Práctica 5: Instalación y primer proyecto con Django

¿Qué es Django?

Django (gdh/ˈdʒæŋgoʊ/jang-goh) es un framework para aplicaciones web gratuito y open source, escrito en Python. Es un WEB framework - un conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápido.

Cuando estás construyendo un sitio web, frecuentemente se necesitan un conjunto de componentes similares: una manera de manejar la autenticación de usuarios (registrarse, iniciar sesión, cerrar sesión), un panel de administración para tu sitio web, formularios, una forma de subir archivos, entre otras cosas.

Los frameworks existen para ahorrarte tener que reinventar la rueda y ayudar a aliviar la carga cuando se construye un sitio.

¿Por qué necesitas un framework?

Para entender para qué es Django, necesitamos mirar más de cerca a los servidores. Lo primero es que el servidor sepa que quieres que te sirva una página web.

Imagina un buzón (puerto) el cual es monitoreado para cartas entrantes (peticiones). Esto es realizado por un servidor web. El servidor web lee la carta, y envía una respuesta con una página web. Pero cuando quieres enviar algo, tienes que tener algún contenido. Y Django es algo que te ayuda a crear el contenido.

¿Qué sucede cuando alguien solicita una página web de tu servidor?

Cuando llega una petición a un servidor web, ésta es pasada a Django, el cual intenta averiguar lo que realmente es solicitado. Toma primero una dirección de página web y trata de averiguar qué hacer. Esta parte es realizada por el `urlresolver` de Django (ten en cuenta que la dirección de un sitio web es llamada URL - Uniform Resource Locator - así que el nombre `urlresolver` tiene sentido). Este no es muy inteligente - toma una lista de patrones y trata de encontrar la URL. Django comprueba los patrones de arriba hacia abajo y si algo coincide entonces Django le pasa la solicitud a la función asociada (que se llama **vista**).

Imagina a un cartero llevando una carta. Él está caminando por la calle y comprueba cada número de casa con el que está en la carta. Si coincide, él deja la carta allí. Esto es similar al funcionamiento del `urlresolver`.

En la función de **vista** se hacen todas las cosas interesantes: podemos mirar a una base de datos para buscar alguna información. ¿Tal vez el usuario pidió cambiar algo en los datos? Como una carta diciendo "Por favor cambia la descripción de mi trabajo." La vista puede comprobar si tienes permitido hacer eso, entonces actualizar la descripción del trabajo para ti y devolverte un mensaje: "¡hecho!". Entonces la **vista** genera una respuesta y Django puede enviarla al navegador del usuario.

Por supuesto, la descripción anterior está sumamente simplificada, pero no necesitas saber todas las cosas técnicas *aun*. Tener una idea general es suficiente.

Así que en lugar de meternos demasiado en los detalles, comenzaremos creando algo con Django y aprenderemos todas las piezas importantes en el camino.

Instalación de Django

Entorno virtual

Antes de instalar Django, instalaremos una herramienta extremadamente útil que ayudará a mantener tu entorno de desarrollo ordenado en tu computadora. Así que, vamos a crear un entorno virtual (también llamado un `virtualenv`). Aislará la configuración Python/Django con base en cada proyecto, lo que significa que cualquier cambio que realices en un sitio web no afectará a otros que también estés desarrollando en la misma máquina.

Todo lo que necesitas hacer es encontrar un directorio en el que desees crear el `virtualenv`; tu directorio `home`, por ejemplo. Para está práctica usaremos un nuevo directorio `'django-daw'` en tu directorio `home`:

```
$ mkdir django-daw
$ cd django-daw
```

Haremos un `virtualenv` llamado `myvenv`. El comando general estará en el formato:

```
$ python3 -m venv myvenv
```

Ahora bien, `myvenv` es el nombre de tu `virtualenv`. Puedes usar cualquier otro nombre, pero mantén el uso de minúsculas y no incluyas espacios. También es una buena idea mantener el nombre corto.

Trabajar con `virtualenv`

Este comando anterior creará un directorio llamado `myvenv` que contiene nuestro entorno virtual (básicamente un montón de archivos y carpetas). Todo lo que queremos hacer ahora es iniciarlo ejecutando:

```
$ source myvenv/bin/activate
```

> Nota: a veces el comando source podría no estar disponible. En esos casos trata de hacerlo esto: `$ . myvenv/bin/activate`

Sabrás que tienes `virtualenv` iniciado cuando veas que aparece este mensaje en la consola:

```
(myvenv) ~/django-daw$
```

Cuando trabajes en un entorno virtual, python automáticamente se referirá a la versión correcta, de modo que puedes utilizar `python` en vez de `python3`.

