung von Prototypen
- Content Management Systeme
- Kompilierte Sprachen sind keine Option oder Laufzeit wird nicht unterstützt

# Du willst auch mal Hand anlegen?



The screenshot shows a web browser displaying the "A Tour of Go" page. The title bar says "GO A Tour of Go". The main content area has a dark header "Hello, 世界". Below it is a paragraph of text about the tour, followed by a code editor window titled "hello.go". The code in the editor is:

```
hello.go
1 package main
2 import "fmt"
3 func main() {
4     fmt.Println("Hello, 世界")
5 }
```

Below the code editor, there's a section with instructions for navigating the tour and running the code. At the bottom right of the editor is a small cartoon character icon.

<https://go.dev/tour>

# Fragen?

