Writing a Simple Web Application

This lesson provides a program, and detailed description to design a web application, printing response in the form of HTML.

WE'LL COVER THE FOLLOWING

- A web application

A web application

In the following program from **line** 7 to **line 10**, we see how the HTML, needed for our web form (which is a simple input form with text box and submit button) is stored in a multi-line string constant form.

The program starts a webserver on port **3000** at **line 39**. It uses a combined `if` statement so that, whenever an error occurs, the web server stops with a `panic` statement (see **line 40**). Before starting a webserver, we see *two* so-called *routing statements*:

- `http.HandleFunc("/test1", SimpleServer)` at **line 37**
- `http.HandleFunc("/test2", FormServer)` at **line 38**

This means that whenever the URL ends with `/test1`, the webserver will execute the function `SimpleServer`, and the same for `/test2` with `FormServer`.

Now, look at the header of `SimpleServer()` at **line 13**. It outputs a hello world string in the browser by writing the corresponding HTML string to `io` with the `WriteString` method at **line 14**.

Now, look at the header of `FormServer()` at **line 19**. The browser can request _two different HTML methods: `GET` and `POST`. The code for `FormServer` uses a switch (starting at **line 22**) to distinguish between the 2 possibilities. If this URL is requested by the browser initially, then the request has a `GET` method, and the response is the constant form, as stated at **line 25**. When entering something in the text box and clicking the button, a `POST` request is issued. In the `POST` case, the content of the text box with a name in it is retrieved with the `request.FormValue("in")`, as you can see at **line 32**, and written back to the browser page.

Start the program in a console and open a browser with the URL https://1dkne4jl5mmmm.educative.run/test2 (in your case the URL will be different) to test this program:

> **Remark**: Change **line 39** to `if err := http.ListenAndServe(":8088", nil); err != nil {` if you're running the server locally.

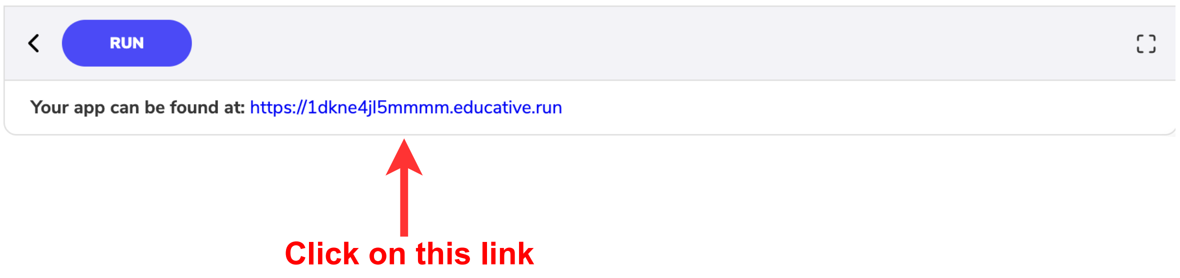
Environment Variables

Key:	Value:
GOROOT	/usr/local/go
GOPATH	//root/usr/local/go/src
PATH	//root/usr/local/go/src/bin:/usr/local/go...

```
package main
import (
"net/http"
"io"
)

const form = `<html><body><form action="#" method="post" name="bar">
<input type="text" name="in"/>
<input type="submit" value="Submit"/>
</form></html></body>`

/* handle a simple get request */
func SimpleServer(w http.ResponseWriter, request *http.Request) {
  io.WriteString(w, "<h1>hello, world</h1>")
}

/* handle a form, both the GET which displays the form
and the POST which processes it.*/
func FormServer(w http.ResponseWriter, request *http.Request) {
  w.Header().Set("Content-Type", "text/html")

  switch request.Method {
    case "GET":
      /* display the form to the user */
      io.WriteString(w, form );
    case "POST":
      /* handle the form data, note that ParseForm must
      be called before we can extract form data with Form */
      // request.ParseForm();
      //io.WriteString(w, request.Form["in"][0])
      // easier method:
```

```
      io.WriteString(w, request.FormValue("in"))
  }
}

func main() {
  http.HandleFunc("/test1", SimpleServer)
  http.HandleFunc("/test2", FormServer)
  if err := http.ListenAndServe("0.0.0.0:3000", nil); err != nil {
    panic(err)
  }
}
```

