Apuntes de web2py

Sergio Alvariño salvari@gmail.com

Agosto-2019

Resumen

Apuntes de web2py: Un framework para desarrollo de aplicaciones web

Índice general

1 Introducción

web2py es un framework para facilitar el desarrollo de aplicaciones web escrito en Python.

web2py funciona correctamente en Python 3. Su curva de aprendizaje no es tan empinada como la de Django y en muchos sentidos es más moderno que

Django.

web2py está basado (no estrictamente) en el modelo MVC

web2py incorpora *Bootstrap 4*

1.1 Referencias

- Evolución del modelo MVC
- Fat models and thin controllers
- Crítica del mantra

2 Empezar rápido

2.1 Instalación

Vamos a ver el proceso de instalación de una instancia de web2py en modo *standalone*. Normalmente uso web2py instalado de esta forma para entornos de desarrollo. Para un entorno de producción lo normal es instalar web2py tras un servidor web como *Apache* o Nginx, aunque dependiendo de la carga de trabajo y de como administres tus sistemas no tiene por que ser imprescindible y lo puedes poner en producción en modo *standalone*.

1. Creamos un entorno virtual

   Como ya hemos comentado ***web2py*** funciona ya en Python 3. Además con Python nunca está de mas encapsular nuestras pruebas y desarrollos en un entorno virtual.[1] Así que creamos el virtualenv que llamaremos *web2py*:

   ```
   mkvirtualenv -p `which python3` web2py
   ```

2. Bajamos el programa de la web de Web2py y descomprimimos el framework:

   ```
   # creamos un directorio (cambia el path a tu gusto)
   mkdir web2py_test
   cd web2py_test

   # bajamos el programa de la web y descomprimimos
   ```

[1] Los siguientes comandos asumen que tienes instalado *virtualenvwrapper* como recomendamos en la guía de postinstalación de Linux Mint, si no lo tienes tendrás que crear un virtualenv con los comandos tradicionales

```
wget https://mdipierro.pythonanywhere.com/examples/static/web2py_src.zip

# opcionalmente borramos el zip, sería mejor guardarlo
# por si queremos hacer nuevas instalaciones
rm web2py_src.zip
```

3. Generamos certificados para el protocolo *ssl*:

   Para usar con comodidad web2py conviene que nos generemos unos certificados para gestionar el ssl:

   ```
   # nos movemos al directorio de web2py
   cd web2py

   openssl genrsa -out server.key 2048
   openssl req -new -key server.key -out server.csr

   Country Name (2 letter code) [AU]:ES
   State or Province Name (full name) [Some-State]:A Coruna
   Locality Name (eg, city) []:A Coruna
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:BricoLabs
   Organizational Unit Name (eg, section) []:Division de Hackeo
   Common Name (e.g. server FQDN or YOUR name) []:testServer@bricolabs.cc
   Email Address []:contacto@bricolabs.cc

   Please enter the following 'extra' attributes
   to be sent with your certificate request
   A challenge password []:secret1t05
   An optional company name[]:Asociacion BricoLabs
   ```

   Y ahora ejecutamos:

   ```
   openssl x509 -req -days 365 -in server.csr \
   -signkey server.key -out server.crt
   ```

4. Arrancamos el servidor:

   Ahora deberíamos tener los ficheros `server.key`, `server.csr` y `server.crt` en el directorio raiz de web2py, una vez generados estos ficheros podemos arrancar el servidor con los siguientes parámetros (recuerda activar el entorno virtual si no lo tienes activo):

   ```
   python web2py.py -a 'admin_password' -c server.crt -k server.key \
   -i 0.0.0.0 -p 8000
   ```

Y ya podemos acceder nuestro server en la dirección https://localhost:8000

5. Servidor de base de datos.

   Para usar ***web2py*** es imprescindible tener acceso a un servidor de base de datos. Podemos usar MySQL o MariaDB por ejemplo. Pero para empezar rápidamente vamos a tirar de SQLite, un servidor fácil de instalar potente y versátil. Es importante usar la versión 3 que introduce grandes mejoras sobre el antiguo *SQLite*

   ```
   sudo apt install sqlite3
   ```

Y ahora si que ya tenemos todo listo para empezar a usar ***web2py***. Podemos crear nuestra primera aplicación.