Tenemos todas las dependencias importantes en su lugar.

Instalar Django

Ahora que tienes tu `virtualenv` iniciado, puedes instalar Django.

Primero, verifiquemos que tenemos la ultima versión de `pip` instalada, este es el software que utilizamos para instalar Django:

```
(myvenv) ~$ pip install --upgrade pip
```

Después ejecuta `pip install django~=1.11.0` (nota que usamos una tilde seguida de un signo de igual: `~=`) para instalar Django.

```
(myvenv) ~$ pip install django~=1.11.0
Collecting django~=1.11.0
  Downloading Django-1.11.3-py2.py3-none-any.whl (6.8MB)
Installing collected packages: django
Successfully installed django-1.11.3
```

Eso es todo. Ahora estás listo (por fin) para crear una aplicación Django.

Tu primer proyecto en Django

Vamos a crear un pequeño blog.

El primer paso para crearlo es iniciar un nuevo proyecto en Django. Básicamente, esto significa que podrás correr algunos scripts proporcionados por Django que crearán el esqueleto de un proyecto para nosotros: un montón de directorios y archivos que vamos a utilizar más adelante.

Los nombres de algunos archivos y directorios son muy importantes para Django. No deberías renombrar los archivos que estamos a punto de crear. Moverlos a un lugar diferente tampoco es una buena idea. Django tiene que mantener una cierta estructura para ser capaz de encontrar cosas importantes.

> Recuerda correr todo en el `virtualenv`. Si no ves un prefijo `(myvenv)` en tu consola necesitas activar tu `virtualenv`. Expliqué cómo hacer eso en la parte de Instalación de Django en la sección Trabajar con `virtualenv`. Puedes hacerlo escribiendo el siguiente comando: `source myvenv/bin/activate`.

En tu consola de Linux debes ejecutar el siguiente comando. **No te olvides de añadir el punto `.` al final**:

```
(myvenv) ~/django-daw$ django-admin startproject mysite .
```

> El punto `.` es crucial porque indica al script que instale Django en el directorio actual (para el cual el punto es una referencia corta) **Nota** Cuando escribas el comando de arriba, recuerda que solo escribes la parte que comienza con `django-admin`. La parte `(myvenv) ~/django-daw$` que se muestra es solo un ejemplo del prompt que estará esperando en tu línea de comandos.

El script `django-admin.py` creará los archivos y directorios para ti. Ahora deberías tener una estructura de directorios parecida a esto (usa el comando `tree .`):

```
django-daw
├── manage.py
└── mysite
        settings.py
        urls.py
        wsgi.py
```

```
        __init__.py
```

El script `manage.py` ayuda con la administración del sitio. Con ello podremos iniciar un servidor web en nuestro ordenador sin necesidad de instalar nada más, entre otras cosas.

El archivo `settings.py` contiene la configuración de tu sitio web.

¿Recuerdas cuando hablamos de un cartero que debía comprobar donde entregar una carta? El archivo `urls.py` contiene una lista de los patrones utilizados por `urlresolver`.

Ignoremos los otros archivos por ahora - no los cambiaremos. Lo único que debes recordar es no borrarlos por accidente.

Cambiando la configuración

Vamos a hacer algunos cambios en `mysite/settings.py`. Abre el archivo usando el editor de código de tu preferencia.

> **Nota** Ten en cuenta que `settings.py` es un archivo normal, como cualquier otro. Puedes abrirlo desde dentro de un editor de texto usando la opción "file -> open" desde el menu. Esto debería de abrir la ya conocida ventana en la que puedes navegar e ir a seleccionar `settings.py` y abrirlo. De manera alternativa puedes abrir el archivo desde el explorador de archivos de tu sistema operativo.

Sería bueno tener el horario correcto en nuestro sitio web. Ve a la lista de usos horarios de Wikipedia y copia tu zona horaria (TZ). Por ejemplo, `America/Tijuana`.

En `settings.py`, encuentra la línea que contiene `TIME_ZONE` y modifícala para elegir tu propia zona horaria, por ejemplo:

```
TIME_ZONE = 'America/Tijuana'
```

Un código de lenguaje consiste en un lenguaje, ej. `en` de Ingles o `de` para Aleman, y un código de país, ej. `de` para Alemania o `ch` para Suiza. Quieres agregar estos código si quieres que los botones por defecto y las notificaciones de Django estén en tu lenguaje. Así tendrás un botón "Cancel" traducido al lenguaje que definas aquí. Django viene con muchas traducciones preparadas.

