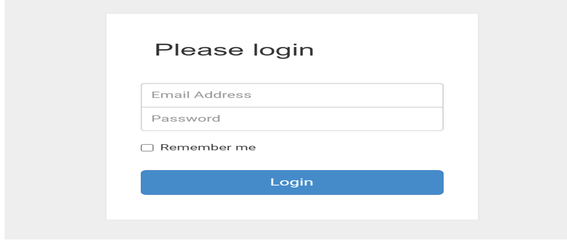

Guión de Prácticas 3: Plantillas, Manejo de Sesiones y Frameworks CSS

Sergio Alonso (zerjioi@ugr.es) y José María Guirao (jmguirao@ugr.es)

Entrega: 15- noviembre

Resumen:

En esta práctica avanzaremos en el uso de `Flask`: usaremos un motor de plantillas (templates) `Jinja` incorporado en Flask, haremos uso de un framework CSS para que el HTML sea adaptable (responsive) utilizaremos sesiones para gestionar la identificación de usuarios en la web y otros posibles datos de sesión de nuestra aplicación web.

Plantillas (templates)

Cuando desarrollamos una aplicación web no es buena idea incluir el código `HTML` de las páginas dentro de nuestra aplicación Python. Utilizando plantillas (templates) conseguiremos separar correctamente el aspecto de la aplicación (vista) de su lógica interna (controlador) (ver Wikipedia: Modelo-Vista-Controlador).

Flask incorpora un motor de plantillas para generar el HTML: `Jinja2`, muy parecido a twig. En las transparencias de clase o en Flask Tutorial: Templates hay información de sobre ellas. No olvidarse de la herencia de plantillas a la hora de usarlas.

Frameworks CSS

Usar una librería de CSS Framework nos va a permitir un diseño consistente, adaptable al tamaño de pantalla del cliente, y mas estandarizado. Bootstrap es el mas utilizado y el mas sencillo, aunque se puede optar por cualquier otro.

Maquetación

Haremos un sitio web en el que haya (al menos) una barra de navegación, un menú vertical, un espacio para mostrar contenidos, y un pie de página. La barra de navegación contendrá el de logo del sitio, el nombre del mismo, enlaces a otras páginas y un mini-formulario para el login, o un enlace para el mismo.

Para ello podemos empezar por la plantilla inicial de de boostrap, añadirle una barra de navegación como en Sticky footer with fixed navbar (inspeccionar el código fuente de esta página para ver como lo hace). En Bootstrap Navigation Bar, y en la docuementación de boostrap hay maś información sobre las barras de navegación.

A continuación añadiremos el menu vertical a la izquierda y el espacio para los contenidos. Se debe hacer uso del sistema de grid y de las utilidades para el espaciado y margenes de bootstrap, o del framework css que hayamos elegido.

Contenido

La aplicación tendrá una funcionalidad de login, con un espacio destinado a ello en la barra de navegación, de manera que cuando el usuario no esté logeado, se le dé la posibilidad de hacerlo o de registrarse, y si ya lo ha hecho, muestre un mensaje de bienvenida y un enlace para deslogearse. En Message Flashing hay un bosquejo de como hacerlo.

En la barra de navegación habrá varios enlaces, uno de los cuales dará acceso a una interfaz del ejercicio 1 de la primera práctica. Para esto habrá que usar los formularios de bootstrap.

En el menú de la derecha aparecerán enlaces a las tres últimas páginas visitadas en su orden.

Para Nota Poner en los enlaces de la barra de navegación, el interface a todos los ejercios de la primera práctica con un menu dropdown.

Manejo de sesiones

En toda aplicación web es necesario gestionar información que se mantenga entre las distintas páginas que visita el usuario. Para ello se hace uso de las sesiones.

Persistencia

Para almacenar la información de nuestra aplicación podemos utilizar distintos mecanismos de persistencia. Cuando la aplicación sea suficientemente sencilla podemos utilizar algún esquema de almacenamiento local, como por ejemplo la biblioteca `pickleshare`.

En esta práctica usaremos dicha base de datos para almacenar la información del usuario que se introduzca en un formulario de registro en nuestra aplicación web. Crearemos además una página nueva de nuestra aplicación que nos permita visualizar los datos del usuario (tendremos acceso a través del menú a esta página cuando estemos identificados en la web). Asimismo crearemos una nueva página que le permita al usuario cambiar los datos personales (volviendo a mandar el formulario, que aparecerá relleno con los datos anteriores).

Para instalar `pickleshare` tendremos que añadirla como dependencia en nuestro fichero de **requirements.txt** (y reconstruir nuestro contenedor para que incluya esta librería).

Modelo-Vista-Controlador

Debemos intentar seguir el paradigma *modelo, vista, controlador* (MVC) lo más fielmente posible, de tal manera que sea posible, por ejemplo, cambiar radicalmente el aspecto de la aplicación web modificando únicamente los templates, o cambiar la Base de Datos, sin afectar al resto del código. Los archivos de `HTML` estarán, por tanto, en un directorio **templates** separado y lo relacionado con la persistencia estará en un arhivo (módulo de python) aparte: **model.py**

Nota: Posible problema en el uso de `PickleShareDB`

En esta práctica se ha propuesto el uso de `PickleShareDB`, que es una biblioteca que permite de manera muy básica que almacena en disco un diccionario. Como habréis visto en el ejemplo de la práctica su manejo es extremadamente simple: simplemente al asignar el valor de una de los elementos del diccionario se guardarán dichos cambios en el disco para que en una posterior ejecución se pueda leer y usar dicha información. Sin embargo es posible que nos estemos enfrentando a un pequeño problema. Si ejecutamos:

```
db = PickleShareDB('miBD')
db['pepe'] = dict()
db['pepe']['pass'] = '1234'
```

Podremos comprobar al relanzar nuestra aplicación que `db['pepe']` es un diccionario vacio (y hemos perdido el valor de `'pass'`).

Este problema ocurre porque la biblioteca `PickleShareDB` no puede detectar cambios en objetos mutables que se usen su diccionario. Es decir, no se da cuenta que hemos cambiado el objeto diccionario que habíamos creado en la segunda línea del ejemplo anterior.

Posibles soluciones

Existen varias opciones que podemos usar para evita este problema. Por ejemplo:

```
db = PickleShareDB('miBD')
db['pepe'] = {'pass': '1234'}
```

En este caso estamos creando el diccionario y rellenándolo ANTES de hacer la asignación en la base de datos de `PickleShareDB`. Otra posible solución es forzar la grabación del objeto mutable despues de haberlo cambiado:

```
db = PickleShareDB('miBD')
db['pepe'] = dict()
db['pepe']['pass'] = '1234'+
db['pepe'] = db['pepe']
```

La última línea de este código parece redundante, pero en realidad lo que está consiguiendo con esa asignación al diccionario de `PickleShareDB` es que se active la grabación de la información en el disco.