# Creating a Simple Web Application using Golang

Steffi Indrayani/13514063[1]
*Computer Science/Informatics*
*School of Electrical Engineering and Informatics*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13514063@std.stei.itb.ac.id*

***Abstract*—Go Programming Language, which is often referred as Golang, is an open source programming language that was created by Google. With its rapid development and well reception from web developers across the world, this programming language is worth to try for. Using Golang for building a web application can be helpful because of its richness of libraries and good documentations. Basic codes in Golang and a simple web application that consists of basic pages using Golang are implemented and documented here. The result shows that Golang is proved to be one of the best programming language that developers can use for creating a web application.**

***Keywords*—Golang, Web Application, Web Server, Packages**

## I. Introduction

A good programming language is need for developing any forms of software, such as web application. Sometimes people prefer to use web application than local application when it comes to "no-installation-needed" and "no-upgrade-needed". Web application can be used as long as there is an internet connection. Creating web application can be tricky because it will be accessed by so many users and developers have to think about security and concurrency.

Google, one of the biggest technology companies in the world, decided to develop a new programming language back in 2007. It was created to have an open development model so that everybody can contribute to develop it further. Go is a C-like language, but more sophisticated. It does not force to do procedural programming like C or object-oriented programming like JAVA. Go gives developer freedom to build program in any paradigms.

Because of Golang's creator, Google is proven to be one of the most established technology companies, Golang is trusted by many developers. However, that fact does not instantly make Golang a good programming language. It has to meet criteria that are mentioned before.

The reasons why author decided to explore this programming language was because of its popularity that is getting higher and reviews that is pretty good for new programming language. Many people have posted their creation using Golang. Documentation and tutorial are well spread in the internet as well. This paper will discuss more about how to use Golang for dummies and the implementation of Golang in simple web application. It will refer mostly to the official website of Golang (https://golang.org/) and readers can learn from it by themselves.

## II. Golang Basic Programming

Before implementing Golang to create an application, readers should first learn the basic programming of Golang, especially for readers who never use Golang as programming language. I wrote this part when I was still exploring about Golang. This part will be written with numbering according to what is discussed.

1. Basic structure of program in Golang

First simple stuff to learn is the structure of the program in Golang itself. It is basically the same as C or C++, but with totally different syntax. Basic structure of program in Golang must be made up of at least a package. Here is the example of program in Golang.

```
package main

import (
      "fmt"
)

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

It can be clearly seen that a main program in Golang has to use package main. It can import any other packages and it has to have the function main. `func` basically the syntax to declare any functions or procedures.

2. Declaring Procedure and Function

```
func multiply (x int, y int) int {
    return x * y
}

func swap (x, y int) (int, int) {
    return y, x
}

func split(sum int) (x, y int) {
      x = sum / 10
      y = sum % 10
      return
}
```

The above program shows four different functions with for different way of declarations. Basically the format of declaring function is `func` *`name of function`* `(`*`input parameters`*`)` *`type of output parameters`*`.`

The first program is a normal function declaration with each of input parameters has each type of variable. If the type of all the parameters is the same, just omit all of the type but one. The second one also shows more than two output parameters. The third one shows that developers can give names to outputs parameters. Types and Declaring Variables

3. Types and Declaring Variables

Basic types that are widely used are `bool, string, int, uint, byte, rune, float64, complex64`. Type casting is used in Golang. There are several ways of declaring variables. Here are some examples.

```
var a                       d := 1.5
var b bool                  e := int(d)
var c int = 1               const Pi = 3.14
```

`var` is used to declaring a variable both inside and outside a function. If there is no type, then the variable will have the type when it is assigned a value. If there is a type but no assigned value, then it will have default value based on the type. `:=` is a different way of declaring variables without declaring the type. It will have type casting, but it cannot be written inside `var` and `func`. *`type(variable)`* is the way of converting type. `const` is the way of declaring constant variable.

4. Looping

Basically all of looping in Golang is using `for`, even `while-do` kind of looping is written using `for`. The program below will show the `for` kind of looping and `while-do` kind of looping.

```
for i := 1; i < 10; i++ {
    fac *= i
}

for i < 10 {
      fac += i
      i++
}
```

5. Conditional

Conditional statement is basically written in `if-else` and `switch` statement. The `if-else` statement is basically the same with statement in C or C++, but does not need bracket in the condition. Declaration inside the `if` and `else` statement is possible although it only applies inside `the if-then-else` statement only.