2.1.1 Los detalles tenebrosos

Si tienes mucha prisa por aprender web2py puedes saltarte esta sección e ir directamente a la sección siguiente

Si por el contrario quieres entender exactamente que hemos hecho para poder arrancar el ***web2py*** este puede ser el primer paso.

¿Qué es un *virtualenv*? Python nos permite definir *virtualenv*. Un *virtualenv* es un entorno python aislado. Todos los *virtualenvs* están aislados entre si y mejor todavía son independientes del python del sistema. Esto te permite tener multiples entornos de desarrollo (o producción) cada uno con distintas versiones de python y diferentes librerias python instaladas en cada uno de ellos, o quizás diferentes versiones de las mismas librerias.

¿Que es *virtualenvwrapper*? Es un frontend para usar *virtualenv*, la herramienta nativa de python para gestionar *virtualenvs*. Es completamente opcional, aunque a mi me parece muy cómoda.

¿Qué es todo eso de los certificados? ***web2py*** viene preparado para usar *https* (estas siglas tienen varias interpretaciones: *HTTP over TLS*, *HTTP over SSL* o *HTTP Secure*). *https* usa comunicaciones cifradas entre tu navegador y el servidor web para garantizar dos cosas: que estás accediendo al auténtico servidor y que nadie este interceptando la comunicación entre navegador y servidor.

Para usar *https* hay que hacer varias cosas:

- Generar un CSR (Certificate Signing Request)

- Obtener con ese CSR un certificado SSL de una autoridad certificadora (CA)
- O alternativamente generar nosotros un certificado a partir del CSR

Lo que hemos hecho con los comandos *openssl* ha sido:

- Generar un par de claves (privada y pública) para nuestro servidor (`server.key`)
- Generar con esa clave un CSR (el CSR lleva la información que le hemos metido de nuestro servidor y la clave pública)
- Generar un certificado firmándolo nosotros mismos con esa misma clave como si fueramos la autoridad certificadora.

Esto nos vale para arrancar ***web2py*** aunque nuestro navegador nos dará una alerta de riesgo de seguridad por que no reconoce a la CA.

Más info de *openssl*

2.2 Nuestra primera aplicación

Vamos a crear nuestra primera aplicación en web2py.

Si has seguido los pasos de la sección anterior ya tienes el ***web2py*** funcionando y puedes seguir cualquiera de los tutoriales que hay en la red para aprender.

En esta guía vamos a ver la creación de una aplicación paso a paso. Crearemos una aplicación de inventario para el material de la Asociación BricoLabs, empezando por una funcionalidad sencilla y añadiendo cosas según se nos ocurran.

Crea una aplicación desde el interfaz de administración, en nuestro caso la llamaremos ***cornucopia***.

Nuestro ***web2py*** nace con algunas aplicaciones de ejemplo creadas, de hecho la pantalla inicial es una de ellas la aplicación “Welcome” o “Bienvenido” (dependerá del lenguaje por defecto de tu navegador).

Para crear nuestra aplicación ***cornucopia***:

- Vamos al botón **admin** en la pantalla principal.
- Metemos la password con la que hemos arrancado el ***web2py*** en la linea de comandos.
- Desde la ventana de administración creamos nuestra nueva aplicación

Inmediatamente nos encontraremos en la ventana de diseño de nuestra nueva aplicación. ***web2py*** nos permite diseñar completamente nuestra aplicación

desde aquí, ni siquiera necesitaremos un editor de texto (aunque nada impide usar uno, desde luego).

2.2.1 `private/appconfig.ini`

El primer fichero que vamos a examinar es `private/appconfig.ini` La sección `private` debería estar abajo de todo en la ventana de diseño.

En la sección `[app]` del fichero podemos configurar el nombre de la aplicación y los datos del desarrollador.

En la sección `[db]` fichero configuramos el motor de base de datos que vamos a usar en nuestra aplicación. Por defecto viene configurado *sqlite* así que no vamos a tener que cambiar nada en este sentido.