Click the **Run** button, and wait for the terminal to start. Once it starts, perform the following steps:

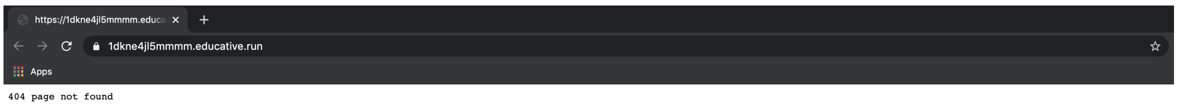


Your app can be found at: https://1dkne4jl5mmmm.educative.run

Step 1

1 of 6

Browser opens a page

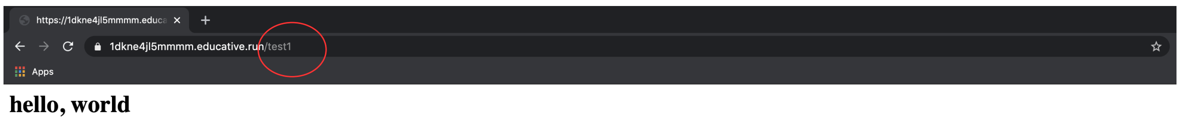


```
404 page not found
```

Step 1

2 of 6

Add "/test1" at the end of URL and reload the browser

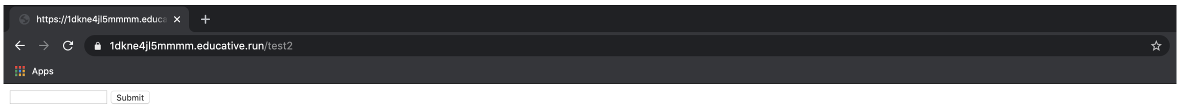

https://1dkne4jl5mmmm.educa
1dkne4jl5mmmm.educative.run/test1
Apps

hello, world

Step 2

3 of 6

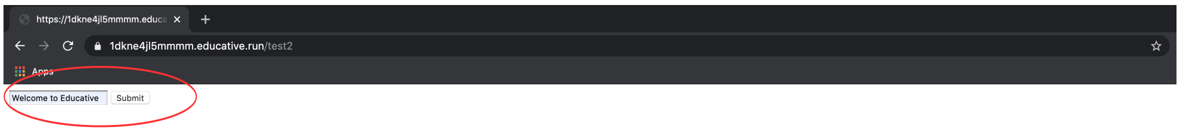
Add "/test2" at the end of URL and reload the browser


https://1dkne4jl5mmmm.educa
1dkne4jl5mmmm.educative.run/test2
Apps
Submit

Step 3

4 of 6

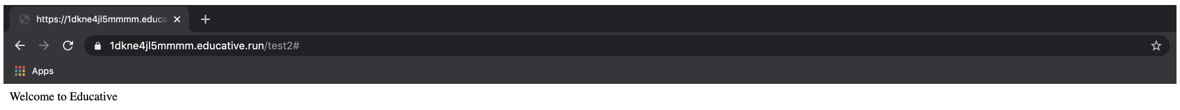
Type anything in the text box and press **Submit** button


https://1dkne4jl5mmmm.educa
1dkne4jl5mmmm.educative.run/test2
Apps
Welcome to Educative
Submit

Step 4

5 of 6

Server prints the text on the screen


https://1dkne4jl5mmmm.educa
1dkne4jl5mmmm.educative.run/test2#
Apps

Welcome to Educative

Step 4

6 of 6

> **Remark:** If you’re running the program locally, try URLs http://localhost:8088/test1 and http://localhost:8088/test2.

When using constant strings, which represent html-text it is important to include the

```
<html><body>... </html></body>
```

to let the browser know that it receives html. Even safer is to set the header with the content-type text/html before writing the response in the handler:

```
w.Header().Set("Content-Type", "text/html")
```

The content-type the browser thinks it can be retrieved with the function:

```
http.DetectContentType([]byte(form))
```

Now that you know how to write a simple web application, the next lesson brings you a related challenge for you to solve.