Solution Review: Web Application for Serving Guests

This lesson discusses the solution to the challenge given in the previous lesson.

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 (
        "fmt"
        "html/template"
        "net/http"
)

const port = 3000
var guestList []string

func main() {
        http.HandleFunc("/", indexHandler)
        http.HandleFunc("/add", addHandler)
        http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
}

// indexHandler serves the main page
func indexHandler(w http.ResponseWriter, req *http.Request) {
        t := template.New("index.html")
        t, err := t.Parse(indexHTML)
        if err != nil {
                message := fmt.Sprintf("bad template: %s", err)
                http.Error(w, message, http.StatusInternalServerError)
        }
        // the HTML output is now safe against code injection
        t.Execute(w, guestList)
}

// addHandler add a name to the names list
func addHandler(w http.ResponseWriter, req *http.Request) {
        guest := req.FormValue("name")
        if len(guest) > 0 {
                guestList = append(guestList, guest)
        }
        http.Redirect(w, req, "/", http.StatusFound)
}

var indexHTML = `
```

```
var indexHTML =
<!DOCTYPE html>
<html>
    <head>
                <title>Guest Book ::Web GUI</title>
    </head>
    <body>
                <h1>Guest Book :: Web GUI</h1>
                <form action="/add" method="post">
                Name: <input name="name" /><submit value="Sign Guest Book">
                </form>
                <hr />
                <h4>Previous Guests</h4>
                <ul>
                        {{range .}}
                        <li>{{.}}</li>
                        {{end}}
                </ul>
        </body>
</html>
`
```

In the code above, The HTML for the web form is contained in the variable `indexHTML`, defined from **line 38** to **line 58**. It contains an `input` field name at **line 47**. The list starting at **line 51** ranges over the current values, showing them as list items.

The web server is started at **line 14** on port with the `ListenAndServe` method. There are *two* routing handlers:

- `indexHandler` for *root* requests (`/`)
- `addHandler` for requests of the form `/add`

Now, look at the header of the `indexHandler()` function at **line 18**. It defines a new template `t` at **line 19**, and parses it at **line 20**. If there is a parsing error, this is displayed in the browser from **line 21** to **line 24**. Otherwise, the template is executed, writing its output to the browser, and using `guestList` (defined at **line 9**) as the current variable to iterate over.

Now, look at the header of `addHandler()` function at **line 30**. At **line 31**, it reads the **name** input value with the `FormValue` method into `guest`. If `guest` is not empty, it is appended to `guestList` at **line 33**. Then, we simulate a `/` request by calling `Redirect` at **line 35**. This will invoke the `indexHandler`, which will show the new `guestList` in the browser.

That is it for the solution. In the next lesson, we'll discuss how to make an elaborated webserver.