

# GOLANG 모각소

임학범

**TIL1(01.19)**

# go 실행 명령어

- `go run main.go`: `main.go` 파일 실행
- `go mod init example`: main module 파일 만들기
- `go build`: 실행파일 만들기(module을 만들고 난 이후에 build 가능)

# go 특징

- 클래스, 상속, 제너릭 프로그래밍, 네임스페이스 기능 x
- 메서드, 인터페이스, 익명함수, 가비지 컬렉터, 포인터 기능 o
- 클래스 대신 메서드를 가지는 구조체 지원
- 익명함수는 함수 리터럴이라는 이름으로 제공

# go type

- 숫자
  - 정수
    - uint8, uint16, uint32, uint64
    - int8, int16, int32, int64
    - byte = uint8
    - rune = int32
    - int, 32bit -> int32, 64bit -> int64
    - uint, 32bit -> uint32, 64bit -> uint64
  - 실수
    - float32, float64
  - 복소수
    - complex64: 8byte, complex128: 16byte
- bool
- string
- array
- slice
- struct
- pointer
- function
- map
- interface
- channel

# Go는 casting이 굉장히 중요하다.

- 타입이 다르면 연산이 안됨 cf) int와 int64끼리도 안됨
- casting 방법
  - ex) `int64(variable)`
- casting 시 주의사항
  - 실수->정수: 소수점 잘림
  - 큰 메모리를 사용하는 수에서 작은 메모리를 사용하는 type으로 변경시 수가 이상해질 가능성 존재

# Go 기본 문법

- 함수 정의
  - `func name() {}`
- 주석
  - `//`
  - `/* */`
- 변수선언
  - `var a int = 20`
  - `var a int // a는 0임`
  - `var a = 10 //type을 안정할시 선언시 초기값이 있어야함`
  - `a:=10`

**TIL2(01.26)**



# Go 연산자

- +, -, \*, /, %
- 비트연산: &, |, ^, &^, <<, >>
- 논리연산: &&, ||, !
- 복합 대입연산자: +=, -=, ...
- 증감 연산자: ++, --

# Go 함수

- 함수 선언:

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

- 함수명: Add
- Parameter: int type의 a,b
- Return 값: int type

# Go 상수

- `const a int = 10`
- `iota`

`iota`는 한 `const` 변수마다 +1이 되어 할당이 된다. ex)  
`constExample.go`

`1 << iota`로 사용하여 비트 계산도 쉽게 가능하다

- `type`없는 상수는 `type`을 설정안해줘도 자동으로 casting 됨

**TIL3(02.10)**

# GOPATH, GOROOT

- Go에는 여러 환경변수가 있다. `go env`를 하면 전체의 환경 변수 목록을 볼 수 있다.
- 그중 GOPATH, GOROOT는 package와 src를 담당하는 path를 담고있다.

# 외부라이브러리 IMPORT

```
package main
```

```
import "fmt"
```

```
import "rsc.io/quote"
```

```
func main() {  
    fmt.Println(quote.Go())  
}
```

- 터미널에 `go get rsc.io/quote`  
실행하면 `rsc.io/quote` package가 받아지며 그 후 `go file`을 실행하면 된다.

# 내부라이브러리 IMPORT(1)

- 같은 디렉토리(패키지)내에 있다면 그냥 그대로 import도 필요없이 사용하면 된다.
- 다른 디렉토리에 있을 시
  - 일단 두 디렉토리에 mod파일을 둘다 만들어야한다.
  - 그 후 module을 가져올 파일에 module이름에 맞게 import를 해준다.
  - 다음의 맞게 코드를 작성하여 shell에 작성한다.
  - ```
go mod edit -replace [module이름]=[module까지의 상대주소]
```

ex) 

```
go mod edit -replace module/config=./module/config
```
  - 
  - 
  - 그 후 

```
go get [module이름]
```

한 후 실행을 돌리면된다.
- 참고로, 대문자로 시작하면 public, 소문자로 시작하면 private이기에 공유하고 싶다면 대문자로 시작해야한다.


# 내부라이브러리 IMPORT(2)

- Workspace 사용
- Main함수 있는 위치에서 다음의 명령어를 실행한다. (`go work init`)
- 그 후 import하고자하는 내부 라이브러리 module이 있다면 다음의 명령어를 실행한다. (`go work use [module 이름]`)
- 이후 import를 사용해서 원하는 라이브러리를 사용가능하다.
- Go mod replace방법보다 굉장히 간단하고 쉬운 방법이다.
- Go 1.18버전 이후부터 사용가능하다.



**TIL4(02.16)**

# 보안없는 로그인 구현

- 단지 database와 연동하여 id와 pw로 로그인 구현
-  사용스택
- Go (framework : gin-gonic)
- DB: firebase의 firestore
- Develop Tool : Visual Studio Code

## mvc 패턴 사용

- controller, model, service, repository 구분

## firestore 사용

- nosql로 객체 그대로 쉽게 삽입 가능

## api 구현

- signup, login 두개의 api구현

**TIL5(02.24)**

# 로그인 + 간단 블로그 구현

- 단지 id, pw를 통신하여 로그인을 구현 및 글쓰기 기능 구현

- 🗝️ 사용스택
- Go (framework : gin-gonic)
- DB: firebase의 firestore
- Develop Tool : Visual Studio Code

## mvc 패턴 사용

- controller, model, service, repository 구분

## firestore 사용

- nosql로 객체 그대로 쉽게 삽입 가능

## api 구현

- 로그인: signup, login 두개의 api구현
- 게시물: CRUD 기능 구현