En la seccion `[smtp]` podemos configurar el gateway de correo que usará la aplicación para enviar correos a los usuarios. Por defecto viene viene la configuración para usar una cuenta de gmail como gateway, solo tenemos que cubrir los valores de usuario y password y la dirección de correo.

2.2.2 El Modelo

En la parte superior de la ventana de diseño (o edición) de nuestra aplicación tenemos la sección `Models`

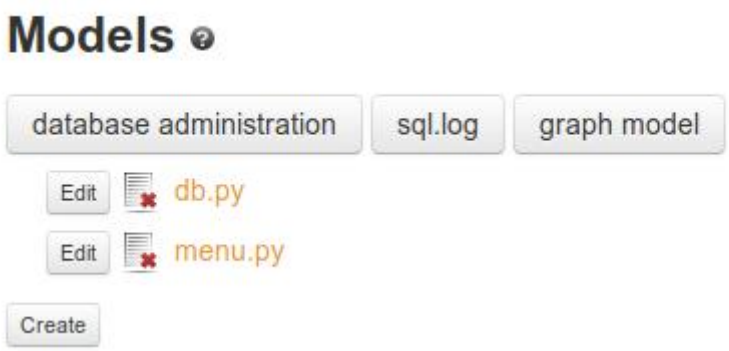

Figura 1: Menú Modelos

web2py se encarga de crear las tablas necesarias en la base de datos que le hayamos indicado que use.

Al crear la aplicación **_web2py** ha creado en la base de datos todas las tablas relacionadas con la gestión de usuarios y sus privilegios.

Si echamos un ojo al modelo gráfico (*Graphs Models*) veremos las tablas que ***web2py*** ha creado por defecto y las relaciones entre ellas.

Si vemos el log de comandos de sql (*sql.log*) veremos los comandos que ***web2py*** ha ejecutado en el motor de base de datos.

Y por último si vemos *database administration* podremos ver las tablas creadas en la base de datos, e incluso crear nuevos registros en esas tablas.

También podemos echar un ojo al contenido del fichero `db.py` o `menu.py` pero por el momento **no** vamos a modificar nada en esos ficheros.

3 Secciones en el futuro

3.1 web2py y git

3.2 Instalación con nginx

3.3 Certificados let's encrypt

Para usar MySQL como motor de base de datos: Editamos el fichero applications/pyfinder/pr

```
; App configuration
[app]
name        = PyFinder
author      = Sergio Alvariño <sergio.alvarino@vodafone.com>
description = TxFinder en Web2Py
keywords    = Thope, TxFinder, web2py, python, framework
generator   = Web2py Web Framework

; Host configuration
[host]
names = localhost:*, 127.0.0.1:*, *:*, *

; db configuration
[db]
; uri       = sqlite://storage.sqlite
uri         = mysql://dbUser:dbPass@localhost/dbName

migrate   = true
pool_size = 10 ; ignored for sqlite
```

```
; smtp address and credentials
[smtp]
server = smtp.gmail.com:587
sender = salvari@gmail.com
login  = username:password
tls    = true
ssl    = true

; form styling
[forms]
formstyle = bootstrap3_inline
separator =
```

Editamos el fichero applications/pyfinder/models/db.py Tenemos que asegurarnos de editar

```
db = DAL(myconf.get('db.uri'),
         pool_size=myconf.get('db.pool_size'),
         migrate_enabled=myconf.get('db.migrate'),
         check_reserved=['mysql'])
#         check_reserved=['all'])
```

Creamos un fichero db_custom.py en el directorio: applications/pyfinder/models El fichero

IMPORTANTE: en cada tabla crear el campo id de tipo integer, es para uso interno de web2py

IMPORTANTE: especificar migrate FALSE al final en todas las tablas externas

Ejemplo de contenido del fichero db_custom.py

db.define_table('afoxtfo', Field('id', 'integer'), Field('opti_of_connection_id' , 'string'), Field('afo' , 'string'), Field('afo_fiber' , 'string'), Field('opti_cable_id' , 'string'), Field('tfo' , 'string'), Field('tfo_fiber' , 'string'), Field('cable_endpoint' , 'string'), Field('side' , 'string'), Field('state' , 'string'), Field('loaddate' , 'string'), migrate = False);

db.define_table('physical', Field('id', 'integer'), Field('circuit_id', 'string'), Field('circuit_name', 'string'), Field('line_type_code', 'string'), Field('mux_type_code', 'string'), Field('vendor_code', 'string'), Field('line_of_sight_distance', 'string'), Field('distance', 'string'), Field('carrier_circuit_name', 'string'), Field('circuit_activation_date', 'string'), Field('circuit_deactivation_date', 'string'), Field('circuit_desconection_date', 'string'), Field('origination_tributary', 'string'), Field('destination_tributary', 'string'), Field('circuit_medium_type', 'string'), Field('phys_conn_bandwidth',

