Golang 써본 후기

신연진

2023년 11월 1일

since:1991 ZEROPAGE

since:1991 ZEROPAGE

들어가는 말

- 기존에 운영되던 다음 사이트들은 Node.js로 개발되어 있었다.
  - https://calendar.puang.network
  - https://pre-timetable.puang.network
  - https://rss.puang.network
- Go언어가 괜찮다던데? 위 사이트들을 Go언어로 재작성해볼까?
  - 그리하여 Go언어를 써보게 됐다.

since:1991 ZEROPAGE

Go 언어?

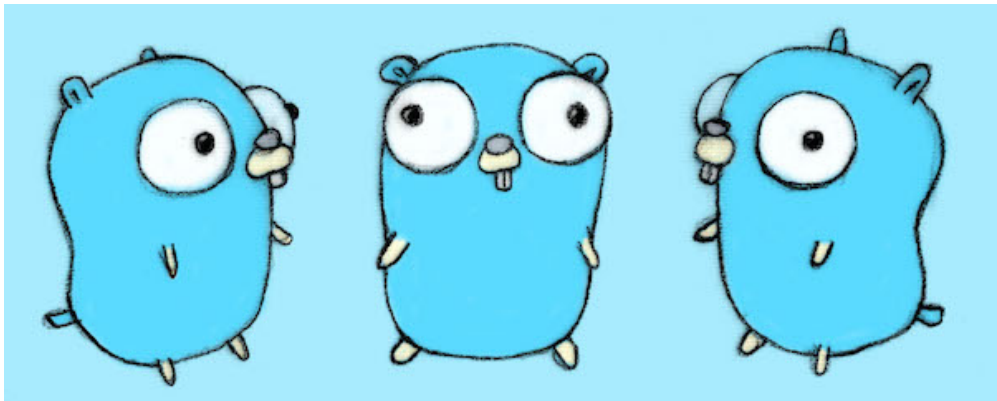

Figure: Go gopher

- Google이 2009년에 개발한 언어
- 뇌비우고 코딩하기 좋은 언어
- 쓰다보면 묘하게 C언어 느낌이 좀 든다...

ZEROPAGE since:1991

Go 언어의 장점

- 추상화의 한계 – 클래스 상속같은 개념이 없기 때문에 지나친 추상화로 인해 고통받을 일이 없다.
  - C언어에서 struct와 함수로만 코딩하는 느낌. 다만 struct에 함수를 묶을 수는 있다.
- 간단한 채널과 비동기 호출
  - 함수 앞에 *go* 키워드 하나만 비동기로 실행된다.
  - 채널을 이용하면 비동기로 호출된 Task 간의 데이터 전달이 쉽다.
- 컴파일 언어
  - 당연하지만 인터프리터 언어보다 빠르다.
- exception handling이 없다. 오류를 return value로 던진다(!)

since:1991 ZEROPAGE

명령어

- *go mod init zeropage.org/example* – 프로젝트 초기화
- *go run* – 실행
- *go build* – 빌드
- *go test* – 테스트
- *go get example.com/example_pkg* – 패키지 설치

Go언어 문법 I

- 변수 선언

```
var test int // Define
test = 1 // Assign

test2 := 2 // Define and assign

test3 = 3 // Error! test3 is not defined
```

- 형변환

```
var i int = 42
f := float64(i)
```

since:1991 ZEROPAGE

Go언어 문법 II

- 반복문

```
for i := 0; i < 5; i++ {
    fmt.Printf("%d\n", i)
}
j := 0
for j < 100 {
    j += 1
    fmt.Printf("%d\n", j)
}
for {
    fmt.Printf("Infinitttttty")
}
```

since:1991 ZEROPAGE

Go언어 문법 III

- 함수 선언

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

- 조건문과 복수 개의 반환값

```
fnuc addPositives(a int, b int) (int, error) {
    if a <= 0 || b <= 0 {
        return nil, fmt.Errorf("They're not
            positives!'")
    }

    return a + b, nil
}
```

since:1991 ZEROPAGE

Go언어 문법 IV

- 기명의 반환값

```
func plusTwoAndFour(a int, b int) (c, d int) {
    c = a + 2
    b = d + 2
    return
}
```

- 패키지 선언 및 패키지 import

```
package main

import (
    "fmt"
    "example.com/example_pkg"
)
```

since:1991 ZEROPAGE

Go언어 문법 V

- defer

```
func example() {
    defer fmt.Println(",-world!")
    fmt.Print("Hello")
}
```

- switch
  - 참고: 조건 없으면 switch true랑 동일함

```
switch a {
    switch 1:
    // Do something
    switch 2:
    // Do something
    default:
    //Do something
}
```

since:1991 ZEROPAGE

Go의 비동기 문법 I

- 비동기 호출

```
func worker(a int) {
    for {
        // Do something
    }
}

func main() {
    for i := 0; i < 0; i++ {
        go worker(i)
    }
}
```

since:1991 ZEROPAGE

Go의 비동기 문법 II

- 채널

```
package main

import "fmt"

func sum(s []int, c chan int) {
  sum := 0
  for _, v := range s {
    sum += v
  }
  c <- sum // send sum to c
}

func main() {
  s := []int{7, 2, 8, -9, 4, 0}

  c := make(chan int)
  go sum(s[:len(s)/2], c)
  go sum(s[len(s)/2:], c)
  x, y := <-c, <-c // receive from c

  fmt.Println(x, y, x+y)
}
```

since:1991 ZEROPAGE

Go의 비동기 문법 III

- range

```
func printer(c <-chan int) {
    for i := range c {
        fmt.Printf("%d", c)
    }
}

func main() {
    ch := make(chan int, 10)
    go printer(ch)
    go printer(ch)
    go printer(ch)
    for i := 0; i < 100; i++ {
        ch <- i
    }
}
```

since:1991 ZEROPAGE

그래서 Go를 왜 쓰는가?

- 단순한 문법
  - 단순하다 $\rightarrow$무식하게 코딩하기 딱 좋다
- 테스트 프레임워크 기본 내장
  - 테스트 코드 짜기 편하다
- 동시성
  - 함수 호출 앞에 *go*만 붙이면 비동기 호출이 된다고?
- 안 쓰는 변수가 있을 시 오류 발생
  - 호불호가 갈리는 부분이지만, 실수를 예방해준다는 측면으로 보면 좋은 점이다.

since:1991 ZEROPAGE