HTTP Server

Introduction

In this example you will learn how to create a basic HTTP server in Go. First, let's talk about what our HTTP server should be capable of. A basic HTTP server has a few key jobs to take care of.
Process dynamic requests: Process incoming requests from users who browse the website, log into their accounts or post images.
Serve static assets: Serve JavaScript, CSS and images to browsers to create a dynamic experience for the user.
Accept connections: The HTTP Server must listen on a specific port to be able to accept connections from the internet.

Process dynamic requests

The `net/http` package contains all utilities needed to accept requests and handle them dynamically. We can register a new handler with the `http.HandleFunc` function. It's first

parameter takes a path to match and a function to execute as a second. In this example: When someone browses your websites (`http://example.com/`), he or she will be greeted with a nice message.

```
http.HandleFunc("/", func (w http.ResponseWriter, r *http.Request) {
    fmt.Fprint(w, "Welcome to my website!")
})
```

For the dynamic aspect, the `http.Request` contains all information about the request and it's parameters. You can read GET parameters with `r.URL.Query().Get("token")` or POST parameters (fields from an HTML form) with `r.FormValue("email")` .

Serving static assets

To serve static assets like JavaScript, CSS and images, we use the inbuilt `http.FileServer` and point it to a url path. For the file server to work properly it needs to know, where to serve files from. We can do this like so:

```
fs := http.FileServer(http.Dir("static/"))
```

Once our file server is in place, we just need to point a url path at it, just like we did with the dynamic requests. One thing to note: In order to serve files correctly, we need to strip away a part of the url path. Usually this is the name of the directory our files live in.

```
http.Handle("/static/", http.StripPrefix("/static/", fs))
```

Accept connections

The last thing to finish off our basic HTTP server is, to listen on a port to accept connections from the internet. As you can guess, Go has also an inbuilt HTTP server, we can start faily quickly. Once started, you can view your HTTP server in your browser.

```
http.ListenAndServe(":80", nil)
```

The Code (for copy/paste)

This is the complete code that you can use to try out the things you've learned in this example.

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", func (w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Welcome to my website!")
    })

    fs := http.FileServer(http.Dir("static/"))
    http.Handle("/static/", http.StripPrefix("/static/", fs))

    http.ListenAndServe(":80", nil)
}
```