


Open · Opened 4 days ago by  **Rubén Montero**

Una app

Resumen

- Hablaremos de *apps* Django
- Crearemos nuestra primera *app* y la incluiremos en `settings.py`
- Modificaremos `views.py` para añadir una función que sirve una página
- Modificaremos `urls.py` para *mapear* una URL a dicha vista, y servir así nuestra primera página

Descripción

Una de las claves de Django es su *modularidad*. Es decir, podemos crear distintas *apps* y lanzarlas por separado.

Para comenzar el desarrollo Django, es recomendable comenzar creando una *app*.

Nuestra primera *app*

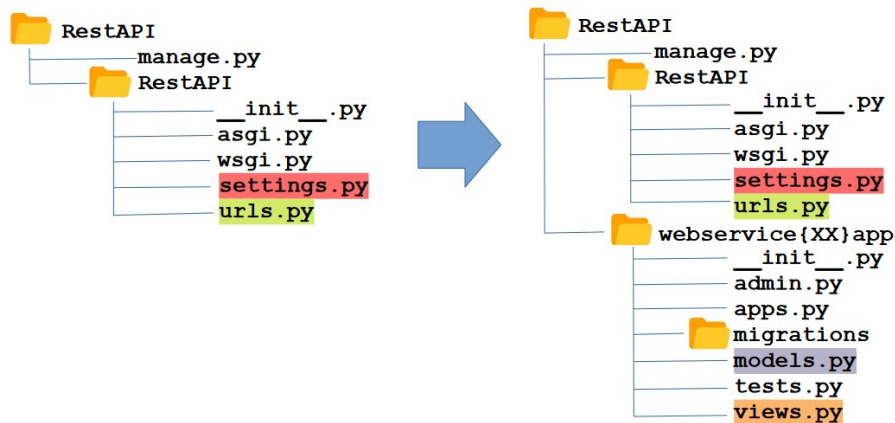
`webservice04app`

Abramos nuestro *terminal* (cmd.exe) de Windows y cambiemos el directorio activo a `python-introduction/api/RestAPI/`.

Desde aquí, volveremos a ejecutar `manage.py`, pero esta vez pasando el argumento `startapp`. Elegiremos el nombre `webservice04app` para nuestra *app*.

```
python manage.py startapp webservice04app
```

Cuando lo hagamos... ¡Python habrá creado un *esqueleto* de *app* para nosotros!



Hay que añadirla en `settings.py`

El fichero de [configuración de Django](#) es ese `settings.py` resaltado en rojo.

Contiene *información* de las *apps* que usa nuestro proyecto.

[Mucha gente desarrolla apps que podríamos decidir usar](#) y serían de utilidad. Nosotros vamos únicamente a incluir `webservice04app`, que acabamos de crear.

Abrimos `settings.py` y buscamos `INSTALLED_APPS`:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',
```

```
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
]
```

Vamos a añadir la nuestra con esta sintaxis: `nombreapp.apps.NombreAppConfig`. O sea:

```
INSTALLED_APPS = [
+   'webservice04app.apps.Webservice04AppConfig',
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
]
```

Verifiquemos que no hay nada roto

Es aconsejable ejecutar `python manage.py runserver` ahora para comprobar que no hemos roto nada.

views.py

Es una pieza clave en nuestro desarrollo.

Contiene las **funciones** que renderizarán el contenido de la web, sea este un HTML o un JSON (REST API).

Es parecido a las clases Java que creábamos en el proyecto anterior para devolver un `StringRepresentation` con `Restlet`.

De momento está vacío:

```
from django.shortcuts import render

# Create your views here.
```

Vamos a crear nuestra primera **vista web** definiendo un método:

```
from django.shortcuts import render

def my_first_view(request):
    pass
```

Intentemos devolver un `string`, a ver si funciona:

```
from django.shortcuts import render

def my_first_view(request):
    return "Hola, mundo"
```

urls.py

Esta función `my_first_view` **no** está conectada a nada. Al igual que hemos hecho con anterioridad, debemos *mapear* una URL a dicha función.

Editamos el archivo que `RestAPI/urls.py`:

```
"""RestAPI URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
```

```
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Tiene un comentario explicativo de cómo usarse.

Nosotros vamos a **importar** el módulo (fichero) `views.py` :

```
from django.contrib import admin
from django.urls import path
+from webservice04app import views

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

PyCharm lo marca en rojo

No es un problema. Como nuestro proyecto Django está en una subcarpeta, a PyCharm [le cuesta entender qué módulos están disponibles](#).

Si te molesta, puedes hacer *click* derecho en la carpeta `RestAPI/` padre (la que cuelga de `api/`) y:

- *Mark directory as > Sources root*

Arreglado.

Prosigamos con `urls.py`

Sólo falta añadir la línea que *mapea* la URL `/example` (por decisión arbitraria) con la función `my_first_view`. Así:

```
from django.contrib import admin
from django.urls import path
from webservice04app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('example', views.my_first_view),
]
```

Listo.

Si ejecutamos el servidor (con el botón **'Play'** o con `python manage.py runserver`) *parece* que todo va bien.

Podemos abrir nuestro navegador favorito y dirigirnos a la **URL**:

```
http://localhost:8000/example
```

¿Qué vemos?

Una página de error

Vaya. No pasa nada. La veremos más de una vez.

Django nos indica que *algo ha ido mal*.

¿Qué?

En concreto, quiere decir que **no** podemos devolver un simple `string` en nuestra función de vista.

¿Y qué hacemos?

Debemos usar un objeto `HttpResponse`, o, más adelante, `JsonResponse`.

Corrige el problema en `views.py` así:

```
from django.shortcuts import render
+from django.http import HttpResponse

def my_first_view(request):
-    return "Hola, mundo"
+    return HttpResponse("Hola a todo el mundo, usando un HttpResponse")
```

Visitemos de nuevo la página

¿Qué muestra tu navegador al ir a <http://localhost:8000/example> ?

¡Parece que has creado tu primera página con Django! ¡Enhorabuena!

Por último

Verifica que tu código pasa el *test* asociado a la tarea.

Haz `commit` y `push` para subir los cambios al repositorio.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 3](#) 4 days ago