```
if v := math.Sqrt(x, n); v < lim {
     return v
} else {
     return lim
}
```

6. Pointers

Go has pointers that hold the memory address of a value. `*p` is pointer to `p` value. `&v` generates a pointer to its operand.

7. Type Structure

```
type Point struct {         P := Point(1,2)
      X int                 fmt.Println(P.X)
      Y int
}
```

Above program is an example how to declare a new type structure. Point is the name of type, `X` and `Y` are called field. `Point(1,2)` is how to declare variable with type `Point` and assigned value for each fields. Dot is used for calling the value of a field.

8. Array and Slice

```
var T [10]int               Capacity := cap(T)
var arr []int               Length := len(T)
a := make([]int, 5)
```

The first statement is the way of declaring an array with certain of type and fixed size. The second and third statements are declaration of an array that is not having fixed size and it is called slice. `cap` is used for counting the capacity of an array or slice, while `len` is for counting the actual length. `nil` is a slice that has zero capacity and length.

9. Maps

Maps are used to map keys to values. `m = make(map[string]Vertex)` is a way to make keys in string as a map for certain vertex. Key is like an index of an array but does not need to be in sequence.

10. Interface

Interface type is a set of method signatures which can hold any value that implements those methods. Below is how to declare an interface.

```
type I interface {
     fn()
}
```

11. Concurrency

Golang provide concurrency features as part of basic language. `goroutine` is as a way of declaring thread. Just write `go` *`name of function`* and it will create thread when running that function. It also have channel to send and receive block until the other side is ready. This allows `goroutines` to synchronize without explicit lock [1].

## III. Building the Web Application using Golang

After finishing the exploration about basic programming in Golang, author started to implement it right away on making a project. Author decided to build a web application because it would be more applicable and useful. The web application that author made is a three-page web that consists of Login Page, Register Page and Home Page. The interface of the web application was referred to Aigars Silkalns and modified as needed (n.d.) [2].

In Building Web Application using Golang, there are several things that are important. Just to remind readers that this Web Application is not using any frameworks. The first thing is to build template for those three pages. As was mentioned above, the templates or the front-end parts were referred to Aigars Silkalns. The templates were saved in home.html, register.html and login.html. They were the same as usual html files. To call variables or methods from Golang, simply use `{{` and `}}` between the Golang statements.

After creating template, templates need to be rendered inside a Golang program which acts as controller. The Golang program also will handle request, connect to database and give response value. The template rendering

is to read what inside a html file so that handler can show it anytime it is needed. This is the template rendering code.

```
var templates = template.Must(template.ParseFiles("register.html", "login.html"))

func renderTemplate(w http.ResponseWriter, tmpl string, p *Page) {
	err := templates.ExecuteTemplate(w, tmpl+".html", p)
	if err != nil {
		http.Error(w, err.Error(), http.StatusInternalServerError)
	}
}
```

The templates have been rendered and now it is time to make the handler for each page. The handler has duty to validate the path from the user input in the browser. If the path is right, then the handler can run other things, which one of it is calling the template rendering and showing the template. Author assumed that other than form submission, the method is GET.

```
//Inside the handler
if r.Method == "GET" {
  p, err := loadPage(title)
	if err != nil {
		p = &Page{Title: title}
	}
  renderTemplate(w, "home", p) //show the interface
}
```

Because both login and register page basically provide form submission, the handler must manage how to receive data from method POST and connect, retrieve from, and modify database. The three handlers of course manage different things based on the task of the page for method post.

1. Login Handler
   If the method is post, handler then connect with database and retrieve data to check whether the input username and password are registered. If they are, then the handler redirects to home page.
2. Register Handler
   If the method is post, handler then connect with database and insert the registration data into the database. If registration succeeds, the handler redirects to home page.
3. Home Handler
   This handler is not connected with any submission so it does not deal with method POST.

To connect with database, developers need to import "`database/sql`" and "`github.com/go-sql-driver/mysql`" (Creative Commons, n.d.) [3]. Author chose to use `mysql` as a driver. Here is how to connect with, retrieve from, and insert into database.

```
//connect DB
db, err := sql.Open("mysql",

"root:1234@tcp(127.0.0.1:3306)/golang")

//select query
rows, err := db.Query("SELECT id FROM USER WHERE username=? AND password=?", username, password)

//insert query
stmt, es := db.Prepare("INSERT INTO USER(name,username, password, email) VALUES (?,?,?,?)")
 _, er := stmt.Exec(name, username, password, email)
```

The three handlers should be called at `http.HandlerFunc` which cannot call the handlers that have been made because of the different type, so that is why there is a function called `makeHandler` that will receive function as parameter and return as a function as well. The validation of path that is in handler now is handled by this function.

The handlers are basically part of building a web server using package "`net/http`" so there is no need to worry about connecting to web server software like apache. Developers only need to call `http.HandleFunc` for each handler and `http.ListenAndServe` to a specific port.

To run this whole program, just open command prompt and then type this command bellow (Google, n.d.) [4].