Cambia el código de lenguaje cambiando la siguiente línea:

```
LANGUAGE_CODE = 'es-MX'
```

También necesitaremos agregar una ruta para los archivos estáticos (aprenderemos sobre los archivos estáticos y CSS más tarde). Ve hacia abajo hasta el *final* del archivo, y justo debajo de la entrada `STATIC_URL`, agrega una nueva llamada `STATIC_ROOT`:

```
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

Cuando `DEBUG` is `True` y `ALLOWED_HOSTS` esta en blanco, el host es validado contra `['localhost', '127.0.0.1', '[::1]']`. Esto no hará juego con nuestro hostname en PythonAnywhere una vez que hagamos el despliegue de nuestra aplicación, así que cambiaremos la configuración a:

```
ALLOWED_HOSTS = ['localhost', '127.0.0.1', '.pythonanywhere.com']
```

Configurar una base de datos

Hay una gran variedad de opciones de bases de datos para almacenar la información de tu sitio. Utilizaremos el que viene por defecto, `sqlite3`.

Esto ya está configurado en esta parte de tu archivo `mysite/settings.py`:

```
DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        }
    }
```

Para crear una base de datos para nuestro blog, ejecutamos lo siguiente en la consola: `python manage.py migrate` (necesitamos estar en el directorio de `django-daw` que contiene el archivo `manage.py`). Si eso va bien, deberías ver algo así:

```
(myvenv) ~/django-daw$ python manage.py migrate
Operations to perform:
  Apply all migrations: auth, admin, contenttypes, sessions
Running migrations:
  Rendering model states... DONE
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
```

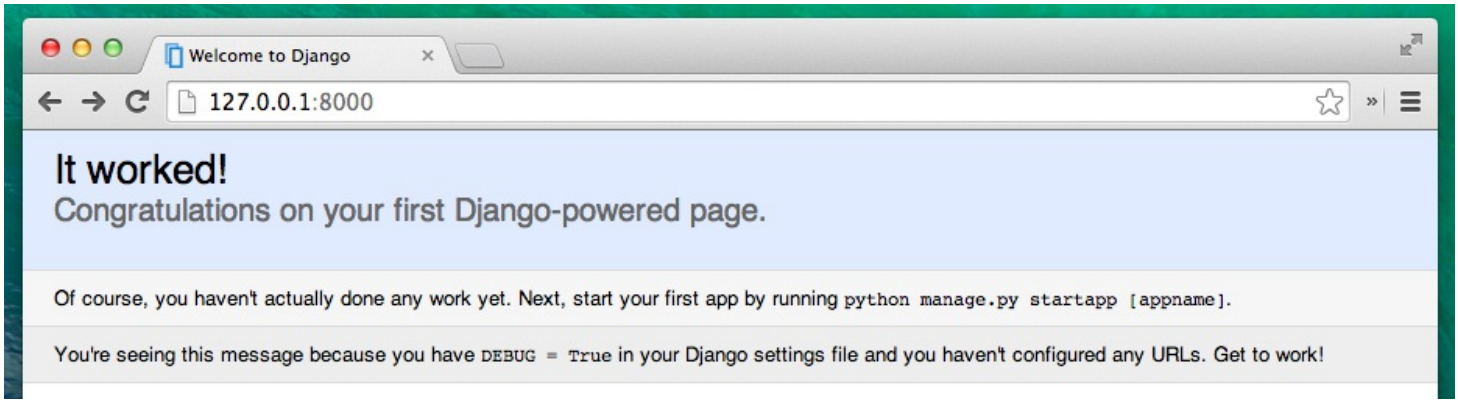
```
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying sessions.0001_initial... OK
```

Iniciar nuestro servidor web

Debes estar en el directorio que contiene el archivo `manage.py` (el directorio `django-daw`). En la consola, podemos iniciar el servidor web ejecutando `python manage.py runserver`:

```
(myvenv) ~/django-daw$ python manage.py runserver
```

Ahora todo lo que debes hacer es verificar que tu sitio esté corriendo - abre tu navegador (Firefox, Chrome, Safari, Internet Explorer o el que utilices) e ingresa la dirección `http://127.0.0.1:8000/`, se debe ver algo com esto:



Mientras el servidor web esta corriendo, no veras el prompt para escribir nuevos comandos. La terminal aceptara nuevo texto pero no ejecutara ningún comando. Esto es porque el servidor web corre continuamente para escuchar las peticiones entrantes.

> Vimos como funcionaban los servidores y el protocolo HTTP en la unidad uno.

Para escribir más comandos o abres una nueva terminal (y no te olvides de activar tu `virtualenv` allí también), o termina el servidor web yendo a la consola en la que está corriendo y presionando `Ctrl+C` - las teclas *Control* y *C* juntas.

Has creado tu primera aplicación web y la has ejecutado usando un servidor web.