Francesco Levorin

Alkaline 1.1

Brief introduction

Alkaline is a lightweight module made to replace old server-side languages using the power and the simplicity of Python.

Basic concept

So, I will start with an example: I want to display the string “Hello World” in my HTML document. There are two methods: command-line and python. In the first example we will use the command-line option, made for debugging. In the root folder, you have to create the file template.html (assuming that you already know HTML). In template.html you can write all you want, remembering to close all the tags due to XML package problems, but in this case write in the file:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <p>$hello$</p>
</body>
</html>
```

This is all HTML markup code, except for $hello$, in fact that is a variable. Variables in Alkaline is referenced in HTML with at the beginning and end with ‘$’ and between the variable name. But if we compile this, it will prompt an error, so what do we have to do to display the string “Hello World”? It’s very simple.
We’re gonna add inside <head>, <body> or <html> a tag called <python> like so:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <p>$hello$</p>
</body>
<python>
self.vars[“hello”] = “Hello World”
</python>
</html>
```

I told you that it was simple! But inside <python> it assign to dict key “hello” the value “Hello World”, but what does it mean? The Python code inside <python> is executed before the effective compiling, so tells what to do before the variables been replaced. In this case tells that the variable “hello” contains the value “Hello World” and, consecutively, the compiler will replace the keyword $hello$ with Hello World. I hope I was clear.

To assign to a variable an HTML tag, we can type:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    $hello$
</body>
<python>
self.vars[“hello”] = tag(“p”, “Hello World”)
</python>
</html>
```

The tag function is mandatory only in templates, to avoid read issues. The working scheme of tag function is this:

```
tag([html tag], [inner text], [attributes in dict])
```

Tag function returns a string, for example in the Hello World example above the result will be: " Hello World ". The Hello World example without and with tag function will produce the same result.
To compile this template, open a terminal inside root folder and type:

```
python3 -m alkaline template.html
```

Basically we called the Alkaline module and passed the file to compile as argument. After we pressed enter, will appear the Alkaline logo and two self-explanatory sentences: rendering and done. This script will automatically detect changes and compile the file index.html. If we open index.html we'll notice that, as expected, the segment $hello$ changed to his value, "Hello World". Furthermore python tag has been eliminated, guaranteeing more security in other applications.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <p>Hello World</p>
</body>
</html>
```

Inside <python> we can type any sort of Python code.

Import and use Alkaline in Python code

First of all, create a new Python file, the name is not important. In this file write down:

```
from alkaline import Engine

eng = Engine("template.html")
print(eng.compile({"hello": "Hello World!"}))
```

This piece of code will import Engine from alkaline and initialize it for the template specified inside brackets. The successive line a print function prints out the result of compile, that is in string format. Inside brackets we optionally can specify manually the value of vars in dict type. If you want to write the results in a file you can type: "open('index.html', 'w').write(eng.compile())", it'll produce the same result of the first chapter example.

Congratulations! Now you are good to go, I have transmitted you all the fundamental instructions to work with Alkaline. Now it's your responsibility to use it with your creativity! Good coding!