```
$ go build wiki.go
$ wiki
```

Web server will run. Users need only to open `localhost:8080/login/MyFirstGolang` and then it will redirect to the login page.

So that is it to make a simple web application. The complete code of the program can be seen in this link: https://github.com/steffiindrayani/MyFirstGolang.

## IV. Result and Discussion

After implementing Golang and HTML in building a web application, the results are three different pages with different tasks. Here are the details of result in each page.

1. Register Page

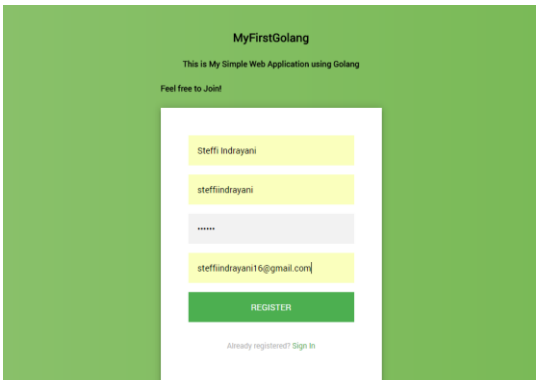


**Figure 1 Register Page**

The register page consists of form submission to receive the information of a user. The inputs that are given are saved in database. The page will be redirected to homepage. If user has already had account, then they should simply click Sign In. It will be redirected to Login Page. This can be run with localhost:8080/register/MyFirstGolang

2. Login Page

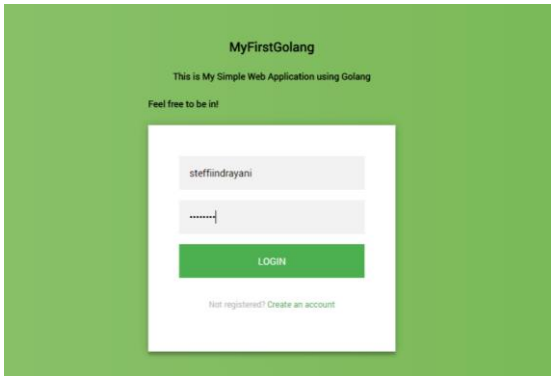


**Figure 2 Login Page**

The login page consists of form submission to give user access to enter home page. After user giving their username and password, the program will check whether they are registered or not. If they are, then it will be redirected to home page. If it is not then it will stay there. This can be run with `localhost:8080/login/MyFirstGolang`

3. Home Page


**Figure 3 Home Page**

After succeeding at logging in or registering, user will be redirected to a page where there is name and email based on the account user signing in. If user wants to go back to login page, simply click Log out.

After learning Golang and building a simple web application using Golang, author felt that Golang is a very hard language to learn. Although Golang seems similar to C or C++, author thought that they are very different in term of syntax. Golang provides much more way of declaring things and so many libraries that make programming easier.

In terms of concurrency, Golang is one of the language that support this. According to Mariana(2016), "concurrency (in Golang) is happening with the static execution speed of C or C++. This is not possible in many back-end languages."

Golang is definitely not an easy language to learn, but once developers master, it should be much easier. Building web application with Golang probably can be easier than using PHP, especially in building web server. Unfortunately, the code for accessing database is a bit tricky. Overall, Golang is hard, but worth trying for.

## V. Conclusion

Golang is a programming language that is developed by Google and although it is not as popular as other programming language, it is certainly developing its market. Building web application with Golang can be considered, especially for its concurrency and ease of making web server.

## VI. Acknowledgment

I would like to thank the lecturers of IF3280 Socio-Informatics and Professionalism course, Dr.Ir. Rinaldi Munir, MT., Dr. Eng. Ayu Purwarianti, ST.,MT. and Dr. Dessi Puji Lestari, who encouraged author to learning new things instead of what author had learned. This gave chance for author to learn Golang as it is what author would be used in author's internship period.

## References

[1] Google. *A Tour of Go*. Retrieved May 1, 2017, from https://tour.golang.org/

[2] Silkalns, Aigar. *Flat HTML5CSS3 Form*. Retrieved May 2, 2017 from https://codepen.io/colorlib/pen/rxddKy

[3] Creative Commons. *Importing a Database Driver*. Retrieved May 3, 2017 from http://go-database-sql.org/

[4] Google. *Writing Web Application*. Retrieved May 2, 2017 from https://golang.org/doc/articles/wiki/

[5] Mariana. (2016). *9 Reasons to Choose Golang for your Next Web Application*. Retrieved May 4, 2017 from

## Statement

This statement shows that I agree that what I wrote is my own writing, not plagiarism or translation from others paper.

Bandung, 5 May 2017

Signature

Steffi Indrayani/13514063