

Los wikis están de moda y son unas de las mejores demostraciones de la web, ¡y ni siquiera son complicados de programar!

WIKI WIKI... ¿PYTHON?

Ward Cunningham fue el pionero en el mundo de los WikiWikis. Su WikiWiki es el más antiguo del mundo:

Cunningham & Cunningham, Inc.. Lo puedes encontrar en <http://c2.com>. Ya que estamos, consulta el

wiki más famoso de la actualidad y podrás obtener datos suyos: http://en.wikipedia.org/wiki/Ward_Cunningham.

Ward_Cunningham. **POR JOSÉ PEDRO ORANTES Y JOSÉ MARÍA RUÍZ**

Ahora entremos en *c2.com* y echémosle un vistazo... ¿ya lo has visto? Entonces te habrás percatado de lo simple que es. Fue creado pensando en la simplicidad de formas, para que fuese una herramienta más que un escaparate. El objetivo es que las personas que los visitasen fueran dejando pequeños trozos de información, de manera que el sistema creciese solo.

c2.com también es muy simple en cuanto a programación y en este número replicaremos gran parte del mismo con un programa en Python que no llega a las ¡¡200 líneas!! De nuevo Python nos demuestra su potencia.

Nuestro propio WikiWiki

Vamos a crear nuestro propio WikiWiki en Python. No es una tarea tan complicada como pudiera parecer. La receta para nuestro WikiWiki es:

- 01 Mucho CGI
- 02 Un poco de Expresiones Regulares
- 03 Algo de Manipulación de ficheros
- 04 Un Servidor Web al antojo del

lector que permita CGI (casi todos lo hacen)

Con estos ingredientes vamos a programar un wiki hecho y derecho, que podremos ampliar a nuestro antojo.

NOTA: Para poder seguir el resto del artículo, convendría tener a mano el listado completo del programa *wiki.py*, que se puede descargar de [5].

Cómo funciona un WikiWiki

Este es el primera paso en el ataque de cualquier programa: saber exactamente lo que queremos. Vamos a seguir las reglas de *c2.com*. El wiki se compondrá de páginas enlazadas entre sí. Los enlaces estarán representados por palabras clave que tienen una forma especial: estarán formadas por 2 o más palabras juntas (sin espacios o separadores) cada una de las cuales comenzará en mayúscula. Esto puede parecer confuso, así que vamos a ver algunos ejemplo. Comencemos con los erróneos, las siguientes NO son palabras clave:

Hola
Una Cosa

Otra-Cosa
softwareLibre
3cerditos

Ninguna de ellas corresponde con nuestro patrón. Ahora vamos a ver unas cuantas que sí lo hacen:

HolaMundo
UnDosTres
LinuxMagazine
FreeBsd
WikiWiki (ya os podeis imaginar porqué se llaman WikiWiki y no Wiki, no serviría como palabra clave ;))

Ahora que hemos visto las palabras clave vamos a ver como se utilizan. La verdad es que no tiene mucho que explicar, simplemente las pones, es decir: *Esto es un texto de un WikiWiki para LinuxMagazine.*

En este ejemplo tanto *WikiWiki* como *LinuxMagazine* se convertirían automáticamente en palabras clave.

Las páginas de wiki serán editables, de manera que cualquiera podrá pulsar el botón "Editar" y cambiar, borrar o

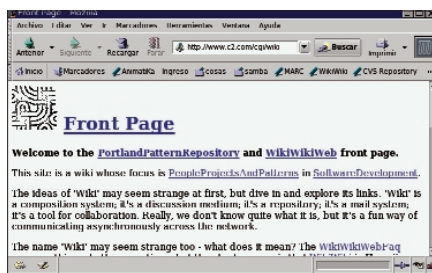


Figura 1: El padre de todos los wikis de Ward Cunningham.

añadir texto a la página. Dentro de este proceso puede introducir palabras clave que no estén presentes en el WikiWiki. De esta manera se crean nuevas páginas.

El texto que editaremos estará en formato HTML, de esta manera podremos introducir formateo, como por ejemplo listas o negritas.

Una vez editada la páginas podremos guardarla, y a partir de entonces estará disponible la versión modificada. En teoría no debería ser posible eliminar páginas, el objetivo del WikiWiki es acumular información. Como la única manera de crear nuevas páginas es añadir palabras clave a páginas ya existentes nos aseguramos de que cualquier página sea accesible desde otra. No tendremos páginas sueltas.

CGI

El “Common Gateway Interface” es un protocolo que permite a un servidor web enviar información y recibirla desde un programa exterior. Básicamente el servidor web delega la generación de las páginas en programas externos que obtienen sus parámetros de éste.

