Diseño de Aplicaciones para Internet (2014-2015)
Guión de Prácticas 3:
Formularios "Avanzados", Templates, Manejo de Sesiones y Persistencia

S. Alonso, J.M. Guirao
zerjioi@ugr.es, jmguirao@ugr.es

Resumen

En esta práctica avanzaremos en el uso de `web.py`: aprenderemos a usar las opciones de validación de formularios, usaremos plantillas (templates) para generar cómodamente nuestro código, utilizaremos sesiones para gestionar la identificación de usuarios en la web y otros posibles datos de sesión de nuestra aplicación web y, por último, veremos como gestionar la persistencia usando diversas bases de datos.

1. Formularios "Avanzados"

En la práctica anterior empezamos a manejar la sub-biblioteca `Form` de `web.py` para manejar formularios. Dicha biblioteca contiene opciones interesantes que pueden facilitarnos el introducir información en una página web, así como su posterior manejo. En este punto de la práctica vamos realizar un formulario algo más complejo que los vistos hasta ahora:

- El formulario debe preguntar los siguientes datos personales a los usuarios de la web: *nombre*, *apellidos*, *DNI*, *correo electrónico*, *fecha de nacimiento*, *dirección*, *contraseña*, *verificación de la contraseña*, *forma de pago preferida (contra reembolso o tarjeta VISA)*, *número de la tarjeta VISA*, *aceptación de las clausulas de protección de datos*, *botón de mandar*.
- Los campos del formulario deben ser del tipo:
  - Nombre, apellidos, correo electrónico, número de VISA: `textbox`.
  - Fecha de nacimiento: `dropdown` (x3, día, mes, año).
  - Dirección: `textarea`.
  - Contraseña y verificación: `password`.
  - Forma de pago: `radio`.
  - Aceptación clausulas: `checkbox`.
  - Botón: `button`.

- El formulario debe verificarse utilizando los mecanismos que nos ofrece la biblioteca (`validators`). Probablemente haga falta utilizar alguna función `lambda` de Python [1, 2]. Particularmente hay que asegurarse de que:
  - Ningún campo de texto esté vacío
  - El correo electrónico sea "válido" (¿recuerdas el ejercicio de expresiones regulares de la sesión 1?)
  - Que el número de la tarjeta VISA sea correcto (4 grupos de 4 dígitos separados por un espacio o `-`).
  - Que la fecha de nacimiento sea una fecha válida.
  - Que la contraseña y su validación coincidan y tengan más de 7 caracteres.
  - Que las clausulas se hayan aceptado.

Nota: La validación de un elemento tipo `checkbox` es un poco "especial" debido a como se implementa en HTML (las dificultades en su uso vienen heredadas de HTML, no es problema especial de `web.py`). Para no malgastar mucho tiempo buscando soluciones, se puede usar el siguiente validador (cortesía de E. Serrano):

```
form.Checkbox("accept_license",
    form.Validator("Acepta las clausulas", lambda i: i == 'true'),
    value='true'
)
```

2. Plantillas (templates)

Cuando desarrollamos una aplicación web (por ejemplo con `web.py`) no es buena idea incluir el código HTML de las páginas dentro de nuestra aplicación Python. Utilizando plantillas (templates) conseguiremos simplificar mucho todas las tareas repetitivas, así como separar correctamente el aspecto de la aplicación (vista) de su lógica interna (controlador) [3].

Una biblioteca potente de templates para Python es `Mako` [4, 5], utilizada en sitios masivos en producción como `http://reddit.com`. En esta práctica vamos a familiarizarnos con esta biblioteca. Para ello tendremos que construir varias páginas usando esta biblioteca.

Dichas páginas deberán mostrar un sitio web en el que haya (al menos) una cabecera, dos columnas y un pie de página. En la cabecera habrá una imagen de logo del sitio, el nombre del mismo, un subtítulo y un mini-formulario de `login`. En la columna izquierda habrá opciones (como por ejemplo menús), la columna de la derecha contendrá el cuerpo principal de la página y el pie contendrá información sobre el autor de la página y los derechos de la misma (licencia). En caso de no querer hacer el diseño de la página desde cero, podemos optar por descargar alguna plantilla web ya creada [6] y adaptarla para poder presentar los contenidos dinámicos de la aplicación.

Debemos intentar seguir el paradigma *modelo, vista, controlador* lo más fielmente posible, de tal manera que sea posible, por ejemplo, cambiar radicalmente el aspecto de la aplicación web modificando únicamente los templates.

Para instalar Mako en Ubuntu Server:

```
$ sudo apt-get install python-mako
```

3. 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 [7]. En `web.py` las sesiones se manejan mediante la clase `Session` [8].

En esta práctica adaptaremos nuestra página web para que el formulario de `login` anteriormente citado funcione correctamente y se muestre distinto contenido si el usuario está ya identificado o no (a esta alturas la comprobación de un usuario único estará codificada "a pelo" en la aplicación, en el siguiente apartado guardaremos distintos usuarios usando persistencia). Por ejemplo, el formulario de login de la cabecera solo debe aparecerle a los usuario no identificados, mientras que a los identificados debe mostrárseles el típico mensaje de *"Bienvenido xxxx"* y un enlace para hacer `logout`.

Además, implementaremos en nuestra aplicación (y utilizando exclusivamente sesiones) un "menu" que muestre -y permita acceder- a las últimas 3 páginas del sitio visitadas (lo podemos incluir en el menú de la izquierda).

4. Persistencia

4.1. Persistencia Sencilla: `dbm`

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 `dbm` [9] o `anydbm` [10].

En esta práctica usaremos dicha base de datos para almacenar la información del formulario de datos de usuario de la sección 1 una vez haya sido validada. Crearemos 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).

4.2. Persistencia NO-SQL: MongoDB

`MongoDB` [11] es una base de datos NO-SQL potente que nos permitirá almacenar cualquier información que nuestra aplicación web necesite.

Copia y modifica la aplicación de la sección 4.1 para usar la base de datos `MongoDB` usando la biblioteca `Pymongo` [12].

Nota: A partir de este práctica continuaremos trabajando con *esta* versión de la aplicación web.

Para instalar `MongoDB` y `Pymongo` en Ubuntu Server:

```
$ sudo apt-get install mongodb
$ sudo easy_install pymongo
```

Referencias

[1] Python: Lambda Functions: `http://www.secnetix.de/olli/Python/lambda_functions.hawk`

[2] Funciones lambda en Python: http://jolthgs.wordpress.com/2012/02/12/funciones-lambda-en-python/

[3] Modelo Vista Controlador: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[4] Mako: http://www.makotemplates.org/

[5] Documentación de Mako: http://docs.makotemplates.org/en/latest/

[6] Free Website Templates: http://www.freewebsitetemplates.com/

[7] Sesiones, autenticación y control de acceso: http://elpuig.xeill.net/Members/vcarceler/asix-m09/uf1/nf2/a5

[8] web.py session: http://webpy.org/sessions/

[9] Biblioteca dbm de Python: http://docs.python.org/2/library/dbm.html

[10] Biblioteca anydbm de Python: http://docs.python.org/2/library/anydbm.html

[11] Biblioteca MongoDB: http://www.mongodb.org/

[12] Gentle Introduction to MongoDB using Pymongo http://altons.github.io/python/2013/01/21/gentle-introduction-to-mongodb-using-pymongo/