‘string’), Field(‘circuit_state_code’, ‘string’), Field(‘from_user_site_id’, ‘string’), Field(‘from_map_site_id’, ‘string’), Field(‘from_site_name’, ‘string’), Field(‘from_zone’, ‘string’), Field(‘from_node_id’, ‘string’), Field(‘from_user_node_id’, ‘string’), Field(‘from_node_class_code’, ‘string’), Field(‘from_node_technology_code’, ‘string’), Field(‘from_node_state_code’, ‘string’), Field(‘from_phys_node_id’, ‘string’), Field(‘from_node_type_code’, ‘string’), Field(‘from_shelf_no’, ‘string’), Field(‘from_user_shelf_id’, ‘string’), Field(‘from_slot_no’, ‘string’), Field(‘from_slot_num_text’, ‘string’), Field(‘from_card_id’, ‘string’), Field(‘from_card_type_code’, ‘string’), Field(‘from_user_card_id’, ‘string’), Field(‘from_port_no’, ‘string’), Field(‘from_port_id’, ‘string’), Field(‘to_user_site_id’, ‘string’), Field(‘to_map_site_id’, ‘string’), Field(‘to_site_name’, ‘string’), Field(‘to_zone’, ‘string’), Field(‘to_node_id’, ‘string’), Field(‘to_user_node_id’, ‘string’), Field(‘to_node_class_code’, ‘string’), Field(‘to_node_tech_code’, ‘string’), Field(‘to_node_state_code’, ‘string’), Field(‘to_phys_node_id’, ‘string’), Field(‘to_node_type_code’, ‘string’), Field(‘to_shelf_no’, ‘string’), Field(‘to_user_shelf_id’, ‘string’), Field(‘to_slot_no’, ‘string’), Field(‘to_slot_num_text’, ‘string’), Field(‘to_card_id’, ‘string’), Field(‘to_card_type_code’, ‘string’), Field(‘to_user_card_id’, ‘string’), Field(‘to_port_no’, ‘string’), Field(‘to_port_id’, ‘string’), Field(‘loaddate’, ‘string’), Field(‘create_user_id’, ‘string’), Field(‘create_date’, ‘string’), Field(‘modified_user_id’, ‘string’), Field(‘modified_date’, ‘string’), migrate = False) db.define_table(‘segment’, Field(‘id’, ‘integer’), Field(‘virtual_link_id’, ‘string’), Field(‘path_name’, ‘string’), Field(‘origination_route_path’, ‘string’), Field(‘destination_route_path’, ‘string’), Field(‘path_type’, ‘string’), Field(‘protection_type’, ‘string’), Field(‘hop_no’, ‘string’), Field(‘physical_conn’, ‘string’), Field(‘physical_conn_id’, ‘string’), Field(‘physical_conn_media’, ‘string’), Field(‘physical_conn_bandwidth’, ‘string’), Field(‘logical_conn_id’, ‘string’), Field(‘logical_conn’, ‘string’), Field(‘logical_conn_type’, ‘string’), Field(‘origination_node_id’, ‘string’), Field(‘origination_node’, ‘string’), Field(‘origination_node_name’, ‘string’), Field(‘origination_card’, ‘string’), Field(‘origination_site_id’, ‘string’), Field(‘origination_site’, ‘string’), Field(‘origination_site_name’, ‘string’), Field(‘origination_site_latitude’, ‘string’), Field(‘origination_site_longitude’, ‘string’),

Receta para instalar desde cero bajando el repo de github Checlist

```
Crear bases de datos en MySQL
    txdb
    txdbnew
    txdbold
Dar privilegios a
```