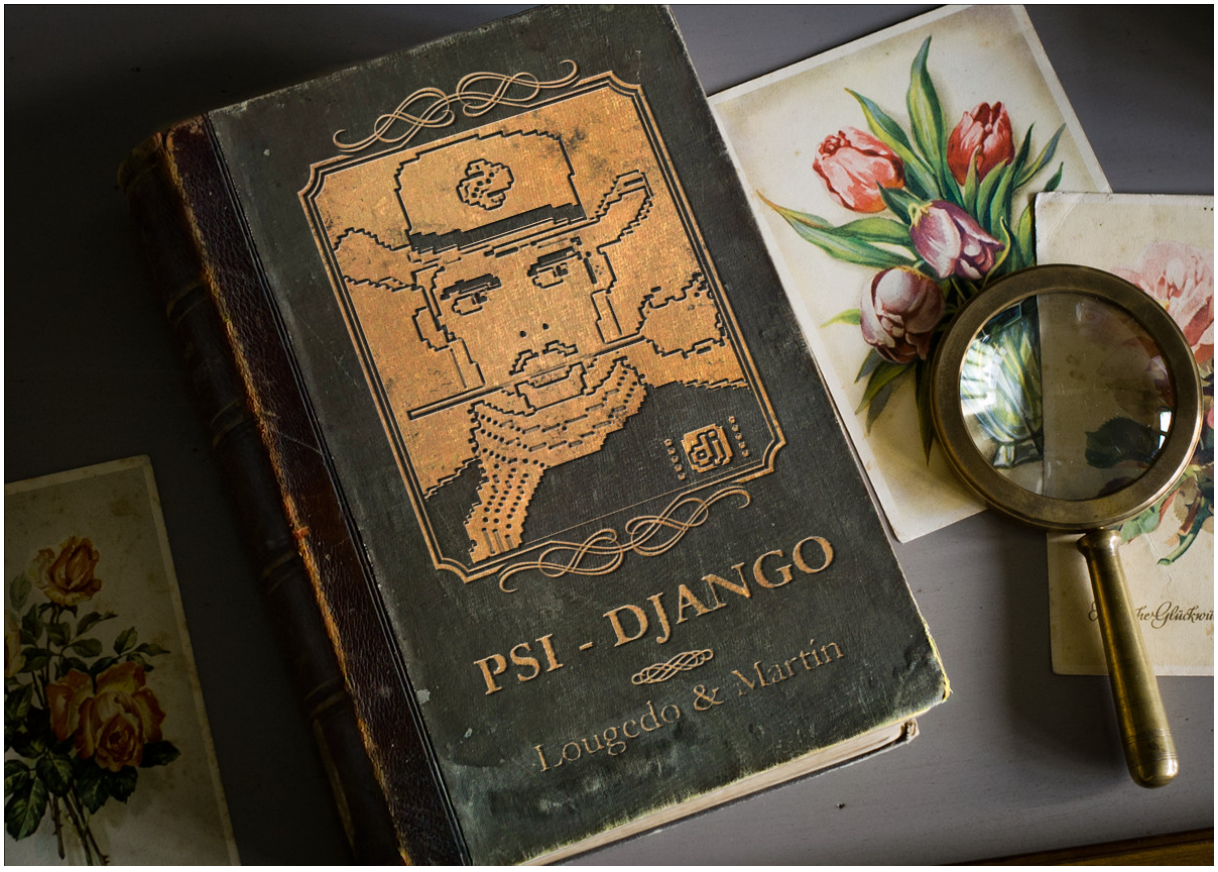
Primeros pasos con *DJANGO*

Proyecto de Sistemas Informáticos - Práctica 2

Javier Alejandro Lougedo Lorente javier.lougedo@estudiante.uam.es
Salvador Martín Barcia salvador.amaya@estudiante.uam.es



Proyecto de Sistemas Informáticos Grupo 1361 - Pareja 2
Ingeniería Informática - Universidad Autónoma de Madrid
Madrid, España
6 de octubre de 2019

Índice

Introducción

En esta práctica aprenderemos el funcionamiento de ***Django*** y nos familiarizaremos con él, además de empezar a utilizar la plataforma *Heroku*. Se nos pide emplear un repositorio en git (distinto al de la práctica anterior) en el que almacenaremos todo nuestro código para poder compartirlo cómodamente entre los dos miembros de la práctica. Este repositorio tiene el nombre de psi-1361-2-p2 y actualmente es privado.

Comenzaremos entonces, como se nos indica en la práctica, con los ejercicios de *Tango with Django*, apartados *"Django Basics"* y *"Templates and Static Media"*.