Estos parámetros pueden ser enviados de dos maneras o métodos distintos: “GET” y “POST”. Es la manera que tiene el navegador de intercambiar información con el servidor web remoto. “GET” simplemente pasa los parámetros a través de la “URL” que el navegador pasa al servidor. Lo podemos ver cuando visitamos muchas páginas, la url tiene un formato parecido a `http://www.algunaweb.org/algo.algo? variable1 = valor1 & variable2 = valor`

Lo que está ocurriendo es que el navegador le está pasando 2 variables al servidor (“variable1” y “variable2”). La URL usa el símbolo ? para separar la ruta de las variables y el símbolo & para separar variables entre sí. No es muy complicado

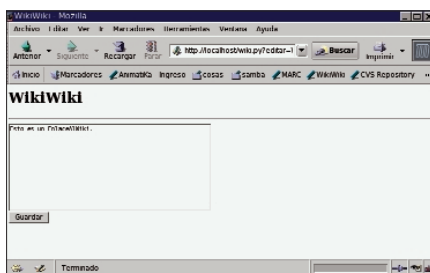


Figura 2: Nuestro formulario de edición del Python Wiki.

¿no? *POST* es distinto, el navegador y el servidor se pasan información usando el protocolo *HTTP*, de manera que no vemos pasar las variables

A efectos de nuestro programa, no nos importa usar *GET* o *POST*, lo que nos interesa es el concepto de *CGI*. Nuestro programa recogerá variables y generará las páginas dependiendo de ellas

Para ello usaremos el módulo *cgi* de Python:

```
import cgi
```

Éste módulo nos permite recoger las variables que nos pasa el servidor web. Para ello tenemos que usar la función *cgi.FieldStorage()*. Esta función nos devuelve un diccionario de objetos del tipo *FieldStorage*. Cada uno de ellos contiene la información de una de las variables. De toda esa información solo nos interesará el valor de la misma, que es accesible haciendo lo siguiente:

```
variables = cgi.FieldStorage()
valor = variables["nombre"].value
```

Así podremos obtener el valor de la variable. Al ser un diccionario lo que devuelve, podremos preguntar si está presente alguna variable en particular usando el método *has_key()*: `variables.has_key("nombre")`

Formularios HTML

Parte vital del WikiWiki es el poder editar el contenido de las páginas. Para ello HTML dispone de los formularios, que nos permiten recoger información y enviarla al servidor web.

Los formularios HTML se componen de muchos controles como por ejemplo botones, campos de texto, etiquetas o casilleros. Se diseñaron para ser lo más

parecidos posible a auténticos formularios burocráticos.

Nuestra necesidad se cubre con un formulario que incorpore un “área de texto” y un botón para guardar el texto editado. Vamos a ver lo básico.

Un formulario HTML está encerrado entre dos etiquetas `<form >` y `</form >`. La primera de ellas lleva una serie de datos que definen el comportamiento del formulario. Nosotros queremos poder controlar el método de envío (*POST* o *GET*) y el programa que recibirá los datos (*wiki.py*).

```
<form method="POST"
action="wiki.py">
...
</form>
```

Dentro del formulario se definen los controles que mostrará. Cada uno de ellos, a su vez, define una serie de parámetros. El primero de ellos será un *textarea*.

```
...
<textarea cols="55" rows="10"
name="texto"></textarea>
...
```

cols indica el número de columnas (de un carácter de ancho) y *rows* el número de filas. El parámetro *name* define el nombre con el que el contenido del *textarea* será enviado al servidor web, hemos escogido el poco original nombre de *texto*.

El segundo tipo de control que necesitaremos es *input*:

```
...
<input type="submit" value="
Editar"> </input>
<input type="hidden" name="
editar" value="algo"> </input>
...
```

Aquí vemos los dos tipos que necesitamos. El parámetro *type* se usa para especificar el tipo, pues existen varios diferentes. En nuestro caso tenemos:

submit, que genera un botón cuyo título se recoge del parámetro *value*. Este botón, al ser pulsado, envía los datos al servidor datos no relacionados con controles. En nuestro caso pasamos la variable *editar = algo*.

El resultado final se puede ver en la Figura 3.

Diseño de la Aplicación

El *WikiWiki* funcionará de la siguiente manera. Cuando se acceda por primera vez, al no estar presente ninguna variable, generaremos una página web de presentación. Esta primera página tendrá un botón que nos permitirá editarla. Para ello incluiremos un botón dentro de un formulario html, que enviará una variable *editar* con la cadena *WikiWiki* como valor. Nuestro programa se arrancará de nuevo, pero en esta ocasión verá la variable *editar* y en lugar de generar una página, buscará en el directorio en que esté la existencia de un fichero *WikiWiki.txt*. Si está presente cargará su contenido en un *textarea* que nos permitirá editarlo. Vendrá acompañado por un botón *guardar* que al ser pulsado enviará dos variables: *guardar* con el nombre de la página (*WikiWiki* en este caso) y *texto* con el texto que hemos editado. Cuando *wiki.py* detecte ambas variables procederá a guardar el contenido de *texto* en un fichero con el nombre que contenga la variable *guardar* concatenado con *.txt*.

