Diseño de Aplicaciones para Internet (2018-2019)
Guión de Prácticas 2:
Micro Framework Web para Python: `Flask`

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

Resumen

`Flask` es un micro framework web para Python de uso sencillo pero suficientemente potente que permite desarrollar una aplicación web en poco tiempo.

En esta sesión trataremos de construir algunas pequeñas aplicaciones web usando las opciones más básicas de `Flask`. En futuras sesiones se practicarán otros conceptos como el manejo de sesiones, templates, el acceso a bases de datos, etc.

1. Aplicación Básica

En la página web oficial de `Flask` [1] podemos encontrar un ejemplo minimalista ("Hola Mundo") en el que se utiliza el framework para crear un aplicación web extremadamente sencilla que saluda al usuario. Copie dicho código, ejecútelo, compruebe que funciona e intente entender cada parte de dicho programa. Es posible que necesite consultar la API [2] o el "Libro de Recetas" [3] de la biblioteca.

Para instalar los paquetes necesarios en Ubuntu Server:

```
$ sudo apt-get update
$ sudo apt-get install python3-pip
$ pip3 install Flask
```

Nota: investigue como poner la aplicación en modo "debug" para facilitar el desarrollo (por ejemplo, evitando tener que relanzar la aplicación cada vez que el codigo cambie).

Nota 2: La aplicación Flask escuchará por defecto en el puerto `5000` y en el interfaz `loop` (`127.0.0.1`). Para poder cargar la aplicación desde fuera del `guest` se hace necesario hacr un `forward` de nuestra máquina `host` al `guest` cambiando la configuración de `Vagrantfile`:

```
config.vm.network "forwarded_port", guest: 5000, host: 8080
```

así como decirle a `Flask` que ponga la aplicación disponible en todos los interfaces (por ejemplo cambiando la instrucción `app.run` en nuestra aplicación:

```
app.run(host='0.0.0.0')
```

2. Sirviendo contenidos estáticos (imágenes, hojas de estilo, etc)

Averigüe el mecanismo más habitual que ofrece `Flask` para servir contenidos **estáticos** tales como imágenes u hojas de estilo. Añada algunas imágenes estáticas a su aplicación y compruebe que el cliente es capaz de acceder a ellas directamente a través de una URL.

Aunque el método habitual para servir páginas web de `Flask` es el uso de templates, modifique el ejemplo original del punto anterior para generar en vez de simplemente el código `¡Hola Mundo!`, generar código `HTML` correcto en el que se incluya, entre los demás elementos necesarios, una página de estilo `CSS` y alguna imagen estática.

3. Manejo de URLs

Averigüe el mecanismo de `Flask` para el análisis y manejo de distintas URLs. Cree una nueva aplicación para servir páginas distintas dependiendo de la URL introducida. Asimismo, sería conveniente ser capaces de obtener los parámetros de una llamada `GET`. Compruebe que puede utilizar variables en parte de la url (por ejemplo, mostrar contenidos distintos para las siguientes urls:

- `http://<ip:puerto>/user/pepe`
- `http://<ip:puerto>/user/zerjillo`
- `http://<ip:puerto>/user/[cualquierUsuario]`

Por último, defina una página para el caso en que una URL no esté definida (error `HTTP 404, not found` [4]).

4. Para Nota: Creando Imágenes Dinámicas [binarias]

A estas alturas debemos ser capaces de hacer un sitio web dinámico sencillo (probablemente no muy bonito hasta que no utilicemos templates). Pese a que muchísimos sitios dinámicos solo cambian su código HTML dependiendo de las entradas de los usuarios, en este último apartado vamos a ir un paso más allá: se creará contenido gráfico dinámico.

La idea de este ejercicio es crear una aplicación web dinámica que a partir de ciertos parámetros sea capaz de generar en directo una imagen fractal. Para esta tarea podemos reutilizar el código de los ejercicios de la primera sesión de prácticas (ejercicio sobre el fractal de Mandelbrot). La aplicación web debe contar con al menos estas características:

- Mediante parametros `GET` debemos poder definir al menos los siguientes parámetros para calcular el fractal: recuadro del plano complejo sobre el que se calculará el fractal (x_1, y_1) - (x_2, y_2) y anchura de la imagen resultante (en píxeles).

- Una vez obtenidos dichos datos debe calcularse y dibujarse el fractal. Se podrá usar alguna función similar a las del guión de prácticas 1 o bien usar las funciones del fichero `mandelbrot.py` (en SWAD).
- La imagen completamente creada debe mostrarse al usuario usando el formato PNG.

Adicionalmente, si se quiere mejorar la aplicación, se puede:

- Añadir algunos parámetros `GET` para cambiar la paleta de color a utilizar cuando se dibuje el fractal y el número máximo de iteraciones a ejecutar cuando se calcula el fractal.
- Implementar algún tipo de caché de la aplicación que evite recalcular el mismo fractal en caso de que se hagan dos peticiones idénticas (ahorrando ciclos de cómputo). Para conseguirlo, por ejemplo, se pueden guardar en disco los fractales con un nombre que identifiquen los parámetro utilizados para el cálculo. Cada vez que se solicite un nuevo fractal lo primero que realizará la aplicación será comprobar si el fichero con dichos parámetros ya ha sido creado. En caso afirmativo se servirá tal cual. En caso negativo, se realizarán los cálculos del fractal oportunos.
- Mejorar el sistema de caché propuesto para que las imágenes de más de un día se borren para evitar colapsar el disco duro del servidor en caso de que se soliciten muchas imágenes fractales.

Para instalar las bibliotecas necesarias para ejecutar las funciones de `mandelbrot.py`:

```
$ pip install Pillow
```

5. Para Nota 2: Creando Imágenes Dinámicas [Vectoriales]

Desarrolle una aplicación web sencilla que nos permita crear una imagen `SVG` [5] dinámica (que cambie cada vez que visitemos la página) y aleatoria. Por ejemplo, que cada vez que se visite la página dibuje elipses, rectángulos, etc. de colores y posiciones distintas.

Referencias

[1] Página oficial de `Flask`: `http://flask.pocoo.org/`

[2] API de `Flask`: `http://flask.pocoo.org/docs/0.10/api/`

[3] Documentación de `Flask`: `http://flask.pocoo.org/docs/0.10/`

[4] Error HTTP 404, not found: `http://en.wikipedia.org/wiki/HTTP_404`

[5] Scalable Vector Graphics: `http://es.wikipedia.org/wiki/Scalable_Vector_Graphics`