Tutorial de Django

Narraremos aquí como hemos ido realizando los distintos ejercicios del tutorial, tanto *"Django Basics"* como *"Templates and Static Media"*.

Django Basics

En primer lugar, se nos pide comprobar si hemos instalado correctamente Django. Aquí fue cuando nos dimos cuenta de un pequeño problema que teníamos, y es que no podíamos instalar la versión 2.1.5 de Django, sino que la última que nos dejaba instalar era la 1.11.24. Decidimos seguir adelante con ella, y en caso de que no funcionase adecuadamente, emplear el Django de Pycharm en Windows.

Una vez comprobado que tenemos Django correctamente instalado, procedemos a crear nuestro primer proyecto de Django. Realizamos esto empleando el siguiente comando:

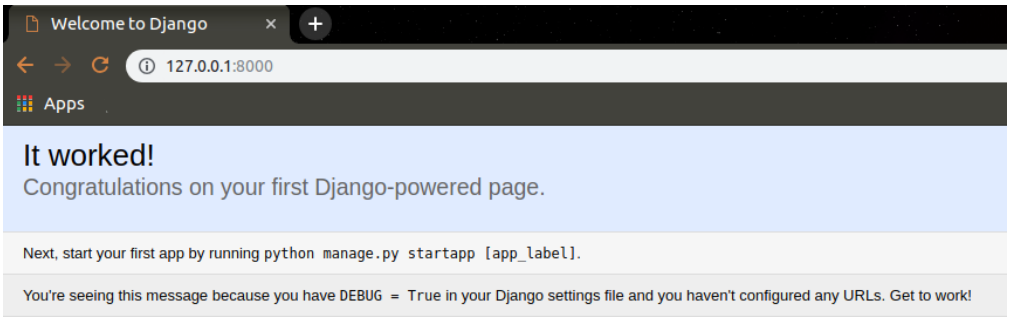
```
$ django-admin.py startproject tango_with_django_project
```

Tras crear este proyecto, se nos explica que es lo que ha sucedido y cual es la función de cada componente en el proyecto.

Posteriormente, se nos pide arrancar el servidor mediante el comando que veremos a continuación. En nuestro caso, hemos intercambiado python por python3.

```
$ python3 manage.py runserver
```

Tras esto, ignoramos el aviso que nos sale en rojo, y vamos al navegador, a la dirección http://127.0.0.1:8000/, donde nos encontrarmos con la siguiente ventana, confirmando que todo ha marchado según lo esperado:



Esta es algo distinta de la que deberíamos ver, según el tutorial, pero suponemos que se debe a la diferencia de las versiones, ya que a pesar de todo hemos recibido un mensaje de éxito.

Tras esto, creamos la app de rango para continuar con el tutorial y añadimos rango a las aplicaciones en *settings.py*. Modificamos entonces *views.py* para introducir la proción de código siguiente:

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("Rango says hey there partner!")
```

Esto funciona perfectamente. Sin embargo, más adelante, nos encontramos con nuestro primer error. Al tratar de añadir a *urls.py* las lineas de código más abajo, nos encontramos con una serie de errores por excepciones no controladas, que actualmente no somos capaces de resolver.