El texto, antes de ser guardado, se analiza en busca de palabras clave. En el caso de que aparezcan se realizará la siguiente transformación en el caso de que exista la página (o sea, el fichero *WikiWiki.txt*)...

```
...WikiWiki... en ...?
<a href="wiki.py=?pagina=?
WikiWiki"> WikiWiki</a>...
```

... o se realizará la transformación...

```
...WikiWiki... en ...?
WikiWiki <a href="wiki.py=?
?pagina=WikiWiki">?</a>...
```

... en el caso de que no exista, para indicar con el ? que esa página hay que cre-

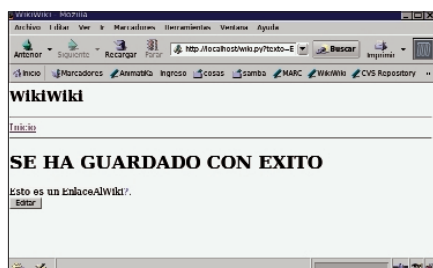


Figura 3: ¡Bien! Nuestra página ha sido guardada.

arla. Para ello solo tenemos que pinchar sobre las ?, rellenar el *textarea* y pulsar en *guardar* (al igual que antes).

Luego podemos tendremos que controlar 4 estados:

- Que no vengan variables, generamos la web de bienvenida
- Que venga la variable *pagina*, y cargaremos la página correspondiente
- Que venga la variable “editar”, y cargaremos en modo de edición la página indicada
- Que vengan las variables “guardar” y “texto”, y procederemos a guardar el texto que nos envían

Análisis del Texto

Para localizar las palabras clave emplearemos “Expresiones Regulares”. Ya se vieron en un número anterior. Lo importante ahora es localizar la expresión regular que encaja con una palabra clave. Ya vimos que están compuestas por 2 o más palabras, cada una de las cuales con la primera letra en mayúscula.

Para indicar que queremos una palabra con la primera letra en mayúscula y al menos 2 letras usamos la expresión regular: $[A-Z][a-z]^+$. Si queremos indicar que son dos palabras usaremos $[A-Z][a-z]^+ [A-Z][a-z]^+$. Pueden ser más, así que metemos la segunda parte en unos $()$ y le adosamos un $+$ para indicar que por lo menos hay una. Ya tenemos nuestra expresión regular:

$[A-Z][a-z]^+ [A-Z][a-z]^+ [A-Za-z]^*$

Ahora que ya podemos localizar las palabras clave, vamos a sustituirlas en el texto por hipervínculos a sus respectivas páginas. Para ello compilaremos la expresión regular y sacaremos una lista de ellas. Para hacerlo usamos el método *findall()*, que nos devolverá una lista con todas las cadenas que se ajusten al patrón. Pero tendremos un problema: algunas de las cadenas estarán duplicadas.

¿Cómo limpiamos una lista de Python de componentes duplicados? Se puede hacer de muchas maneras, (alguien en Internet lo ha hecho así: $reduce(lambda l, x: x not in l$

and $l.append(x)$ or $l, a, []$) siendo a la cadena, pero explicar esta simple línea nos llevaría un artículo entero). Lo que podemos hacer es emplear un poco de matemáticas. Python nos permite generar un conjunto desde una lista. Un conjunto puede contener diversos elementos, pero no importa su cantidad solo si están presentes. Veámoslo de una manera sencilla. Supongamos que tenemos la lista: $[1,1,1,3,4,5,5,6]$ y que generamos un conjunto usando la función $set([1,1,1,3,4,5,5,6])$. El resultado será el conjunto $\{1,3,4,5,6\}$. Python nos permite iterar sobre un conjunto, extrayendo un elemento distinto cada vez. Esto nos permite realizar el siguiente truco:

```
01 >>> lista =
    [1,1,1,3,4,5,5,6]
02 >>> conj = set(lista)
03 >>> for elemento in conj:
04 ...     print "Elemento: ",
    elemento
05 ...
06 Elemento: 1
07 Elemento: 3
08 Elemento: 4
09 Elemento: 5
10 Elemento: 6
11 >>>
```

Así que convertiremos nuestra lista de palabras en un conjunto y lo recorreremos de esta manera.

