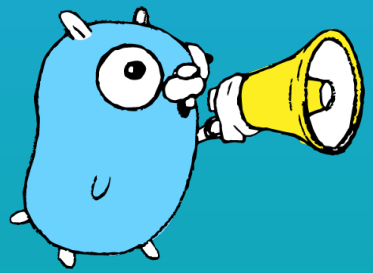
KM#14 @Mastertech, AUG 30 2019

RESTful + DB

GolFz

Mastertech

Engineer, R&D



Hi, Let’s talk about

01

Workspace

ตรวจสอบเวอร์ชันของ Go

- ใช้คำสั่ง go version

Go Workspace (2)

สร้าง GOPATH สำหรับ Go

- ใช้คำสั่ง go env เพื่อดูค่า GOPATH
- ใช้คำสั่ง mkdir -p $GOPATH

สร้าง Directory ใน GOPATH

- src
- bin
- pkg

Go Workspace — Environment Variable

```
export PATH=$PATH:$(go env GOPATH)/bin
```

```
export GOPATH=$(go env GOPATH)
```

Go Workspace — Your first program

```
mkdir $GOPATH/src/mastertech.co.th/hello
```

02

struct

struct คือ

ประเภทข้อมูล Collection

ที่มีชื่อ field

Struct

```
type person struct {
        name string
        age  int
        tel  *string
}
```

Struct — Create (1)

```
type person struct {
        name string
        age  int
        tel  *string
}

p := person{}
```

Struct — Create (2)

```
type person struct {
        name string
        age  int
        tel  *string
}

p := person{
        name: "Michael Jordan",
        age: 56,
}
```

03

package

package คือ

- หน่วยที่เล็กที่สุดของ Software ใน Go
- package ชื่อเดียวกับ directory
- Go ไม่มี Class
- export member ด้วย UpperCase

วันนี้เราจะใช้

- github.com/gorilla/mux
- database/sql
- github.com/go-sql-driver/mysql
- encoding/json

04

RESTful

Router

```
r := mux.NewRouter()


r.HandleFunc("/user/{user_id}", getOneUser)
  .Methods("GET")
```

JSON —> Struct

```
body := requestBody{}
err := json.NewDecoder(r.Body).Decode(&body)
```

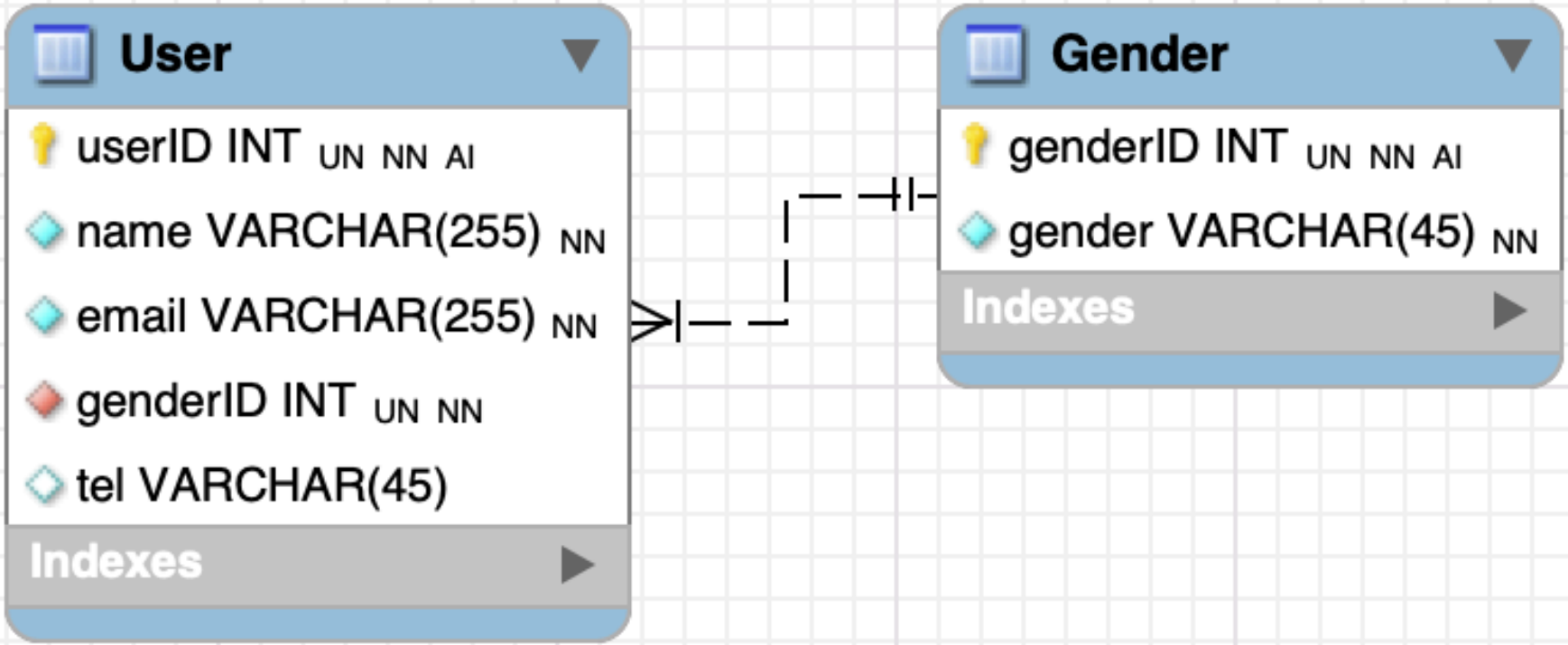
Get param in path

```
args := mux.Vars(r)
userID, hasUserID := args["user_id"]
if !hasUserID {
    // response error
    return
}
```

05

Database

Database EER Diagram



Query

```
qry := “SELECT * FROM `User` WHERE `User`.`userID`=?”

rows, err := db.Query(qry, userID)
if err != nil {
    // response error
    return
}
defer rows.Close()
```

Next & Scan

```
if rows.Next() {
    u := User{}
    err := rows.Scan(&u.UserID, &u.Name, &u.Email)
    if err != nil {
        // response error
        return
    }

    // Do something
}
```

Insert

```
sql := "INSERT INTO `User`(`name`, `email`) VALUES(?, ?)"

stmt, err1 := db.Prepare(sql)
if err1 != nil {
}
defer stmt.Close()

res, err2 := stmt.Exec(newUser.Name, newUser.Email)
if err2 != nil {
}

userID, err3 := res.LastInsertId()
if err3 != nil {
}
```

Update

```
sql := "UPDATE `User` SET `name`=? WHERE `userID`=?"

stmt, err2 := db.Prepare(sql)
if err2 != nil {
}
defer stmt.Close()

res, err3 := stmt.Exec(newUser.Name, userID)
if err3 != nil {
}

nRows, err4 := res.RowsAffected()
if err4 != nil {
}
```

Let’s code