```
from rango import views
urlpatterns = [
    path('', views.index, name='index'),
    path('admin/', admin.site.urls),
]
```

Aquí nos encontramos con un error, y es que el utilizar la función *path()* nos daba error. Por ello, decidimos reemplazar la instrucción por *url()* como la que nos encontrábamos antes de comenzar, lo que funcionó correctamente, dándonos la salida esperada: *Rango says hey there partner!*

Posteriormente, se nos pide actualizar las *urlpatterns* tanto de urls.py de nuestro proyecto como el de rango, para poder entender correctamente el funcionamiento de las distintas urls y como se solicita la información. Acabado el tutorial, haremos los ejercicios que se nos recomiendan.

Ejercicios

En los ejercicios relativos a **Django Basics**, realizamos lo siguiente:

A. Revisamos en primer lugar el procedimiento (cosa que hemos hecho para realizar la memoria).

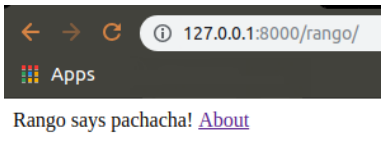
B. Creamos un nuevo método *about* con *HttpResponse:'Rango says here is the about page.'*. en views.py de rango.

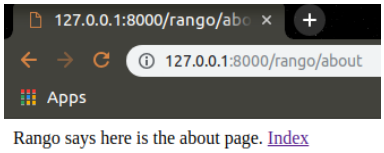
C. Mapeamos la view dentro de urls.py de rango.

D. Cambiamos el HttpResponse de index para que tenga un link a about.

E. Hacemos lo mismo con about, y deberíamos de haber acabado exitosamente.

En nuestro caso no fue así, y tardamos bastante en descubrir a que se debía esto. Posteriormente descubrimos, tras estar debuggeando y siguendo el trayecto que debería seguir Django, que al añadir la url de **about** al *urls.py* de rango, debíamos ponerlo como *path('about', views.about, name='about'),* y no como *path('', views.about, name='about'),*. Resuelto este error, todo funcionó correctamente, como se muestra a continuación.



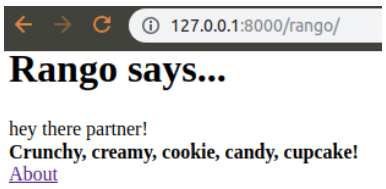


Hecho esto, pasamos a la segunda parte del tutorial.

Templates and Media Files

En esta sección aprendemos, en resumen, como funcionar con plantillas y emplear archivos multimedia. No recorreremos el tutorial tan detalladamente como con el anterior apartado ya que este es excesivamente largo en comparación.

En primer lugar, creamos nuestra primera plantilla y reemplazamos el código de views.py para index, obteniendo lo siguiente (el resultado esperado):



Tras esto añadimos nuestra primera imagen en la carpeta *static/images*, para lo cual escogimos el personaje Django y nos funcionó adecuadamente.

Posteriormente hacemos algo similar con media para poder "añadir" archivos multimedia a partir de urls y habremos acabado con el tutorial.

A continuación veremos, por tanto, los ejercicios.

Ejercicios

Tras el tutorial, llevamos a cabo lo siguiente:

A. Primero, pasamos el HTML de about a una template también.

B. Añadimos como antes una imagen a nuestro HTML. Podemos reutilizar la anterior.

C. Posteriormente, se nos pide añadir las lineas *"This tutorial has been put together by..."* y nuestros nombres. Decidimos llevarlo a cabo reemplazando *"Crunchy, creamy, cookie, candy, cupcake!"* por dicho mensaje.

D. Comprobamos que todo va bien hasta el momento, como es el caso, y añadimos ahora la foto de un gato.

A continuación, el resultado obtenido de todo el proceso:

Con esto terminamos el segundo capítulo y pasamos al de *Models and Databases*.

Testeo Semana 1

Tras haber realizado los ejercicios recomendados hasta el momento actual, se nos pide testearlos tal y como se expone en moodle. Lo realizamos y la salida es la siguiente:

```
lou@lou-ubu:~/Desktop/workspace/tango_with_django_project$ python3 ./manage.py test rango.tests.GeneralTests
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default'...
```

Para que funcionase, tuvimos que cambiar el nombre que habíamos puesto a la imagen, ya que anteriormente al llamarse *django.png*, el test no la localizaba adecuadamente (buscaba *rango.png*) y saltaba, por tanto error.

Respecto a lo del coverage, no conseguimos realizarlo correctamente. Preguntaremos en clase al respecto.

Tutorial de Modelos y BdD

Aquí veremos lo relacionado con las bases de datos y los modelos.

Models and Databases

Este será seguramente el apartado más difícil, ya que deberemos cambiar la base de datos predeterminada por la de PostgreSQL.

Efectivamente, lo ha sido. Tras mucho dolor, tortura, latigazos y sufrimiento, logramos instalar **PostgreSQL**, vincularlo a **Django** y que funcionase (para nuestra enorme sorpresa) adecuadamente. La principal complicación fue intercambiar **SQLite3** por **PostgreSQL**, ya que la información que nos daban al respecto era mínima. Por lo demás, lo que resta del proceso fue relativamente sencillo.

Tras esto, procedemos a la realización de los ejercicios.

Ejercicios

Procedemos a la realización de los ejercicios:

A. Primero añadimos los atributos pertinentes (views y likes), cosa que hacemos mediante un pequeño cambio en *models.py*, añadiendo "views = models.IntegerField(default=0)", y lo mismo con likes, dentro de la clase *Category*.

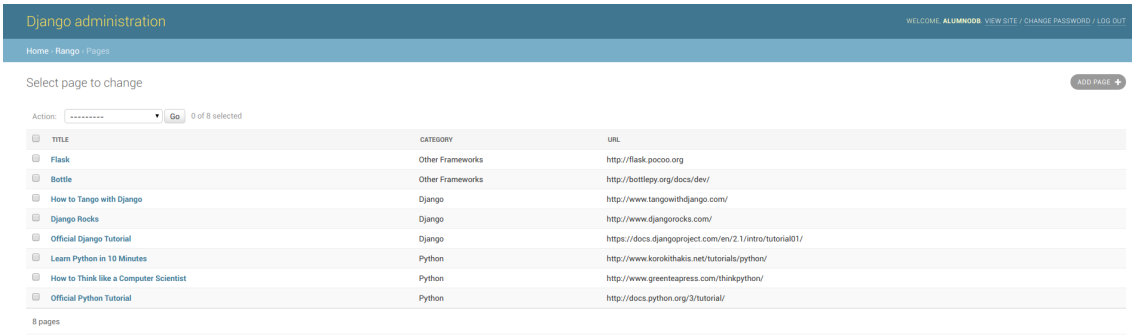
B. Migramos nuestra app.

C. Actualizamos el script para que añada los valores necesarios a views y likes, como vemos a continuación:

```
...
cats = {'Python': {'pages': python_pages, 'views': 128, 'likes': 64},
        'Django': {'pages': django_pages, 'views': 64, 'likes': 32},
        'Other Frameworks': {'pages': other_pages, 'views': 32, 'likes': 16}}
...
for cat, cat_data in cats.items():
    c = add_cat(cat, cat_data['views'], cat_data['likes'])
        for p in cat_data['pages']:
            add_page(c, p['title'], p['url'])
...
def add_cat(name, views, likes):
    c = Category.objects.get_or_create(name=name, views=views, likes=likes)[0]
    c.save()
    return c
...
```

D. Eliminamos y volvemos a crear la base de datos como en el tutorial y la poblamos con este nuevo script.

E. Por último, cambiamos el admin, para que nos muestre también las urls y su categoría, como vemos a continuación.



Hecho esto concluyen los ejercicios del tercer apartado. Vamos a por los del cuarto.

Models, Templates and Views

Realizamos el tutorial con éxito, como se podrá comprobar en nuestro código y más adelante en los tests, y procedemos a la realización de los ejercicios. Cabe mencionar en este punto que estamos utilizando pgadmin3 para manejar la base de datos cuando hace falta (eliminarla, volverla a crear y demás).

Ejercicios

A continuación, procedemos con la realización de los ejercicios:

A. Se nos pide que actualicemos el script de poblado para que añada unos pocos valores a la *views count* de cada página. Esto lo hacemos tal que así:

```
python_pages = [
    {'title': 'Official Python Tutorial',
     'url': 'http://docs.python.org/3/tutorial/',
     'views': 28},
    {'title': 'How to Think like a Computer Scientist',
     'url': 'http://www.greenteapress.com/thinkpython/',
     'views': 765},
    {'title': 'Learn Python in 10 Minutes',
     'url': 'http://www.korokithakis.net/tutorials/python/',
     'views': 654}]
```

Hacemos lo mismo con los demás que tengamos.

B. Modificamos el *index.html* para incluir las 5 páginas más vistas. Esto lo hacemos cambiando unos pequeños detalles como a continuación en *views.py*:

```
category_list = Category.objects.order_by('-likes')[:5]
pages_list = Page.objects.order_by('-views')[:5]
context_dict = {}
context_dict['boldmessage'] = 'Crunchy, creamy, cookie, candy, cupcake!'
```

```
context_dict['categories'] = category_list
context_dict['pages'] = pages_list
```

Y luego, por otro lado, añadimos lo siguiente en *index.html*:

```
{\% if pages \%}
<ul>
    <h2>Paginas mas vistas</h2>
    {\% for page in pages \%}
    <li>
        <a href="/rango/page">{{ page.title }}</a>
    </li>
    {\% endfor \%}
</ul>
{\% else \%}
<strong>There are no pages present.</strong>
{\% endif \%}
```

C. Se nos pide que incluyamos un heading a cada una de las categorías, cosa que ya habíamos realizado, como se puede observar.

D. Posteriormente añadimos un link a Index desde las páginas de categoría, cosa que logramos modificando ligeramente el fichero *categories.html*, añadiendo la linea de código siguiente: *Index*, y modificando *urls.py* si no lo habíamos hecho antes.

Testeo Semana 2

Para estos testeos íbamos a adjuntar captura como en los anteriores, pero al tratarse de unas salidas tan grandes y extensas, hemos decidido adjuntar la salida en el fichero tests2.txt, para no sobrecargar excesivamente la memoria de capturas de pantalla.