Hello World

Introduction

Go is a battery included programming language and has a webserver already built in. The `net/http` package from the standard library contains all functionalities about the HTTP protocol. This includes (among many other things) an HTTP client and an HTTP server. In this example you will figure out how simple it is, to create a webserver that you can view in your browser.

Registering a Request Handler

First, create a Handler which receives all incomming HTTP connections from browsers, HTTP clients or API requests. A handler in Go is a function with this signature:

```
func (w http.ResponseWriter, r *http.Request)
```

The function receives two parameters:

An `http.ResponseWriter` which is where you write your text/html response to.

An `http.Request` which contains all information about this HTTP request including things like the URL or header fields.

Registering a request handler to the default HTTP Server is as simple as this:

```
http.HandleFunc("/", func (w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
})
```

Listen for HTTP Connections

The request handler alone can not accept any HTTP connections from the outside. An HTTP server has to listen on a port to pass connections on to the request handler.

Because port 80 is in most cases the default port for HTTP traffic, this server will also listen on it.

The following code will start Go's default HTTP server and listen for connections on port 80. You can navigate your browser to `http://localhost/` and see your server handing your request.

```
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"
)

```

```go
func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
    })

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