Como ya tenemos las palabras solo tenemos que reemplazarlas





por sus hipervínculos correspondientes. Pero, como la presentación variará dependiendo de si la página ya existe o no, tendremos que comprobar con un *IF* esa condición. Entonces reemplazaremos en el contenido de la página la palabra clave por el hipervínculo correcto. Usaremos la función *replace()* de *string*. *replace()* acepta 2 parámetros, el primero es la cadena a ser sustituida y la segunda la cadena con la que vamos a sustituir. Una sola ejecución de *replace()* realizará todas las sustituciones de una vez. Esa es la razón por la que hemos realizado el truco del conjunto: si se repitiese alguna cadena en la lista de palabras claves nos ocurriría lo siguiente. Con la primera realizaríamos el cambio:

```
...WikiWiki... -> ...<br><a href="wiki.pl\$pagina="<br>WikiWiki">WikiWiki</a>...
```

Pero al encontrar otra vez la palabra clave *WikiWiki* en la lista sustituiríamos los *WikiWiki* recién insertados por el hipervínculo de nuevo:

```
...<a href="wiki.pl\$pagina="<br>WikiWiki">WikiWiki</a>... -><br>...<a href="wiki.pl\$pagina="<br><a href="wiki.pl\$pagina="
```

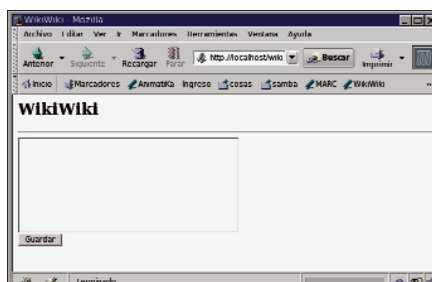


Figura 4: La creación de una página con el formulario vacío.

```
WikiWiki">WikiWiki</a>...">...<br><a href="wiki.pl\$pagina="<br>WikiWiki">WikiWiki</a>...</a>..."
```

Un desastre vamos, así que usaremos el conjunto. Con esto hemos visto la función *parseaEnlaces()* de nuestro *WikiWiki*, una función vital y muy útil.

Generación de HTML

Bueno, ahora nos encargaremos de generar el HTML. Gracias a *CGI* la tarea consiste en realidad en imprimir el código HTML usando *print*. El HTML devuelto por *CGI* tiene una peculiaridad, debe ser enviado con una etiqueta que indica el tipo de información que contiene. Además, esta etiqueta tiene que ir separada por dos retornos de carro (dos *intros*) del resto de la página. La etiqueta en cuestión es:

```
print "Content-Type:Bs<br>text/html"<br>print<br>print
```

Esos *print* sin parámetros imprimen los retornos de carro. Para facilitar el diseño del código hemos dividido la generación de la página en 3 funciones:

```
def cabecera(titulo)<br>def cuerpo(titulo, texto)<br>def editarCuerpo(titulo, texto)<br>def pie()
```

Toda página tendrá una cabecera, que imprime la parte no visible, como las etiquetas *<title>*, y un *pie*, que cerrará la página HTML (*</body> </html>*). En medio estará el cuerpo, que puede ser generado por *cuerpo* o por *editarCuerpo*. Esta segunda función lo que presenta es un formulario para editar el contenido de una página.

El texto de *cuerpo()* se genera normalmente usando una variable *contenido* de tipo *string*. Dependiendo de las circunstancias contendrá una información u otra.

Las funciones:

```
def pagina(titulo, texto)<br>def editaPagina(titulo, texto)
```

Se encargan de generar las páginas usando las 3 funciones anteriores. Esto simplifica mucho el código y nos permite generar páginas de manera sencilla.



Figura 5: La página de bienvenida de nuestro wiki.

Cada vez que tenemos que interactuar con el sistema de ficheros usamos:

```
def cargaPagina(titulo)<br>def guardaPagina(titulo, texto)
```

cargaPagina() devuelve el contenido de la página que se pasa como parámetro.

Últimos Detalles

Hemos de tener en cuenta que el programa *wiki.py* debe tener permisos de escritura en el directorio donde se encuentre, puesto que ahí será donde deposite los ficheros. También tenemos que darle permiso de ejecución (basta con *chmod +x wiki.py*)

Ya sólo nos queda copiar *wiki.py* al directorio adecuado en introducir en el navegador la URL (varía según nuestra configuración): <http://localhost/cgi-bin/wiki.py>

Aparecerá una página como la mostrada en la Figura 4

RECURSOS

- [1] <http://c2.com>
- [2] http://en.wikipedia.org/wiki/Ward_Cunningham
- [3] <http://www.python.org>
- [4] <http://cgi.resourceindex.com/>
- [5] Descarga del Listado: <http://www.linux-magazine.es/Magazine/Downloads/08>

LOS AUTORES

José Pedro Orantes Casado cursa 3º de Ingeniería Técnica en Informática de Sistemas y desde hace varios años utiliza linux como herramienta de trabajo y como sistema de escritorio. Jose María Ruíz está realizando el Proyecto Fin de Carrera de Ingeniería Técnica en Informática de Sistemas y lleva mas de 7 años usando y desarrollando Software Libre, y desde hace 2 en FreeBSD