Embed Modules

Embed Module

This module is a super duper cool module in the sense that you are able to pack static files into your binary, rather than having your program relying on the actual file being present in your filesystem when it is ran.

Normally when you do deployment with a Go server, you would have the server binary, the static frontend files that is open and serve by the Go server. You can pack everything into a Docker image so the server that's running your web application can just spin up the container which will have everything. However, in Golang 1.16 you can directly embed your static frontend files into the binary so you do not need a docker image to pack your static files in. You can just embed it directly into your binary.

Embed directives

To get started with using the Embed module you would actually use the embed directives which instructs compiler at compile time what to do.

For example:

```
//go:embed $PATH
var content $WANTED_DATA_TYPE
```

1. `$PATH` is the file or directory that you want to include. If you use dataa type `embed.FS` then you can put multiple files or multiple directories
2. `$WANTED_DATA_TYPE` needs to be replace with one of the followings
    1. `string` = Accepts a single file and when the binary is built, it wil lread the content of that entire file into the variable as a string. If you use this you can only embed in ONE file. No multiple files or a directory
    2. `[]byte` = Same as String, but it will be read in as `[]byte`
    3. `embed.FS` = Using this data type you are allowed to embed multiple directories and even files. This struct implements the `io/FS` interface, which means that `net/http` package can use it as part of the handler

`embed.FS` after you embed the files into the binary, you can open files that are included as part of the binary. Even if the files / directory doesn't exist after the program is ran. Magic. Power of embedded files.

Note about `embed.FS`

Do keep in mind that the directory that you embed, the variable that it is assigned under refer to the root directory that contains the directory or file. For example:

```
//go:embed static/
var embeddedFS embed.FS
```

The variable `embeddedFS` refers to a "root directory" that contains the "static" directory (I put it in quotes because it is in the binary not in the actual OS filesystem

Thus if you want to say open a file you need to do:

```
embeddedFS.Open("static/index.html")
```

And not this:

```
embeddedFS.Open("index.html")
```

In terms of `net/http` package, that means the files will be serve the subdirectory that you embed as. For example, if you embed `static/index.html`, and you add a handler `"/"` that references the particular `static/index.html` file, you will have to visit `localhost:80/static/index.html` or `localhost:80/static` in order to see the file rather than just `localhost:80`.

To solve that problem you would need use `fs.Sub` which:

```
Sub returns an FS corresponding to the subtree rooted at fsys's dir.
```

Basically returns you a filesystem that is rooted at the provided filesystem, making an "alias" per se.

Revision #4
Created 2023-05-29 21:40:09 UTC by Tamarine
Updated 2023-05-30 01:26:10 UTC by Tamarine